Universitat Oberta de Catalunya

uoc.edu

Aplicación de algoritmos predictivos para la eficiencia en la gestión del riego.

María del Mar Pousada González

Máster en Ingeniería de Telecomunicación

Smart Cities

David Crespo García

Víctor Monzón Baeza

10 de Enero de 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Aplicación de algoritmos predictivos para la eficiencia en la gestión del riego.*
Nombre del autor:	*María del Mar Pousada González*
Nombre del consultor/a:	*David Crespo García*
Nombre del PRA:	*Víctor Monzón Baeza*
Fecha de entrega (mm/aaaa):	01/2021
Titulación:	Máster en Ingeniería de Telecomunicación
Área del Trabajo Final:	*Smart Cities*
Idioma del trabajo:	*Castellano*
Palabras clave	*Eficiencia en gestión del agua, Riego Inteligente, Sostenibilidad, Medio Ambiente, Machine Learning, Business Intelligence, IOT*

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

La gestión eficiente de los recursos hídricos debido a la previsión de escasez de agua a corto plazo ha supuesto un cambio importante, en especial en el sector agrario.

Por este motivo, han surgido los conceptos de agricultura de precisión o riego inteligente cuya finalidad es la gestión óptima del riego gracias a la tecnología.

En base a ello, el presente documento plantea el diseño de un sistema de riego inteligente desde la adquisición del dato mediante el uso de sensores hasta el tratamiento de este.

El estudio se centra en el análisis de un modelo de datos establecido mediante la aplicación de métodos predictivos. Bajo esta premisa es posible determinar la necesidad de irrigación para favorecer el uso eficiente del agua en los sistemas de regadío.

El resultado establece el diseño de un sistema autónomo que garantiza la correcta gestión de la utilización del agua en el momento necesario, ya que se aporta un valor añadido a los programadores de riego convencionales al incluir el análisis de las condiciones del terreno, condiciones atmosféricas, etc.

Finalmente, tras plantear el modelo de datos, las pruebas y simulaciones realizadas, se observa que mediante la aplicación de modelos predictivos el sistema es capaz de modelar la gestión de irrigación a partir del conjunto de datos de entrenamiento, favoreciendo la automatización, el ahorro de costes y la gestión de un uso eficiente del agua y, por consiguiente, el impacto medioambiental.

Abstract (in English, 250 words or less):

Efficient management of water resources due to forecast of short-term water shortage has meant an important change, especially in agricultural sector.

For this reason, the concepts of precision agriculture and intelligent irrigation have emerged whose purpose is the optimal irrigation management through technology.

Based on this, this document presents the design of an intelligent irrigation system from data acquisition with sensors to its treatment.

The study focuses on the analysis of a data model established by predictive methods. With this premise, it is possible to determine the need for irrigation for the efficient use of water in irrigation systems.

The result establishes the design of an autonomous system that guarantees efficient water management, since it provides added value to conventional irrigation controllers because it includes the analysis of ground conditions, atmospheric conditions, etc.

Finally, after proposing the data model, the tests and simulations obtained, it is observed that through predictive models the system models the irrigation management from the training data set, allowing automation, cost savings and the management of an efficient use of water and, consequently, the environmental impact.

Índice

Lista de figuras

1. Introducción

Este proyecto consiste en el estudio de los modelos predictivos y su aplicación en sistemas de irrigación. El objetivo es conseguir la eficiencia en la gestión del riego mediante el análisis de un modelo de datos (condiciones atmosféricas, necesidades hídricas, temperatura...) que permita establecer el uso óptimo de los recursos minimizando el consumo de agua.

1.1 Contexto y justificación del Trabajo

Recientemente, la UNESCO ha publicado un informe [1] donde se expone que el cambio climático afectará gravemente a la disponibilidad, calidad y cantidad de agua para las necesidades humanas en los próximos años.

Actualmente, el 40% de la población mundial ya reside en áreas afectadas por estrés hídrico, y en 2025, los estudios realizados indican que más de dos tercios de la población experimentará problemas de escasez de agua.

Así mismo, cabe mencionar que el consumo de agua se ha multiplicado por seis en el último siglo y crece a un ritmo de un 1% anual. De ese volumen, se estima que entre el 75 y el 80% se destina a la agricultura [2].

Teniendo en cuenta el crecimiento de la población mundial, durante los próximos años se prevé la necesidad de aumentar la producción de alimentos y, por consiguiente, la superficie de regadío.

Es por ello por lo que, la óptima gestión de los recursos hídricos supone un reto especialmente importante en el sector agrario, ya que debemos avanzar hacia un modelo más sostenible, eficiente y respetuoso con el Medio Ambiente.

En base a esta necesidad, surgen los conceptos de agricultura de precisión y riego inteligente. La tecnología y el IOT ha permitido el desarrollo de sistemas autónomos de irrigación que permiten automatizar y ajustar la programación del riego en base a las necesidades, favoreciendo el uso eficiente de los recursos (agua, energía y fertilizantes), así como el ahorro de costes para los agricultores.

Si bien es verdad que ya empiezan a existir soluciones comerciales en las que mediante programadores de riego y sensores conectados es posible analizar las condiciones necesarias que establecen la necesidad de riego tales como, temperatura, humedad, condiciones atmosféricas, evapotranspiración, etc., el objetivo del proyecto es el diseño de un sistema para mejorar la eficiencia en la gestión del riego, con especial foco en la aplicación de modelos predictivos que faciliten la toma de decisión de irrigación en función de diferentes variables.

Por tanto, este proyecto plantea la aplicación de tecnologías de análisis y explotación del dato en los sistemas de riego inteligente con el objetivo de paliar los efectos de la problemática planteada.

1.2 Objetivos del Trabajo

Este proyecto tiene como objetivo a alto nivel el estudio de métodos predictivos aplicados a la necesidad de la eficiencia en la gestión del agua mediante el riego inteligente.

En base a ello, hemos definido los siguientes objetivos específicos de cara a realizar la posterior planificación del proyecto:

- Disponer de un estado del arte con los sistemas diseñados e infraestructura necesaria para medir la eficiencia en la gestión del agua.

- Propuesta de arquitectura de sistemas para la monitorización, adquisición de datos, procesamiento y análisis de estos.

- Diseño de la prueba a partir de datasets públicos para la simulación del sistema.

- Definición del proceso de ingesta de datos y el ecosistema de aplicaciones para llevarlo a cabo.

- Comparativa y elección de los mejores algoritmos en base a los datos o variables relevantes en el ámbito del riego.

Por otro lado, cabe mencionar los objetivos didácticos:

- Aplicar los conocimientos de gestión de proyectos adquiridos en las asignaturas de Gestión Avanzada de Proyectos y Dirección Estratégica en SI/TI.

- Aplicación de los conocimientos específicos adquiridos a lo largo del Máster.

- Aprendizaje de lenguaje programación Python.

- Adquisición de conocimientos en Machine Learning y Big Data. Estudio de los distintos algoritmos predictivos, así como el proceso de aprendizaje de un sistema para el análisis del dato en la toma de decisiones en cualquier modelo de negocio.

1.3 Enfoque y método seguido

El enfoque de este proyecto será más analítico que orientado al desarrollo, ya que no consistirá en la realización de un producto nuevo, sino en diseñar y plantear la arquitectura necesaria desde la adquisición del dato hasta el tratamiento de este de un sistema de irrigación.

El proyecto está orientado de acuerdo con la metodología CRISP-DM donde se establecen las fases o guías seguidas en nuestro estudio.

Inicialmente, se realizará una investigación y comprensión del modelo de negocio que permitirá establecer la arquitectura y diseño del producto.

Una vez establecida la infraestructura y cómo se adquiere e ingesta el dato obtenido, el análisis, comprensión y preprocesado de la información será un punto relevante de cara a aplicar algoritmos predictivos y evaluar el comportamiento de nuestro sistema.

Gracias al proceso de aprendizaje, el sistema será capaz de comprender y replicar las decisiones, mejorando de esta manera los programadores de riego convencionales.

Finalmente, se realizará la simulación y pruebas, así como la representación de resultados con el fin de establecer las conclusiones del proyecto.

1.4 Planificación del Trabajo

La planificación del proyecto se ha realizado de acuerdo con las fechas establecidas para la entrega de las pruebas de evaluación continua, considerándose cada una de las PECs un hito para el proyecto.

En la Figura 1 podemos observar un diagrama de Gantt con la división de las tareas e hitos más relevantes en los que se descompone el proyecto.

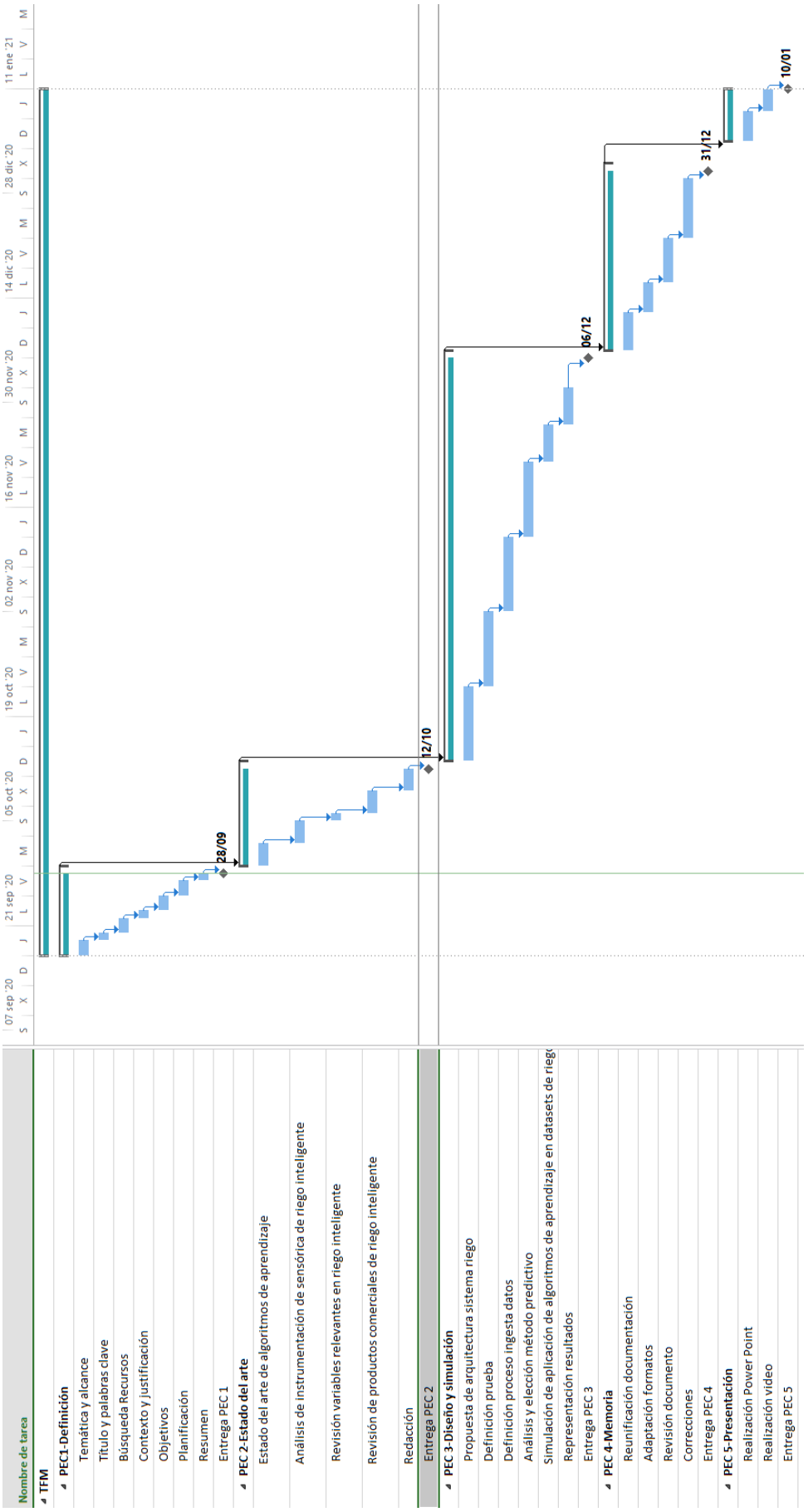

Figura 1: Diagrama de Gantt

1.5 Breve sumario de productos obtenidos

Como resultado de este proyecto se presentarán los siguientes entregables:

- Diseño del sistema de riego
- Definición de variables relevantes
- Plan de pruebas
- Resultado de la simulación

1.6 Breve descripción de los otros capítulos de la memoria

En este primer capítulo de introducción, se contextualiza y justifica la motivación de la elección del proyecto, así como la definición de los objetivos y planificación de éstos como punto de situación del alcance, temática y gestión del tiempo que deberemos tener presente a lo largo del TFM.

En el capítulo 2 se detallará el estado del arte donde conocer las diferentes alternativas existentes en el mercado para el riego inteligente, así como los productos relacionados con Machine Learning y modelos predictivos aplicados en el ámbito definido.

Así mismo, tras una búsqueda exhaustiva de la instrumentación utilizada, infraestructuras desarrolladas y análisis de los parámetros más relevantes adquiriremos una visión del producto con el objetivo de definir mejor nuestro diseño y su funcionamiento.

Posteriormente, en el capítulo 3 se definirá el diseño de sistema donde se elaborará una propuesta de infraestructura desde la captación del dato hasta el tratamiento de este para el posterior análisis de cara al funcionamiento del sistema en base a la lógica obtenida.

Una vez que se define la infraestructura del sistema, en el capítulo 4 se realizará el análisis de los distintos métodos predictivos y la elección del óptimo en el ámbito aplicado.

Finalmente, en el capítulo 5 se establecerán las pruebas y simulación realizada que servirán de apoyo en las conclusiones obtenidas y reflejadas el capítulo 6 de la memoria.

2. Estado del arte

2.1. Antecedentes

Desde tiempos remotos el hombre comprendió que el agua era indispensable para el crecimiento de las plantas y que cuando éstas carecían de suficiente humedad en el suelo, se podía facilitar su desarrollo aportándosela cada cierto período de tiempo.

Las primeras referencias al riego en agricultura se remontan al año 6000 a. C. En Egipto y Mesopotamia se comenzó aprovechando las inundaciones del Nilo o del Tigris y Eufrates que se producían determinados meses del año y se desviaban hacia los campos, dando lugar al primer riego por inundación. Una vez que la tierra estaba húmeda y se establecía el proceso de cultivo, se drenaba el agua hacia el río de nuevo.[3]

Posteriormente, cuatro siglos después, surge el primer proyecto de riego a gran escala con la construcción de presas y canales para dirigir las aguas de inundación.

Así mismo, es importante mencionar que, un milenio más tarde los romanos construyeron los famosos acueductos cuya función era el transporte del agua salvando los desniveles del terreno.

Fueron muchos los inventos posteriores, tales como el shadoof, poste grande que se balanceaba en una viga transversal con una cuerda y una cubeta atada a un extremo y un contrapeso en otro extremo, de modo que se tiraba de una cuerda para bajar el cucharón a una fuente de agua, levantándolo y balanceándolo alrededor del poste para regar los campos cuando no había inundaciones; la rueda hidráulica egipcia, el qanat (primera técnica para usar agua subterránea al construir un pozo en vertical), la rueda de agua persa (primer uso conocida de la bomba),etc.

Ya a principios de siglo, tuvo origen el riego por aspersión en el que se lanza una gran cantidad controlada de agua de manera uniforme en forma de lluvia. En los años 30, el coste de los sistemas de riego por aspersión se redujo con el desarrollo de los aspersores[4]

Así mismo, en 1960 nace el concepto de “pivote” [5], el cual consiste en un dispositivo de control que permite llevar el agua de riego hasta el cultivo gracias a que dispone de movimiento circular y de esta manera los aspersores riegan el terreno de una forma automatizada y con menor coste.

En la actualidad se realizan grandes inversiones en la construcción de grandes infraestructuras de riego, así como en la reparación de las antiguas.

Por lo tanto, es fácil observar que independientemente de la época, todas las civilizaciones se han adaptado para optimizar los recursos de los que disponían, potenciando la gestión eficiente hasta llegar a los programadores de telecontrol actuales que analizaremos en los apartados sucesivos.

2.2. Sistemas de riego

Los sistemas de riego están formados por la infraestructura necesaria que permite conducir, distribuir y racionalizar el agua de una manera eficiente a lo largo de toda la unidad de cultivo.

2.2.1. Sistemas de riego por tiempo

Actualmente, el sistema de riego más utilizado es el temporizado o programado por tiempo. Este sistema, simplemente automatiza la apertura o cierre de las electroválvulas, facilitando al regante la irrigación en una franja horaria programada o durante los intervalos diarios que sea necesario realizar el riego en función del tipo de cultivo, así como las condiciones del terreno y región geográfica, sin necesidad que éste se encuentre físicamente en la parcela.

Si bien es verdad que, aunque inicialmente se automatizó este servicio y fue un gran paso para el agricultor, en ocasiones este tipo de sistema de riego puede regar los cultivos innecesariamente.

Es importante destacar de cara a la importancia de este futuro análisis que, los programadores por tiempos o frecuencia pueden desperdiciar los recursos ya que en base a una serie de parámetros que analizaremos a continuación, la irrigación puede no ser necesaria en ese momento si el terreno ya cuenta con suficiente humedad, hay previsión de precipitaciones de acuerdo con las condiciones climáticas, etc.

2.2.2. Sistemas de riego inteligente

El riego en los cultivos ha incorporado en estos últimos años los avances tecnológicos en sus sistemas para un mejor uso del agua, lo que permite optimizar los procesos. En base a esto, el riego inteligente [6] consiste en la utilización de las tecnologías de la información y comunicación para gestionar de un modo óptimo la irrigación.

Gracias al concepto de IOT [7] o Internet de las cosas se está invirtiendo en la instalación de sensores con conexión a Internet que permiten, a través de protocolos específicos, el intercambio de información y comunicaciones, lograr un reconocimiento inteligente, así como contribuir a mejorar el rendimiento de los cultivos, la rentabilidad del agricultor y el impacto medioambiental.

Por ese motivo, los avances tecnológicos en este sector han permitido a partir de ciertas variables facilitar la toma de decisiones, y como segundo paso adicional, el Machine Learning permite predecir y mejorar la toma de decisiones a partir del entrenamiento de modelos y la detección de patrones.

De esta manera, las TIC favorecen el aumento de la eficiencia hídrica al establecer el momento concreto que se debe regar, la frecuencia del riego y el tiempo necesario para garantizar un correcto crecimiento del cultivo con el agua imprescindible y necesaria para ello.

2.2.3. Tipos de sistemas de riego inteligente

Existen dos tipos principales de sistemas de riego inteligente, atendiendo al tipo de dato que utilizan para establecer las decisiones de necesidad de irrigación de un terreno o parcela.

- Sistemas de riego inteligente basados en el clima:

Este tipo de programadores de riego basan su activación o desactivación de acuerdo con los datos climatológicos de forma que se calcule de una manera precisa la cantidad de agua necesaria mediante datos de viento, humedad, radiación solar y temperatura.

- Sistemas de riego inteligente basados en sensores de humedad:

Estos sistemas permiten medir con precisión la humedad de un punto del terreno con el fin de ser más eficientes con el consumo de agua si el suelo ya dispone de la cantidad necesaria para el crecimiento del cultivo.

Estos sistemas se pueden implementar de manera que, si la humedad está por debajo de un determinado umbral, active la programación del riego o de manera inversa, es decir, para los riegos programados se establece lógica de parada en el caso de que las condiciones del terreno alcancen el umbral de humedad óptimo establecido previamente.

2.3. Instrumentación e infraestructura

Los elementos de control de la red de riego permiten la automatización de esta, así como regular y controlar los caudales, los tiempos, etc. Los principales elementos que constituyen una instalación de riego son:

- Válvulas:
  Permiten regular y controlar las redes de distribución de agua. Existen dos tipos: válvulas de control que funcionan manualmente, y de regulación, actúan de acuerdo con un parámetro.

- Caudalímetro:
  Sensor que permite medir la cantidad de agua que atraviesa una tubería por unidad de tiempo. El caudal depende de las dimensiones de la tubería y la presión del suministro. La utilización de caudalímetros en la red de riego puede determinar el consumo de una instalación, regular el flujo actuando sobre una bomba, controlar el llenado de un depósito o controlar un sistema de riego.

- Contador:
  Mide el volumen real en m^3 acumulado que pasa a través de la red de riego y se destina para la irrigación. Es un elemento muy importante para controlar el consumo durante la campaña de riego y que el regante no se exceda.

- Electroválvulas:
  Las electroválvulas gestionan de manera automática la regulación del paso de agua tras producirse un disparo eléctrico. Las electroválvulas pueden ser de dos tipos: normalmente abiertas o normalmente cerradas, en función del estado inicial del solenoide en ausencia de alimentación.

- Presostato:
  El presostato es un dispositivo que permite controlar la apertura o cierre de un circuito eléctrico, por ejemplo, una válvula, en función de la presión ejercida sobre ese elemento.

- Pluviómetro:
  Es un dispositivo que se emplea para poder medir las precipitaciones que caen en una zona durante un periodo de tiempo. En base a la cantidad de lluvia diaria es posible establecer la necesidad de programación de riego o parada del mismo.

- Programador:
  Dispositivo de telecontrol responsable de la activación de las electroválvulas que determina el inicio de un ciclo de riego y permite automatizar la irrigación en el periodo programado o ejecutar un nuevo ciclo de riego de acuerdo con una lógica preestablecida.
  Así mismo, puede disponer de un determinado número de entradas a las que se le pueden conectar distintos tipos de sensores con el fin de captar ciertos parámetros, transmitir esos datos mediante conexión inalámbrica y posteriormente, almacenarlos en una BBDD para su posterior tratamiento, actuando en este caso como dataloggers.

- Sensores:
  Los sensores son los elementos que permiten captar datos de las diferentes magnitudes físicas que vamos a evaluar, con el objetivo de facilitar la toma de decisiones óptimas para la gestión eficiente del riego y los recursos hídricos. Dado que necesitamos analizar las condiciones climáticas y del terreno, podemos destacar la utilización de sensores de temperatura y humedad.

En los últimos años, han adquirido especial importancia las redes de sensores inalámbricos o WSN [8]. Esta tecnología está basada en dispositivos (MEMS) de baja demanda energética y procesadores muy potentes, distribuidos e interconectados de forma inalámbrica entre ellos, capaces de monitorizar el entorno mediante sensores.

Típicamente, el modelo seguido es realizar una serie de mediciones sobre el medio, transformar dicha información en digital en el propio nodo y transmitirla fuera de la red de sensores vía un elemento Gateway a una estación base, donde la información pueda ser almacenada en una BBDD y procesada para permitir su análisis.

En base a los elementos anteriormente descritos, una posible infraestructura en una red de riego podría estar organizada de acuerdo con la siguiente imagen:

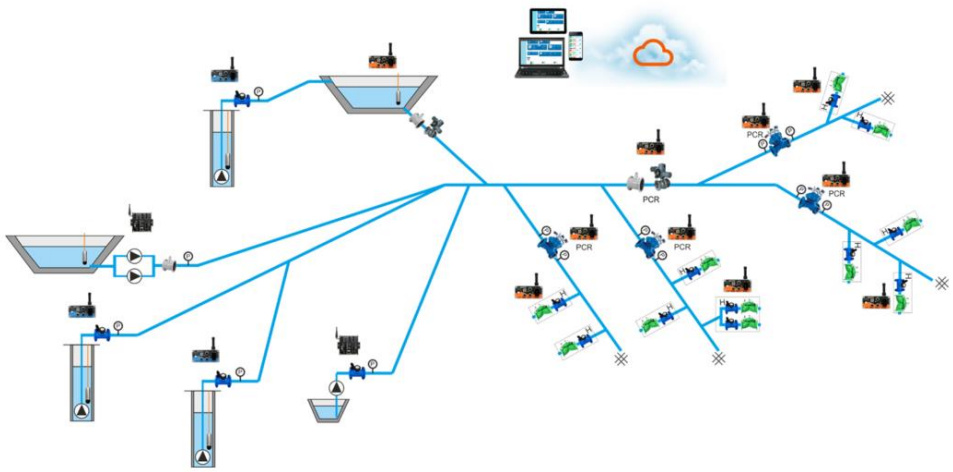

Figura 2: Infraestructura red de riego. Fuente: https://orionis-iot.com/

El agua está acumulada en balsas, pozos o depósitos y mediante estaciones de bombeo se distribuye el agua en toda la red de hidrantes o puntos de entrega. En cada parcela se dispondrá de un programador de riego con conexión inalámbrica que automatizará mediante telecontrol la irrigación mediante un SW de gestión.

Así mismo, este dispositivo podrá captar mediante sensorización distintas magnitudes físicas que aporten valor añadido al dispositivo remoto con el fin de determinar mediante métodos predictivos la necesidad de irrigación.

En relación con las comunicaciones es importante destacar la existencia de distintas tecnologías inalámbricas que se pueden utilizar para la interconexión entre dispositivos y transmisión de información:

- WIFI: El estándar de comunicaciones IEEE 802.11 es una tecnología de elevado ancho de banda muy utilizada por la mayoría de los dispositivos.
- Bluetooth: Posibilita la transmisión de datos y voz entre diferentes equipos mediante un enlace por radiofrecuencia. Es un estándar muy empleado en telefonía móvil.
- WiMAX: método de transmisión de datos a través de ondas de radio, y que utiliza frecuencias de 2'5 a 5'8 GHz.
- Sistemas de comunicación móvil: telefonía celular de cobertura GSM(2G), GPRS (2.5G), UMTS (3G), …, hasta el 5G actual.
- Comunicaciones satelitales, por ejemplo, el modem Iridium.
- LPWAN o redes de área amplia de baja potencia que constituyen una alternativa a las redes malladas. Dentro de este grupo podemos destacar Sigfox, LoRaWAN o Zigbee (estándar 802.15.4), protocolo de comunicaciones inalámbricas de bajo coste y consumo reducido, específicamente diseñado para interconectar sensores.

Finalmente, cabe mencionar que las RTUs (Remote Terminal Unit) diseñadas de cara a entrar en licitaciones de obras públicas deben ofrecer la posibilidad de comunicación a través de un driver OPC ya que es un estándar de comunicación que ofrece una interfaz común que permite que componentes de SW individuales interactúen y compartan datos. Es una solución universal, flexible y abierta a todos los dispositivos electrónicos de cualquier fabricante, por lo que facilita la sustitución de HW en caso de deterioro. [9]

2.4. Variables de análisis

Son muchos los parámetros [10] para tener en cuenta que pueden influir en la toma de decisiones en un sistema de irrigación inteligente y en función del tipo de riego utilizado, primará la elección de las variables a considerar en nuestro sistema.

Dado que el objetivo de este proyecto es la aplicación de algoritmos predictivos para conseguir una mayor eficiencia en la gestión de riego, cabe mencionar las variables más relevantes.

2.4.1. Temperatura

La temperatura ambiente es un parámetro importante, no solo en el caso de aumento de ésta ya que favorece la evaporación de agua acumulada en suelo sino, por ejemplo, para control de la helada. En función del tipo de cultivo y época del año existen programadores de riego actuales que, por medio de una sonda de temperatura, si ésta baja de cierto umbral, accionan la apertura de las electroválvulas y de esta manera reducen el riesgo de pérdida del cultivo. Es muy

habitual en el caso de plantaciones de frutales con las heladas primaverales en ciertas regiones.

2.4.2. Pluviometría

Las precipitaciones o probabilidad de precipitación estiman el volumen de lluvia acumulada en un periodo de tiempo o la probabilidad de que dicho fenómeno se produzca de acuerdo con la previsión meteorológica estimada.

Por un lado, si nuestro sistema tiene en cuenta la previsión meteorológica, podemos establecer que se van a producir precipitaciones cuando se supere el 90% de probabilidad estimado en cultivos de alto valor económico y un umbral del 60% en cultivos con poco valor económico. En general, el umbral recomendado es del 75% para garantizar un valor óptimo en la gestión del riego.

Por otro lado, cabe mencionar que no toda el agua que cae sobre la superficie del suelo es aprovechada, parte del agua de la lluvia se infiltra en el terreno y parte fluye sobre la superficie, medible mediante el coeficiente de escorrentía superficial. El agua que se encuentra en la superficie del suelo se evapora por lo que de acuerdo con la FAO [11] la precipitación efectiva o fracción de lluvia realmente disponible se estima en áreas con pendientes inferiores al 4-5% como:

$$P_e=0'8xPP-25 \text{ si } PP>75mm/mes$$
$$P_e=0'6xPP-10 \text{ si } PP<75mm/mes$$

2.4.3. Densidad aparente

Las óptimas condiciones del terreno son imprescindibles para el crecimiento del cultivo. Es por ello importante mencionar la densidad aparente [12], ya que determina la porosidad del suelo y, por tanto, el comportamiento del agua al filtrarse desde la superficie.

La densidad aparente define la capacidad del sustrato de acumular agua para favorecer el crecimiento de las plantas. Se define como la relación entre el peso del suelo seco entre el volumen total, por lo que, a mayor densidad aparente, menor porosidad.

$$\rho_a=W_{ss}/V_s$$

En base a esto, cuanto menor sea el valor de la densidad aparente, el suelo tendrá mayor capacidad de acumular humedad.

2.4.4. Humedad del suelo

La humedad del suelo es la cantidad de agua almacenada en el sustrato por volumen de tierra. Existen dos métodos para medir la humedad del suelo:

- Los métodos directos que ofrecen una mayor exactitud, pero destruyen el suelo y su ejecución lleva mucho tiempo como, por ejemplo, el método termogravimétrico o el método termovolumétrico.

- Los métodos indirectos como los sensores volumétricos que miden la cantidad de agua existente a la profundidad colocada monitorizando la constante dieléctrica; los tensiómetros que miden la intensidad de la fuerza con la que el suelo retiene el agua o los sensores de estado sólido que consisten en dos electrodos incrustados en un bloque de material poroso.

De forma genérica, podemos fijar un umbral de riego o porcentaje de humedad aprovechable que tiene que consumirse antes de volver a regar en torno al 50%.

2.4.5. Evapotranspiración

La evapotranspiración [13] es la combinación de dos procesos por los que se pierde el agua en la superficie del terreno (evaporación) y por la transpiración del cultivo.

La radiación solar, la humedad atmosférica, la temperatura ambiente y la velocidad del viento son parámetros climatológicos que influyen en el proceso de evapotranspiración.

2.5. Machine Learning y algoritmos predictivos

El Machine Learning o Aprendizaje Automático consiste en la capacidad de un dispositivo o máquina de aprender, es decir, identificar patrones complejos gracias a la ingesta de datos en su sistema, mediante la aplicación de ciertos algoritmos.[14]

Así mismo, cabe mencionar la existencia de numerosas herramientas/SW para la implementación de dichos algoritmos. El lenguaje R o Python son los más habituales para dicho fin.

Python dispone de librerías específicas como por ejemplo SciPy, que contiene NumPy y Pandas que permiten definir funciones de álgebra lineal fácilmente, así como Scikit Learn, librería de código abierto que proporciona algoritmos de aprendizaje supervisado y no supervisado ya definidos en Python.

Adicionalmente, es importante hacer alusión a la existencia de plataformas que disponen de herramientas que facilitan dicha implementación y análisis, tales como por ejemplo IBM Watson Analytics, interfaz muy intuitiva que permite subir un fichero a la nube y trabajar con él fácilmente para extraer información.

El análisis predictivo es uno de los más frecuentes del Machine Learning y consiste en la utilización de métodos matemáticos que permiten la extracción de información existente a partir de unos datos de entrada proporcionados al sistema y a partir de los cuales se puede predecir eventos o resultados futuros. [15]

Con el objetivo de realizar el análisis de datos podemos establecer dos modelos de aprendizaje:

- Modelo supervisado donde los algoritmos trabajan con datos “etiquetados”, intentando encontrar una función que, dadas las variables de entrada, se asigne la etiqueta de salida adecuada. El algoritmo se entrena con datos registrados previamente mediante un training set y su objetivo es predecir el valor de salida en el conjunto de prueba establecido o test set.

- Modelo no supervisado, utilizado cuando no se dispone de datos “etiquetados” para el entrenamiento. Solamente se dispone de los datos de entrada, pero se desconoce la salida correspondiente a un determinado input, por lo que simplemente se describe la estructura de los datos para intentar encontrar alguna característica común, como por ejemplo las tareas de clustering o agrupamiento.

Una vez realizado el análisis de los datos, a partir del conocimiento adquirido por el sistema es posible evaluar la toma de decisiones mediante dichos resultados y, por tanto, aplicado en el ámbito de la agricultura, optimizar los recursos de cara a conseguir la eficiencia en la gestión del agua. Dicho proceso se puede observar en la siguiente figura [16]

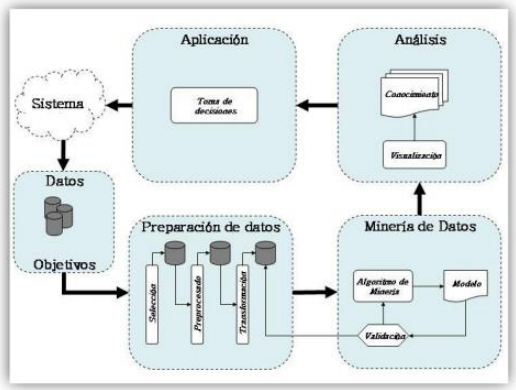

Figura 3: Proceso de análisis de los datos

En relación con la imagen anterior, la metodología CRISP-DM es el principal modelo utilizado en minería de datos. El ciclo de vida del proyecto de minería de datos consiste en seis fases principales: Comprensión del negocio, Comprensión de los datos, Preparación de los datos, Fase de Modelado, Evaluación e Implantación.

Por otro lado, cabe mencionar que existen dos tipos de modelos predictivos: de clasificación que permiten predecir la pertenencia a una clase o los modelos de regresión que permiten predecir un valor. Independientemente de ello, se pueden definir distintos tipos de algoritmos utilizados en ambos casos. Mencionaremos los más relevantes:

2.5.1. Regresión lineal

El modelo de predicción por regresión lineal analiza la relación entre la variable de respuesta y un conjunto de variables predictoras, expresando esa relación como una ecuación de una función lineal de los parámetros.

El objetivo del algoritmo es calcular la suma de las distancias al cuadrado entre los puntos que representan los datos reales y los puntos definidos por la recta estimada a partir de las variables introducidas en el modelo, de forma que la mejor estimación será la que minimice esas distancias de acuerdo con la siguiente figura [17]. En base a ello, también es conocido como método de los mínimos cuadrados. [18]

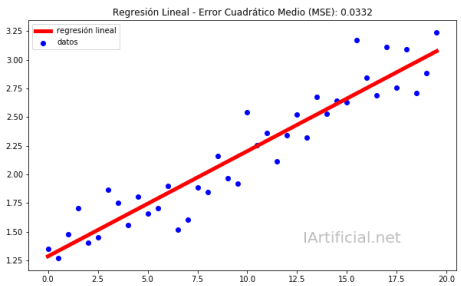

Figura 4: Regresión lineal

2.5.2. Árboles de decisión

Este modelo de clasificación permite plantear un problema y descomponerlo en distintas ramificaciones de modo que todas las opciones sean analizadas. El esquema cuantifica el coste de los resultados y las probabilidades de que ese suceso se produzca. [19]

2.5.3. Redes neuronales

Son sistemas dinámicos auto adaptativos capaces de modelar funciones complejas. Una red neuronal no necesita un algoritmo para resolver un problema, ya que ella puede generar su propia distribución de pesos en los enlaces mediante aprendizaje basado en el entrenamiento. Esta técnica es utilizada cuando se desconoce la relación entre los valores de entrada y salida. [20]

2.5.4. Máquina de vectores de soporte (SVM)

Este modelo transforma el espacio de entradas en un espacio característico de dimensión superior y en base a él construye la función de clasificación lineal óptima dentro de ese nuevo espacio. Se usan para detectar patrones complejos de datos agrupando, ordenando y clasificando los datos. [21]

2.5.5. K-vecinos más cercanos

Consiste en reconocer patrones mediante la etapa de entrenamiento y cuando se introduce un nuevo dato en el sistema éste se clasifica de acuerdo con el elemento más cercano. [22]

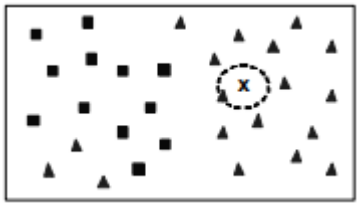

Figura 5: Predicción de acuerdo con el vecino más cercano

2.6. Sistemas comerciales existentes

Son muchos los sistemas existentes hasta la fecha que se comercializan en función de las necesidades de la irrigación.

Actualmente en el mercado podemos encontrar, desde dispositivos básicos como temporizadores o programadores a pilas hasta programadores que incorporan un amplio análisis de datos tras la captación de magnitudes físicas y que se reflejan en una interfaz de gestión con la que el regante puede ver los datos medidos en tiempo real y actuar en base a ellos. Mencionaremos brevemente algunos de ellos por su relevancia:

- Envira IOT diseña soluciones de monitorización y análisis de parámetros ambientales para explotaciones agrícolas que permiten optimizar los recursos, pero no dispone de un controlador para la automatización del riego. La infraestructura consiste en sensores inalámbricos conectados a la nube que aporta información al administrador a través de un SW SCADA para que él realice la toma de decisión y de esta forma optimizar la productividad.[23]

- AquArson instaló el primer sistema de telecontrol de riego en la CCRR del Valle Inferior del Guadalquivir. La instalación está formada por una red de

comunicaciones radio muy fiable con un consumo bajo. Consta de una red fija de dispositivos que se comunican a través de un Driver OPC con un SW de control donde se reciben todos los datos en tiempo real: caudales, volúmenes, presiones, así como se dispone de la planificación de las demandas o la previsión de los consumos. La CCRR realiza la gestión a través del SW y la App de control de riego, a la que es posible acceder desde un dispositivo móvil y desde donde es posible controlar mediante alarmas programadas posibles pérdidas o fugas.[24]

- Orionis Smart Water Networks es una empresa muy presente en grandes CCRR de España en las que instalan dispositivos remotos por comunicación radio para automatización de bombas y disponen de una amplia variedad de RTUs de riego o dataloggers, gama Sigma o Delta con conectividad 2G/3G mediante comunicaciones M2M. Estos programadores de telecontrol automatizan programaciones de riego además de captar datos extraídos de sensores conectados a dichos dispositivos y que se representan en una App móvil en tiempo real.[25]

- Blossom fue el primer sistema de riego inteligente diseñado por un equipo de desarrolladores californianos en 2014 que utiliza las predicciones meteorológicas al estar permanentemente conectado a Internet y tiene la capacidad de decidir la necesidad de irrigación.[26]

- Watersense de Hunter: A los programadores Hunter convencionales de corriente alterna es posible añadirles un sensor SolarSync, desarrollado por Hunter, que calcula la evapotranspiración (ET) diaria y ajusta automáticamente el tiempo de riego del programador, basándose en las condiciones climáticas locales. Además, incorpora los sensores Rain-clik y Freeze-clik que permiten cortar rápidamente el riego en situaciones de lluvia y baja temperatura.[27]

- Bynse es la primera solución Big Data para la agricultura. Esta solución está compuesta por las bysenbox o unidades sensoras autónomas que recogen hasta 14 parámetros: humedad, temperatura, conductividad del suelo, radiación solar, etc. y envían estos datos a través de telefonía móvil GSM/GPRS a bysencloud que es el servicio de información en la nube que representa la información. La plataforma procesa esta información y predice las necesidades hídricas. [28]

- Skydrop es un dispositivo de riego para el jardín que dispone de una app móvil tanto para Android como para IOS desde la que es posible configurar el programador. A partir de los datos introducidos de acuerdo con las características del suelo, tipo de planta, radiación solar, tipo de emisor, inclinación, etc., junto con los datos meteorológicos establece la mejor programación de riego. Así mismo, es importante destacar que calcula la cantidad de agua evaporada e infiltrada creando un calendario de suministro y posteriormente en base a las predicciones meteorológicas mediante un proceso de aprendizaje determinará la necesidad de irrigación.[29]

Por otro lado, cabe mencionar el proyecto IOT@AS donde colaboran cuatro empresas españolas (Egatel, Odin Solutions, Isotron y Tragsa) donde se diseña una plataforma abierta e inteligente para sistemas de telecontrol y gestión de las zonas de riego agrícola, que resuelve las problemáticas de interoperabilidad, eficiencia y conectividad de los sistemas existentes. El proyecto consiste en la instalación de remotas de distintos fabricantes que son gestionados mediante un SW basado en el estándar MEGA o NGSI/FIWARE que permite su interoperabilidad, así como la posibilidad de ser conectadas mediante nuevas redes móviles 5G. [30]

3. Diseño del sistema

Este proyecto consiste en el desarrollo global de un producto o sistema inteligente de riego, el cual aporta un valor añadido a la automatización de la irrigación, al dotar al sistema de un proceso de aprendizaje previo. En base a la analítica observada, será posible comprender los resultados de decisión de irrigación.

Para lograr una mayor comprensión del sistema y poder afrontar un proyecto de estas características, se ha dividido el proceso en varias partes diferenciadas que proporcionan una visión desde la fase inicial de definición de los requisitos hasta el análisis de los datos obtenidos mediante la utilización de sensores y el modelo de decisión en el cual está basado nuestro sistema inteligente de riego.

3.1. Definición de requisitos

Los sistemas de irrigación actuales deben cumplir la normativa vigente en relación con la energía y el medioambiente, así como la sensorización ambiental para construir un sistema de riego inteligente.

Actualmente de acuerdo con la Asociación Española de Normalización (UNE) se encuentra vigente la norma UNE 178405:2018 [31], donde se definen las características comunes que deben cumplir los sistemas de riego.

Así mismo, de acuerdo con dicha norma se definen los requisitos mínimos que debe tener un sistema de riego inteligente con respecto a:

- Características generales de un sistema de riego inteligente.
- Los elementos de monitorización de condiciones ambientales, hidráulicas, estado de la planta y/o estado del subsuelo de la zona verde en cuestión.
- Los elementos de actuación sobre el sistema de riego.
- La infraestructura de comunicación entre dispositivos de monitorización y control instalados en los parques con riego inteligente.
- Consideraciones sobre la realización del riego.
- Al software de monitorización y control que se le ofrece al usuario.
- A la interoperabilidad con otros sistemas.

En base a la norma definida se tendrán en cuenta sus requisitos para el diseño de nuestro sistema de cara a que el producto planteado en este proyecto pueda tener aplicación real.

Así mismo, es importante destacar que el diseño planteado en este proyecto se ha orientado pensando en las necesidades del pequeño agricultor, el cual no puede en muchas ocasiones realizar una gran inversión económica en tecnología, lo que hace inalcanzable la idea de implementar Sistemas de Riego Inteligente existentes en el mercado.

Por consiguiente, tal y como describiremos a continuación, se ha optado por una solución de bajo coste en la elección del hardware utilizado.

3.2. Arquitectura del sistema

La arquitectura del sistema planteada en nuestro diseño está basada en una red de sensores inalámbricos, los cuales son capaces de obtener de forma periódica, sin necesidad de intervención humana, distintos parámetros del entorno y enviar los datos capturados a través de módulos de radiofrecuencia.

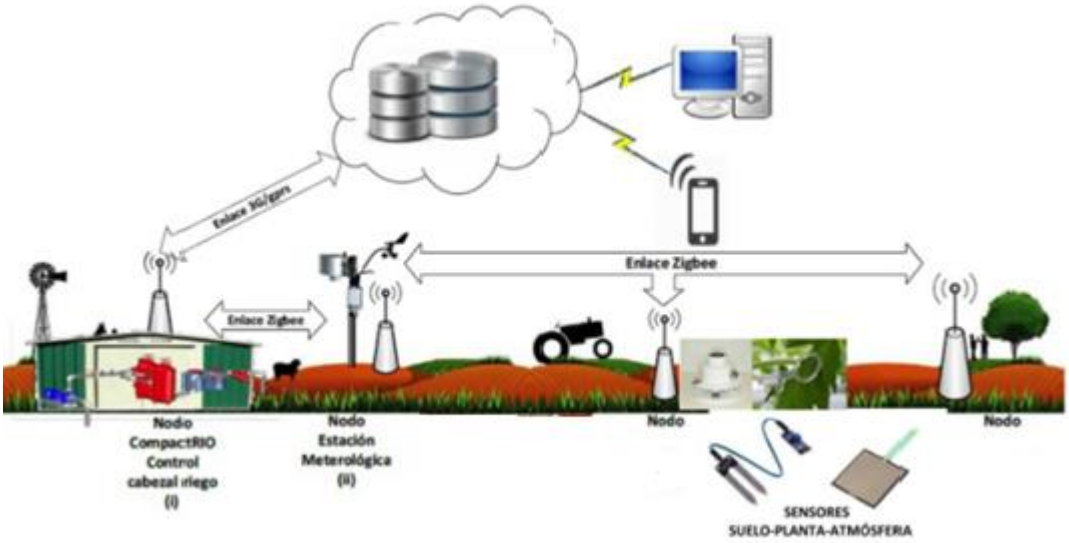

Figura 6: Infraestructura de la red adaptada a nuestro diseño [32]

En la imagen podemos ver una aproximación a alto nivel del diseño, cuya infraestructura está compuesta por:

- Nodos sensores (que incluyen sensores de temperatura, sensores de humedad, un procesador de Arduino cuya función es leer los parámetros medidos por los sensores, procesar los datos y transmitirlos mediante la conexión con un módulo de comunicación), los cuales se distribuyen a lo largo de la unidad de cultivo con el objetivo de disponer de muestras reales de las variables de decisión de toda la parcela que poder transmitir al nodo de control.

- Nodo estación meteorológica (que incluye una estación que registra valores atmosféricos mediante sensores de temperatura, barómetros para medir la presión atmosférica, higrómetro para registrar la humedad, pluviómetro con el que estimar la cantidad de agua en un periodo determinado, etc.; así como un procesador de Arduino para la lectura y procesado de los datos registrados) con el objetivo de reenviar los datos climáticos al nodo de control para su posterior análisis.

- Nodo de control cabezal de riego o nodo coordinador (que incluye un procesador Arduino con un módulo de comunicación Zigbee, encargado de recibir los parámetros medidos tanto por los nodos sensores como por el nodo de estación meteorológica, procesarlos y subirlos a Internet mediante un enlace 3G/GPRS para su almacenamiento en una BBDD y analizarlos posteriormente mediante algoritmos predictivos.

  Así mismo, dicho elemento será responsable de enviar la señal para activar/desactivar las electroválvulas de los distintos sectores de riego en función de la simulación obtenida mediante el entrenamiento del sistema con datos registrados previamente.

Cabe mencionar que, las comunicaciones entre los distintos nodos de la infraestructura diseñada y el nodo coordinador se realiza mediante el protocolo de comunicación Zigbee mientras que la comunicación entre el nodo de control y la nube por medio de cobertura GPRS.

Además, el prototipo planteado implementa una topología de red escalable, junto con una App móvil, con el objetivo de que el usuario pueda monitorizar y programar el sistema de una forma fácil y amigable.

Finalmente, el sistema mediante un modelo de Machine Learning basado en un proceso de entrenamiento de acuerdo con los datos registrados de un calendario de riego programado manualmente por el agricultor en la interfaz móvil, es capaz de simular cuándo o cuánto regar de acuerdo con el comportamiento humano predefinido en las mismas condiciones registradas.

En los apartados sucesivos se detalla en profundidad el diseño de la arquitectura, topología y características de la red, el hardware y protocolo de comunicación elegidos, así como su justificación frente a las distintas soluciones disponibles.

En relación con este primer análisis general detallado de la arquitectura a alto nivel, podemos definir o dividir nuestro sistema en subsistemas o módulos independientes que describiremos a continuación:

3.2.1. Módulo de sensores

Las redes de sensores inalámbricos o WSN están formadas por un grupo de dispositivos conectados con un propósito común y que permiten formar redes sin una infraestructura física predefinida ni administración central. [33]

Las redes de sensores están constituidas por sensores o nodos, que se ubicarán de forma que sea favorable su comunicación en función de la orografía o parcela a cubrir.

La estructura general estará diseñada para que cada nodo capte los parámetros necesarios a evaluar del entorno mediante un sensor analógico y se realiza una conversión a digital para posteriormente enviar dicha información a un nodo de control.

Así mismo, en base a las siguientes características y topologías descritas, prima la elección de la infraestructura establecida en nuestro diseño.

3.2.1.1. Características de red WSN [34]

- Tolerancia a fallos

  Es la capacidad de mantener la operabilidad de una WSN sin ninguna interrupción, a pesar de que uno o varios nodos no esté funcionando correctamente.

- Escalabilidad

  Es la característica que permite la conexión de gran cantidad de nodos en la red, así como la posibilidad de agregar más según la necesidad con el paso del tiempo.

- Costes de producción

  En función del propósito de la aplicación del sistema puede estar justificado o no este modelo en lugar de las soluciones tradicionales cableadas y, por tanto, es importante que el coste de cada nodo individual no sea muy elevado para que el presupuesto global no supere los modelos convencionales.

- Limitaciones de hardware

  Se debe tener en cuenta la necesidad de un tamaño reducido de los nodos, disponer de un consumo de potencia bajo, ser autónomos y funcionar sin supervisión, así como tener la capacidad de adaptación al medio.

- Consumo energético

  La conservación y administración energética en el entorno agrario tiene una especial importancia ya que los nodos inalámbricos por lo general dispondrán de una fuente limitada de energía y el tiempo de vida de los nodos sensores depende significativamente del tiempo de vida de la batería.

  Así mismo, es importante destacar la existencia de distintas topologías de redes de sensores inalámbricos o WSN que describiremos a continuación para una correcta elección del diseño utilizado en nuestro proyecto.

  3.2.1.2. Topología de la red

  La topología se define como la configuración de los componentes hardware en el espacio, así como el modo en el que se transmiten los datos por medio de esa estructura definida.

- Topología en estrella

  Esta configuración está formada por un nodo coordinador estratégicamente colocado en el medio de la parcela que se encarga de la sincronización de la red, así como de la recogida de datos enviados por los nodos sensores.

Figura 7: Topología en estrella [35]

La topología en estrella presenta la ventaja de un menor gasto energético y mayor rapidez de comunicación ya que es monosalto, pero está limitada por la distancia de transmisión entre los nodos.

- Topología en malla

  La topología en malla es un sistema multisalto en el que todos los nodos de la red son idénticos y actúan de router, de forma que pueden enviar y recibir información de cualquier vecino.

  Está configuración presenta mayor alcance y tiene alta tolerancia a fallos, dado que cada nodo dispone de varios caminos de comunicación y en caso de que cualquier enlace presente algún error, este podrá automáticamente enrutar hacia otro nodo.

  En contrapartida, en función de la distancia entre nodos y el número de estos, la red puede experimentar periodos de espera elevados.

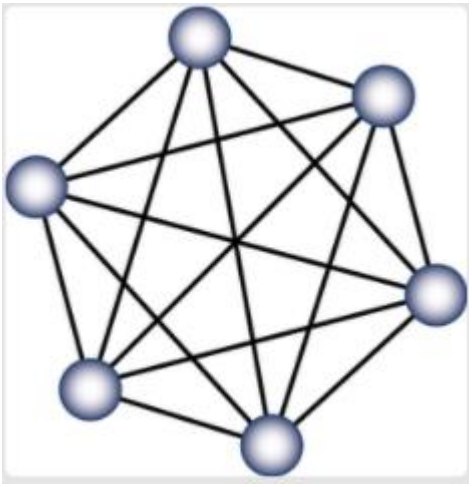

Figura 8: Topología en malla

- Topología híbrida

  Es una combinación de los dos modelos mencionados anteriormente, en la que redes malladas son conectadas en una red en estrella, combinando las ventajas de simplicidad y bajo consumo, así como la posibilidad de mayor rango de cobertura y tolerancia ante fallos.

  De acuerdo con las topologías descritas, cabe mencionar que en nuestro diseño se ha elegido la topología en estrella dado que es la más adecuada para monitorizar de forma distribuida pequeñas parcelas de explotación agrícola.

  A lo largo de este proyecto, se ha tenido en cuenta cubrir las necesidades del pequeño agricultor que no dispone de parcelas muy extensas que requieran de un gran rango de cobertura y alcance para el correcto funcionamiento del sistema, de forma que de acuerdo con los requisitos establecidos logremos un diseño de bajo coste y bajo consumo energético.

3.2.1.3. Elementos de la red

- Nodo sensor

Los nodos sensores son dispositivos electrónicos capaces de captar información del entorno, procesarla y transmitirla inalámbricamente. Su diseño debe orientarse hacia un hardware reducido, consumo muy bajo de energía y bajo coste, además de conseguir una transmisión de datos eficaz y amplio alcance.

El hardware básico de un nodo sensor está compuesto por sensor de temperatura, sensor de humedad, procesador Arduino UNO responsable de procesar los datos registrados por los sensores, módulo de reloj, módulo de alimentación, tarjeta de memoria, así como un módulo Xbee que permita la comunicación e intercambio de datos con el nodo de control.

En base a ello, se detalla a continuación cada uno de los elementos en los que se descompone el nodo sensor diseñado en este proyecto:

- Arduino UNO:

Arduino UNO es una placa de desarrollo que contiene un microprocesador ATmega328P. El microcontrolador ATmega328P dispone de 14 contactos de E/S digital (6 se pueden utilizar como salidas de PWM) y 6 entradas analógicas.

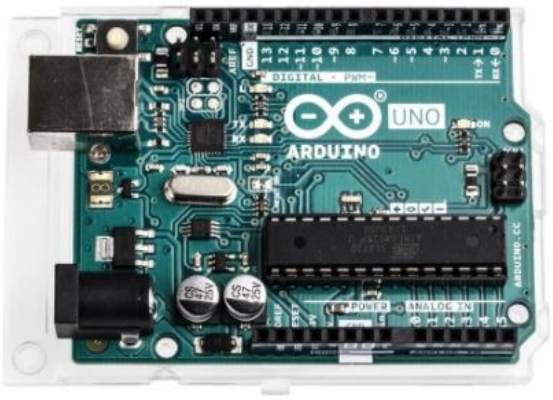

Figura 9: Arduino UNO [36]

Además, se dispone de una plataforma de código abierto u open-source para realizar la programación de una forma sencilla y cuya descarga es gratuita. Cabe mencionar que, no se detallará la programación de los componentes ya que no es objetivo de este proyecto más que la definición del sistema diseñado que nos proporcionará los datos para el posterior análisis mediante Modelos Predictivos.

Arduino UNO será el encargado de controlar el nodo sensor, leer los valores de temperatura y humedad medidos por los sensores, procesar los datos y transmitirlos mediante la conexión con un módulo de comunicación.

Así mismo, cabe mencionar que siguiendo con los requisitos definidos inicialmente se ha elegido Arduino UNO ya que además de cubrir las necesidades de procesar los datos obtenidos de los periféricos conectados,

dispone de bajo coste (alrededor de 20 euros en RS [36]), facilidad de programación y flexibilidad.

- Sensor de temperatura:

Se ha escogido el sensor de temperatura LM35 en los nodos sensores dado que es un sensor ampliamente conocido y de fácil uso que presenta buenas prestaciones a un bajo precio, ya que tiene un coste de unos pocos céntimos [37]

El sensor LM35 posee un rango de trabajo de entre -55 y 150 ºC, su salida es analógica y lineal con una pendiente de 10 mV/ºC, y dispone de una precisión de 0'5ºC.

Así mismo, dicho componente puede conectarse a una placa Arduino por medio de uno de sus pines analógicos de entradas de acuerdo con la siguiente figura.

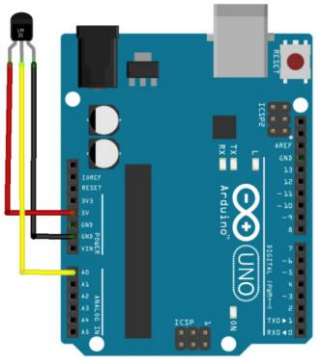

Figura 10: Conexión LM35 con Arduino UNO [38]

- Sensor de humedad:

Siguiendo con los criterios establecidos, se ha elegido para el diseño del sistema de riego el sensor FC-28 por su sencillez y bajo coste. [39]

El sensor mide la humedad del suelo por la variación de su conductividad. Si bien es verdad que no tiene una precisión absoluta de la humedad del suelo, con el fin de cubrir la necesidad de medir la humedad en un punto del terreno podemos considerar que es suficiente para el objeto de nuestro proyecto.

El FC-28 proporciona una medición analógica de dicha variable, así como una salida digital que es activada cuando la humedad supera un cierto umbral, lo cual puede ser interesante como consigna de desactivación de la irrigación.

Los valores registrados por el sensor pueden oscilar entre 0 (totalmente sumergido en agua) hasta 1023 donde podemos considerar suelo muy seco. Típicamente, un suelo ligeramente húmedo registra valores entre 600-700 mientras que un suelo seco entre 800-1023.

Por otro lado, la conexión con Arduino es muy sencilla, se alimenta el módulo conectando GND y 5V a las entradas correspondientes de Arduino y la salida analógica a una de las entradas analógicas de éste:

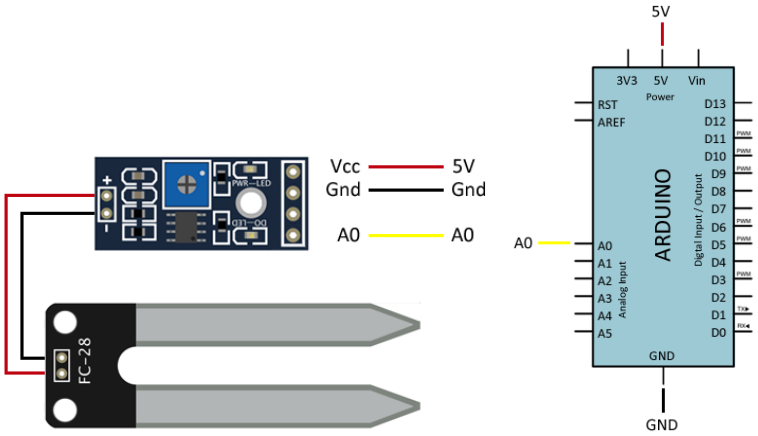

Figura 11: Conexión FC-28 con Arduino UNO [40]

- Módulo de reloj (RTC):

Es importante mencionar que Arduino dispone de la facilidad de implementar un reloj por software mediante programación con la librería "*Time.h"*, pero dispone de muchas limitaciones como retraso del reloj o incluso pérdida de la hora en el caso de que nuestro sistema se quede sin energía.

En base a esto, se determinó la implementación por hardware mediante un módulo RTC que garantiza mayor fiabilidad y exactitud de la hora registrada (dato muy importante para cotejarlo con la estación meteorológica próxima como se detallará más adelante).

A pesar de la amplia gama de RTCs que pueden ser implementados con Arduino, destacan dos componentes, el DS1307 y el DS3231 dentro de los circuitos integrados del mercado Tiny RTC y ZS-042, respectivamente.

En nuestro diseño, se ha optado por el ZS-042 ya que la gran diferencia entre ambos componentes es la precisión. ZS-042 tiene un oscilador interno al que no le afecta tanto los cambios de temperatura.

En contrapartida, el DS1307 presenta una mala respuesta en temperaturas extremas que afecta a su precisión, descartándose por este motivo dada la ubicación y en función del periodo estacional pueden someterse a heladas o temperaturas muy altas en medio del entorno agrario.

Así mismo, aunque el ZS-042 presenta un precio más elevado que el DS1307, sigue ofreciendo un bajo coste de acuerdo con los criterios que se han establecido a lo largo de este diseño [41], por lo que en relación precisión-precio se considera que está justificado ese pequeño incremento en el presupuesto con la garantía de una medición más precisa.

En relación con el conexionado, es muy sencillo gracias a la utilización del bus I2C de acuerdo con el representado en la siguiente figura:

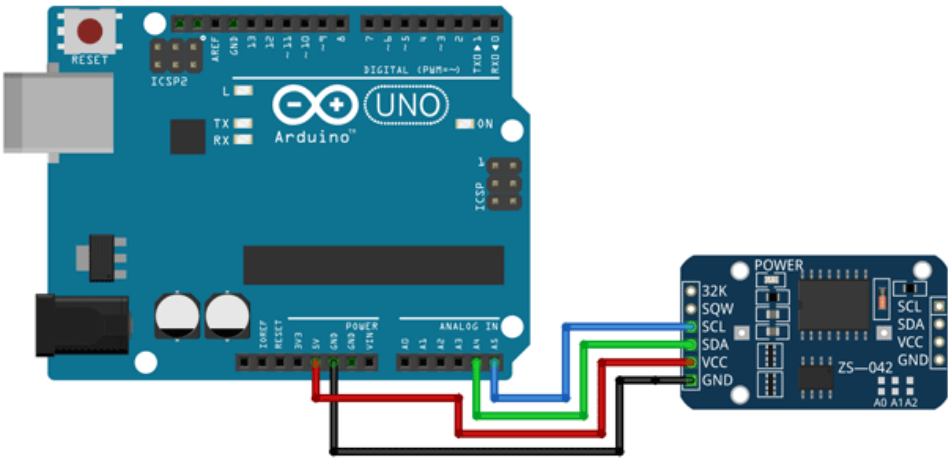

Figura 12: Conexión RTC ZS-042 con Arduino UNO [42]

- Tarjeta de memoria SD:

Se ha incluido en el prototipo un módulo de tarjeta Micro SD [43], dado que no suponía un incremento considerable del presupuesto y disponer de un respaldo de registro de datos es importante.

La topología en estrella si bien aporta grandes ventajas como hemos visto, dispone del inconveniente de pérdida de información en el caso de problemas en la conexión y de esta manera podremos almacenar dichas mediciones y enviarlas posteriormente.

Mediante la programación en Arduino usando la librería SD Arduino, que no detallaremos en este proyecto, es posible escribir y almacenar los datos registrados mediante el protocolo de comunicación SPI o Serial Peripherical Interface.

Así mismo, al igual que con los anteriores componentes es fácilmente conectable a Arduino de acuerdo con la siguiente figura [44]:

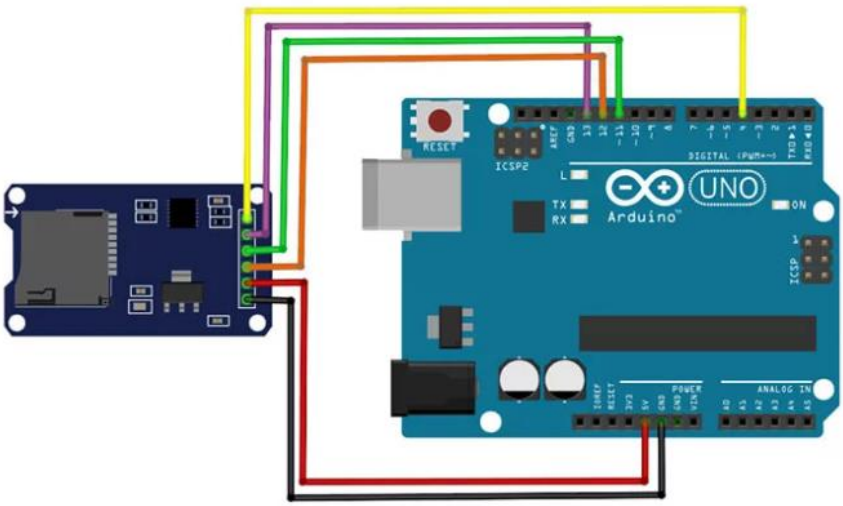

Figura 13: Conexión tarjeta Micro SD con Arduino UNO [44]

- Módulo XBee:

Dado que se debe dotar al nodo sensor de un elemento que permita la comunicación con el nodo coordinador, se ha seleccionado el módulo XBee ya que han sido diseñados para aplicaciones que requieren un alto tráfico de datos, baja latencia y una comunicación predecible.

XBee es el nombre comercial de la familia de módulos de radio basado en el protocolo Zigbee que en el apartado de comunicaciones indicaremos el motivo por el cual se ha optado por este protocolo.

Los módulos más sencillos de XBee son los Serie 1 y suficientes para nuestro diseño. En concreto, se ha seleccionado el Módulo Transceptor RF Digi International XB24CAWIT-001, frecuencia 2.4 GHz [45], que se puede adquirir por un precio inferior a 20 euros y proporciona una comunicación directa de largo alcance 1200m. Una característica destacable es que pueden permanecer en estado sleep, lo cual optimiza el rendimiento del sistema de alimentación.

En relación con el conexionado, para usar un módulo XBee con Arduino es necesario un Shield o adaptador para conectar el puerto serie de XBee con el de Arduino y los pines de XBee se representan en la siguiente figura [46]:

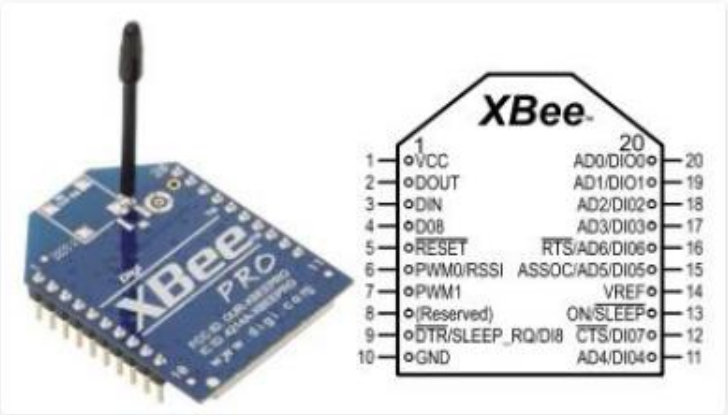

Figura 14: Módulo XBee conexionado [46]

- Módulo de alimentación:

Finalmente, tras valorar las distintas opciones de alimentación para el Arduino [47] como pueden ser mediante USB, con un adaptador de CA a CC, con la entrada de 5V o alimentación mediante baterías, se ha definido la utilización de baterías recargables LIPO (de iones de polímero de Litio), dada la posible ubicación del nodo sensor en un lugar alejado de la red eléctrica y la posibilidad de conectar un panel solar.

Por consiguiente, teniendo en cuenta la orientación del diseño hacia la aplicación real y aprovechando la infraestructura de riego de la que ya dispone el regante, si pensamos por ejemplo en un cabezal de riego y riego por sectores de la parcela, podemos establecer un nodo sensor en cada sector que mida los parámetros en ese punto.

- Nodo de estación meteorológica

El nodo de estación meteorológica es un dispositivo capaz de captar los datos atmosféricos registrados en la parcela y transmitirlos al nodo de control con el objetivo de disponer de variables adicionales de decisión en la ingesta de datos de nuestros algoritmos predictivos.

El HW básico de dicho nodo está formado por sensor de temperatura, sensor de humedad, barómetro, pluviómetro, con el objetivo de medir los datos climáticos de la ubicación de riego.

Análogamente con el nodo sensor, este elemento incluye un procesador Arduino UNO cuya función es la lectura y procesado de los datos registrados por los distintos componentes anteriormente mencionados, módulo de reloj, módulo de alimentación, tarjeta de memoria, así como un módulo de Xbee que permita la comunicación e intercambio de datos con el nodo de control.

A continuación, se describe únicamente los elementos que contribuyen al registro o medición de los parámetros climáticos dado que la conexión y comunicación con el nodo de control es análoga a la descrita para el nodo sensor.

- Sensor de temperatura/humedad:

En este caso se ha valorado la utilización del sensor DHT22 el cual puede medir ambas variables con unas prestaciones que se acercan bastante a la alta precisión. Se alimenta entre 3’3 y 6 V, dispone de señal de salida digital, rango de temperatura entre -40 y 125ºC con precisión de 0’5ºC y medición de humedad desde 0% RH hasta los 100% RH con precisión del 2-5% RH.

La conexión con Arduino es sencilla, ya que se alimenta al sensor a través de los pines GND y Vcc del mismo. Por otro lado, conectamos la salida Output a una entrada digital de Arduino. Necesitaremos poner una resistencia de 10K entre Vcc y el Pin Output. El esquema eléctrico se establece de acuerdo con la siguiente imagen:

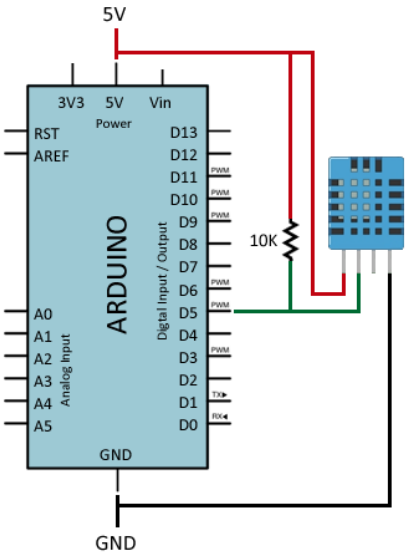

Figura 15: Conexión DHT22 con Arduino UNO [48]

- Barómetro:

Con el objetivo de disponer de datos de presión atmosférica en nuestro análisis, se selecciona para nuestro diseño el BMP180 cuyo precio ronda los pocos céntimos y proporciona alta precisión con baja potencia. El rango de medición es de 300 hPa a 1110 hPa. La comunicación se realiza a través de bus I2C y la tensión de alimentación es de bajo voltaje entre 1’8 y 3’6 V.

Frecuentemente se encuentran integrados en módulos como GY-68 que integran la electrónica necesaria para su conexión con Arduino de forma sencilla, alimentando el módulo desde Arduino mediante GND y 5 V y conectando el pin SDA y SCL de este con los pines correspondientes del barómetro.

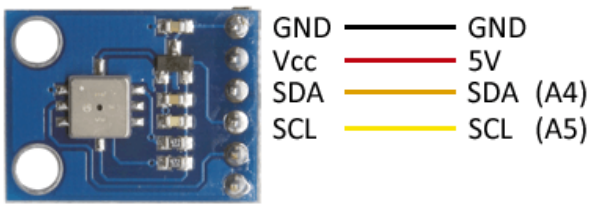

Figura 16: Conexión BMP180 con Arduino UNO [49]

- Pluviómetro:

La utilización del módulo YL-83 permite detectar la presencia de gotas de lluvia cuando se produce un cortocircuito entre las pistas conductoras de la placa, así como la cantidad de agua mediante la resistencia producida en el potenciómetro que es usado para la calibración y protege al resto del sistema.

Así mismo, el conexionado es sencillo. Se conecta el sensor a la placa de medición y se alimenta conectando los pines GND y 5 V a los pines correspondientes de Arduino.

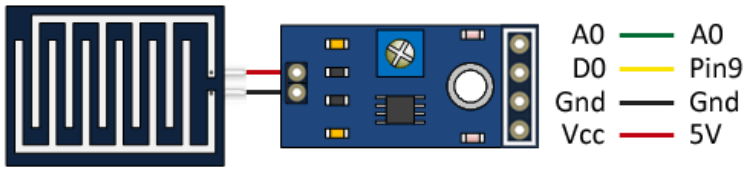

Figura 17: Conexión YL-83 con Arduino UNO [50]

- Nodo de control cabezal de riego

El nodo de control o coordinador análogamente incluye un procesador Arduino con un módulo de comunicación Zigbee en concreto se escoge el módulo XBee PRO, encargado de recibir los parámetros registrados tanto por los nodos sensores como por el nodo de estación meteorológica.

Así mismo, es importante mencionar que el nodo de control actúa también como cabezal de riego, cuya función es controlar la apertura o cierre de las válvulas solenoides o electroválvulas ubicadas en los distintos sectores de la parcela, así como registrar el caudal consumido al conectarse el Arduino a un sensor de flujo y electroválvulas existentes en la infraestructura del terreno.

Una vez procesada la información conjunta de todas las fuentes, se encarga de enviar una trama, a través de un modem GSM al servidor PC, para almacenar la información en Internet y su posterior análisis.

La tarjeta SIM900 GSM GPRS SHIELD elegida es compatible con todos los modelos de Arduino y la comunicación se realiza mediante comandos AT [51]

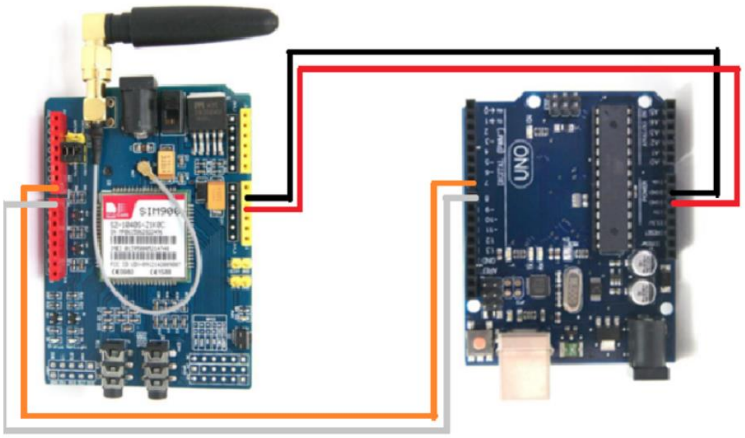

Figura 18: Conexión tarjeta GSM/GPRS con Arduino UNO [51]

3.2.2. Módulo de comunicación

En relación con las comunicaciones se debe distinguir entre las comunicaciones entre nodos y la comunicación del nodo de control con el servidor o servicio en la nube donde se almacenan, procesan y analizan los datos registrados por los sensores.

Con el objetivo de escoger el modelo más eficiente de comunicación entre los distintos elementos de nuestra infraestructura, se han analizado los siguientes tipos de comunicaciones descritas a continuación atendiendo a las siguientes características:

- Consumo de energía
- Distancia entre emisor y receptor
- Tasa de datos
- Naturaleza de los obstáculos
- Distorsión del ruido
- Regulaciones gubernamentales

En la siguiente figura podemos observar el gran abanico de tecnologías inalámbricas existentes atendiendo a su alcance, de las cuales describiremos las más relevantes que hemos valorado en el diseño de nuestro prototipo de sistema de riego inteligente.

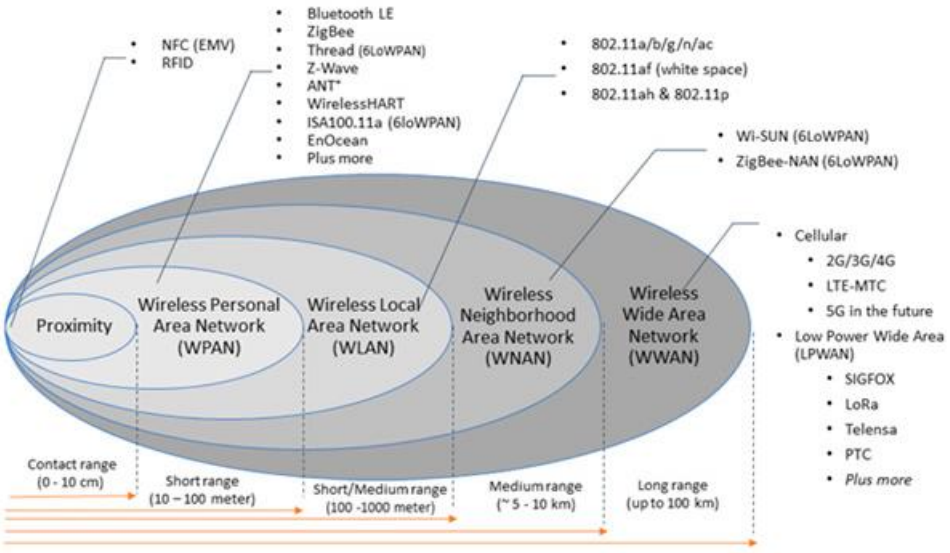

Figura 19: Representación de las tecnologías inalámbricas por alcance [52]

3.2.2.1. WIFI

La tecnología WIFI o red Wireless es una tecnología de transmisión de datos inalámbrica mediante ondas de radio basada en el estándar 802.11.

Las frecuencias más utilizadas por esta tecnología son 2’4 GHz o 5 GHz. La diferencia de frecuencia con respecto al resto de modelos de comunicación permite la transmisión de una tasa mayor de datos.

En contrapartida, cabe mencionar que la eficiencia de esta tecnología depende en gran medida de la cobertura y de las interferencias del entorno.

Teniendo en cuenta que en determinadas áreas rurales el acceso WIFI dispone de baja señal y adicionalmente, el coste en cuanto a hardware y energía es muy elevado, podemos considerar que no es un modelo de comunicación apropiado para nuestro sistema y en base a ello, se ha descartado en nuestro diseño.

3.2.2.2. WiMAX

WiMAX es un método de transmisión de datos a través de ondas de radio que utiliza frecuencias de 2’5 a 5’8 GHz basada en el estándar 802.16. Presenta la gran ventaja de ofrecer grandes alcances, conexión rápida y baja latencia sin disponer de infraestructura física o cableado, lo que lo hace especialmente interesante en el entorno rural.

En contrapartida, es necesario contacto directo con el repetidor, de manera que la antena instalada debe orientarse directamente a la conexión y evitando obstáculos que impidan la visión directa como por ejemplo árboles.

Por lo tanto, en base a lo descrito anteriormente, se ha descartado WiMAX en nuestro diseño ya que no podemos garantizar en una parcela en medio del campo una correcta conexión.

3.2.2.3. Sistemas de comunicación móvil

Las redes móviles han evolucionado en las distintas generaciones desde el GSM (2G) hasta el 5G, y son especialmente interesantes en aplicaciones IOT que requieran elevadas tasas de transmisión de datos o gran alcance.

Por ese motivo, hemos optado por conexión móvil para la transmisión de datos entre el nodo coordinador y el servidor de Internet.

3.2.2.4. Bluetooth

Bluetooth es una tecnología de corto alcance que permite la transmisión de datos y voz entre diferentes equipos mediante un enlace por radiofrecuencia y se caracteriza por su bajo consumo energético.

Adicionalmente, si bien es verdad que en las últimas versiones se han conseguido importantes mejoras en alcance, consumo (Bluetooth Low Energy o BLE) y en incrementar el número de nodos de la red (Bluetooth Mesh), dispone de poco alcance para la aplicación en nuestro diseño entre la comunicación de los nodos sensores con el nodo coordinador, además de no ofrecer una comunicación segura.

3.2.2.5. LPWAN o redes de área amplia de baja potencia

Las redes LPWAN se comunican a mayores distancias que otras redes de baja potencia, como en el caso de Bluetooth o NFC, por lo que pueden ser una buena alternativa para la conexión entre nodos ya que presentan un buen alcance y bajo consumo.

Las comunicaciones de baja potencia a largas distancias solo permiten transmitir pequeñas cantidades de datos por lo que las comunicaciones LPWA se emplean principalmente en el ámbito de los dispositivos IOT y las comunicaciones M2M.

M2M (machine to machine) hace referencia al intercambio de datos entre dos dispositivos conectados entre sí, a través de un enlace directo o a través de una red, para enviar y recibir datos.

Así mismo, dentro de esta categorización podemos distinguir varias redes de comunicaciones diferentes que presentarán ventajas e inconvenientes. Se ha considerado las más relevantes de acuerdo con las tecnologías implementadas en agricultura de precisión:

- Sigfox

Sigfox se caracteriza por una transmisión de radio de banda ultra estrecha que opera en una frecuencia sin licencia en las bandas de 868 MHz o 902 MHz, su cobertura puede alcanzar un rango de hasta 50 Kilómetros y una tasa de transferencia de datos baja.

En contrapartida, presenta el inconveniente de no ser un protocolo abierto y la única forma de utiliza Sigfox es a través de la compañía propietaria, contando adicionalmente con una limitación en cuanto al envío de datos diarios.

- LoRaWAN

LoRa es una red LPWAN muy similar a Sigfox, pero se diferencia en la transmisión de los mensajes se realiza a través de una banda muy ancha de frecuencias.

Adicionalmente, el protocolo LoRaWAN es un estándar abierto, pero solamente hay una compañía que fabrique chips que empleen ese protocolo.

Así mismo, cabe mencionar que no proporciona una infraestructura de red, por lo que el coste de inversión es importante para tener en cuenta en la elección de esta tecnología.

- Zigbee

El estándar IEEE 802.15.4 es un protocolo de comunicaciones para redes inalámbricas de corto alcance, bajo coste y consumo reducido. Este estándar se ajusta a redes de sensores, donde pueden existir múltiples nodos que requieran de un pequeño ancho de banda, pero una gran autonomía energética.

Por lo tanto, en base a lo descrito, el protocolo Zigbee ha sido el elegido para la comunicación entre los nodos sensores de nuestro diseño, ya que de acuerdo con los requisitos fijados proporciona optimización en el ahorro energético y cumple con la distancia de alcance de una parcela de un pequeño agricultor.

Otro aspecto importante, es la configuración de sus topologías de trabajo, ya que ésta puede ser adaptada en función de la necesidad de distribución de los nodos del sistema y permite la interoperabilidad entre diferentes dispositivos de distintos fabricantes.

A continuación, a modo de resumen se presenta un comparativa de las distintas tecnologías evaluadas y su caracterización de acuerdo con los requisitos comparados de alcance, consumo, tasa de datos, etc.

TECNOLOGÍA	CONSUMO	ALCANCE	MADUREZ	DISPONIBILIDAD	SEGURIDAD	USABILIDAD	TASA DE DATOS
GSM/GPRS	Muy alto	Alto	Muy Alto	Muy alto	Alta	Alta	Alta
SigFox	Bajo	Medio	Alto	Medio	Media	Alta	Muy baja
LoRa	Bajo	Medio	Bajo	Muy bajo (ad hoc)	N A	Baja	Muy baja
NB IoT							
WiFi	Alto	Bajo	Muy alto	Alto	Baja	Alta	Muy alta
BLE	Muy bajo	Muy bajo	Alto	Bajo	Baja	Media	Baja
ZigBee	Medio	Bajo	Medio	Muy bajo	Alta	Baja	Baja

Figura 20: Comparativa comunicaciones [53]

3.2.3. Módulo de almacenamiento

Brevemente, hay que indicar que los datos transmitidos por el nodo coordinador son transmitidos como se indicó en la infraestructura del sistema mediante una tarjeta SIM900 y almacenados en un servidor en la nube que dispondrá de una base de datos donde registrar los datos obtenido en cada punto de la parcela en cuanto a riego, caudal, temperatura, humedad y datos climatológicos proporcionados por la estación meteorológica.

3.2.3. Módulo de procesamiento

El subsistema de procesamiento hace relación al servicio que define tanto el proceso de ingesta de datos, análisis de cada uno de los parámetros registrados y transmitidos por los sensores, hasta definir y programar los distintos métodos predictivos que, mediante un proceso de entrenamiento supervisado previo, sea capaz de analizar o modelar el comportamiento registrado anteriormente con respecto a la decisión o cantidad de irrigación.

Para tal fin, se ha elegido la utilización de Python frente a otros lenguajes como R, Matlab y Julia también utilizados en Machine Learning. Es importante destacar que la capacidad de combinar librerías como NumPy y ScyPi, permite que Python sea uno de los lenguajes con mejor rendimiento para realizar proyectos de Machine Learning.

Así mismo, la librería Scikit-learn, es una librería de código abierto que unifica los principales algoritmos y funciones que facilitan todas las etapas de preprocesado, entrenamiento, optimización y validación de los modelos predictivos.

A continuación, se detallará en el siguiente capítulo el análisis completo de los datos desde el proceso de ingesta, limpieza y análisis de estos, hasta la programación de los distintos algoritmos predictivos que, tras un proceso de entrenamiento, proporcionará un análisis y comprensión del comportamiento de nuestro modelo de decisión.

4. Análisis Métodos Predictivos

El análisis predictivo es el proceso de utilizar análisis de datos para realizar predicciones basadas en estos. En este proceso se implementa técnicas analíticas, estadísticas y de aprendizaje automático con el objetivo de crear un modelo predictivo y modelar eventos futuros.

De acuerdo con la metodología CRISP-DM, tal y como se refleja en los apartados siguientes se puede observar que se ha orientado el proceso analítico en base a las 6 fases de esta: Comprensión del negocio, Comprensión de los datos, Preparación de los datos, Fase de Modelado, Evaluación e Implantación.

Por tanto, una vez comprendido y estudiado los procesos, elementos y variables que intervienen en la agricultura y el proceso de irrigación, mediante los datos históricos registrados por los equipos de medición combinados con algoritmos predictivos, es posible realizar un análisis de la decisión de irrigación, cantidad y periodo de esta.

Generalmente, el conjunto o conjuntos de datos utilizados para el análisis de métodos predictivos no es un conjunto estructurado y coherente, por lo que es necesario eliminar anomalías y transformarlo para que aporte información.

Si bien es verdad que el objetivo final es la aplicación de métodos predictivos, es indispensable realizar un preprocesado y limpieza de los datos previos que garanticen el correcto y óptimo funcionamiento de los algoritmos empleados, lo que supone aproximadamente un 60% del tiempo.

Por lo tanto, podemos diferenciar las siguientes fases o etapas en la aplicación y análisis de métodos predictivos que se han tenido en cuenta en el desarrollo del código empleado en nuestro proyecto y registrado en el Anexo.

4.1. Obtención e importación de datos

Idealmente, se montaría el sistema basado en la arquitectura descrita en el apartado anterior, y se estaría recopilando la información para posteriormente analizarla, pero por motivos de tiempo, logísticos… no se puede obtener un dataset mínimo implantando la arquitectura descrita, por lo que para poder simular el segundo objetivo principal del TFM, es necesario partir de datos existentes, intentar realizar paralelismos con nuestro sistema.

En base a ello, una gran parte de dedicación de este proyecto consistió en la búsqueda de un dataset válido para la realización del análisis de los distintos modelos predictivos.

Tras una exhaustiva búsqueda en distintas plataformas de datos Open Source, no fue posible la localización de un dataset válido y con suficiente número de muestras para poder entrenar al sistema.

Al tratarse de un proyecto de Smart Cities y con el objetivo de trabajar con datos reales, se optó por contactar con alguna Comunidad de Regantes, por lo que en el desarrollo de este proyecto utilizaremos mediciones realizadas por un equipo de telecontrol de la Comunidad de Regantes de Najerilla situada en Zarratón (Haro).

Gracias a la colaboración de dicha Entidad, disponemos del fichero proporcionado, junto con esta memoria, *“Sigma5_T63.xlsx”*, donde se incluyen registros dentro del intervalo temporal 01/08/2020-31/10/2020 medidos en campo por el equipo, tales

como: nivel de batería, cobertura, riego activo, caudal, etc. que describiremos a continuación.

Es importante indicar que, en función del parámetro registrado la frecuencia de muestreo es variable debido a que algunas variables se registran por tiempo, por comunicación del equipo o por cambio de estado cuando se produce el evento, por ejemplo, la apertura de la válvula general.

Debido a ello, los datos no son uniformes dentro del rango temporal por lo que, tal y como se puede comprobar, en la etapa de preprocesado implica la existencia de datos faltantes.

Adicionalmente, se han obtenido datos meteorológicos de la ubicación donde se encuentra situado realmente el equipo de telecontrol de riego mediante la página oficial del Gobierno de La Rioja [54] en las mismas fechas registradas por el equipo con un intervalo temporal entre muestras de 15 minutos:

Exportar

Haro - Datos cada 15 minutos - Fecha-Hora desde: 01-08-2020 08:15:00 - Fecha-Hora hasta: 30-10-2020 23:59:59

Fecha - Hora	07-D.Viento (Grados)	04-Hum. Rel (%)	08-Batería (V)	03-Lluvia (mm)	05-Presión (mb)	02-Rad. Sol (W/m2)		01-Temp. Ai (Celsius)			11-Temp.int (Celsius)				0
	Med	Med	Med	Acu	Med	Max	Med	Max	Med	Min	Max	Med	Min	Max (m/s)	Max (km
01-08-2020 08:15:00	4 (NNE)	79	13,7	0,0	0	47	39	19,3	19,2	19,2	20,5	20,5	20,4	7,4	26,6
01-08-2020 08:30:00	356 (NNW)	79	13,7	0,0	0	47	39	19,3	19,2	19,0	20,5	20,4	20,4	6,3	22,7
01-08-2020 08:45:00	346 (NNW)	80	13,7	0,0	0	47	40	19,2	19,0	18,9	20,5	20,4	20,4	5,5	19,8
01-08-2020 09:00:00	349 (NNW)	80	13,7	0,0	0	46	39	19,1	19,1	19,0	20,5	20,5	20,4	5,5	19,8
01-08-2020 09:15:00	350 (NNW)	79	13,7	0,0	0	46	39	19,2	19,1	19,0	20,6	20,5	20,5	5,3	19,1
01-08-2020 09:30:00	345 (NNW)	79	13,7	0,0	0	47	39	19,0	18,9	18,8	20,7	20,7	20,6	6,5	23,4
01-08-2020 09:45:00	351 (NNW)	77	13,7	0,0	0	47	39	19,1	19,0	18,9	20,9	20,8	20,7	5,5	19,8
01-08-2020 10:00:00	357 (NNW)	76	13,7	0,0	0	47	39	19,0	18,8	18,7	21,0	21,0	20,9	6,2	22,3
01-08-2020 10:15:00	1 (NNE)	76	13,7	0,0	0	47	39	18,9	18,8	18,7	21,0	21,0	21,0	5,5	19,8
01-08-2020 10:30:00	6 (NNE)	75	13,7	0,0	0	47	40	19,1	18,9	18,7	21,1	21,0	21,0	6,4	23,0
01-08-2020 10:45:00	6 (NNE)	74	13,7	0,0	0	48	40	19,3	19,1	18,9	21,2	21,1	21,0	6,7	24,1

Figura 21: Extracción datos meteorológicos

Desde la ubicación web mencionada es posible realizar un exportado fácilmente a Excel seleccionando la estación, en nuestro caso Haro y las variables que queremos comparar como por ejemplo Temperatura, Humedad, Precipitación, etc.

Dicho exportado se adjunta con la memoria de este proyecto en el fichero *"Datos_metereológicos_Haro.xlsx"*, con el cual poder comprobar el código desarrollado.

En este caso, disponemos de dos fuentes de datos que importaremos mediante Python a nuestro entorno de trabajo y posteriormente uniremos en un único dataset del que poder extraer información tras realizar el preprocesado de los mismos, tal y como se describe a continuación.

4.2. Preprocesamiento de datos

El preprocesamiento de datos es una etapa esencial, encargada de la limpieza de datos, su integración, transformación y reducción para la siguiente fase que garantice el óptimo rendimiento de los algoritmos de predicción.

Tal y como se observa en el código adjunto en los Anexos apartado *9.1.Código Preprocesado de datos,* en esta etapa se distinguen distintas fases que permiten la obtención de un dataframe único de datos.

En la etapa de preprocesado se detectan y corrigen valores duplicados, valores faltantes, outliers (valores anómalos fuera de rango), etc. así como se realiza la integración de ambas fuentes de datos por lo que podemos distinguir las siguientes fases:

- Exploración de datos

Una vez realizado la importación de ambos ficheros a nuestro entorno utilizado, Spyder, es importante entender los datos de los que disponemos con el objetivo de disponer de una visión de los procedimientos a desarrollar para el preprocesamiento de estos.

En base a ello, verificar los tipos de datos de los dataframes al inicio del análisis es importante para evitar posibles errores posteriormente, ya que por ejemplo Panda puede asignar automáticamente tipos de datos como pueden ser objetos y las funciones matemáticas para Machine Learning solamente aceptan datos numéricos.

En este caso, revisando el fichero del equipo de riego se observa que el formato de los datos es incorrecto y se debe realizar un cambio en el mismo para que sea más sencilla su manipulación.

De acuerdo con lo observado en Spyder, el fichero dispone de datos tipo object (Nombres variables y Unidades), datatime (Fecha de registro) y float (Valor medido).

```
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Variable gráfica  9212 non-null   object
 1   Fecha             9212 non-null   datetime64[ns]
 2   Valor             9212 non-null   float64
 3   Unidades          8738 non-null   object
dtypes: datetime64[ns](1), float64(1), object(2)
```

En este caso, las variables se encuentran mezcladas y es necesario realizar una transformación de los datos donde para cada fecha se disponga de los valores de las distintas variables correspondientes.

	A	B	C	D
1	Variable gráfica	Fecha	Valor	Unidades
2	T-63 - Nivel señal GPRS	01/08/2020 8:30	56	%
3	T-63 - Nivel de batería	01/08/2020 8:30	6,6	V
4	T-63 - temperatura	01/08/2020 8:30	20,9	ºC
5	VG1 - Estado RedePlan	01/08/2020 8:30	1	
6	VG1 - Riego activo	01/08/2020 8:30	0	
7	VG1 - ID de riego en memoria	01/08/2020 8:30	0	
8	VG1 - Caudal	01/08/2020 8:30	0	l/s
9	VG1 - Paso de caudal	01/08/2020 8:30	0	
10	VG1 - Alarma de caudal	01/08/2020 8:30	0	
11	VG1 - Velocidad (Q/S)	01/08/2020 8:30	0	m/s
12	VG1	01/08/2020 8:30	0	
13	T1	01/08/2020 8:30	0	
14	T2	01/08/2020 8:30	0	
15	T3	01/08/2020 8:30	0	
16	T4	01/08/2020 8:30	0	
17	T-63 - temperatura	01/08/2020 8:45	21,7	ºC
18	T-63 - Nivel de batería	01/08/2020 9:15	6,6	V
19	T-63 - temperatura	01/08/2020 9:30	22,3	ºC
20	T-63 - Nivel señal GPRS	01/08/2020 9:35	59	%
21	T-63 - Nivel señal GPRS	01/08/2020 10:05	56	%

Figura 22: Fichero datos dispositivo telecontrol riego

En contrapartida, el fichero de datos meteorológicos está formado por valores numéricos correctamente, tal y como se puede observar en los datos obtenidos del dataframe importado.

```
RangeIndex: 8797 entries, 0 to 8796
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Fecha-Hora              8797 non-null   datetime64[ns]
 1   04-Hum. Rel (%) Med     8797 non-null   int64
 2   03-Lluvia (mm) Acu      8797 non-null   float64
 3   05-Presión (mb) Med     8797 non-null   int64
 4   02-Rad. Sol (W/m2) Max  8797 non-null   int64
 5   02-Rad. Sol (W/m2) Med  8797 non-null   int64
 6   01-Temp. Ai (Celsius) Max  8797 non-null   float64
 7   01-Temp. Ai (Celsius) Med  8797 non-null   float64
 8   01-Temp. Ai (Celsius) Min  8797 non-null   float64
 9   11-Temp.int (Celsius) Max  8797 non-null   float64
 10  11-Temp.int (Celsius) Med  8797 non-null   float64
 11  11-Temp.int (Celsius) Min  8797 non-null   float64
 12  06-V.Viento Max (m/s)   8797 non-null   float64
 13  06-V.Viento Max (km/h)  8797 non-null   float64
 14  06-V.Viento Med (m/s)   8797 non-null   float64
 15  06-V.Viento Med (km/h)  8797 non-null   float64
dtypes: datetime64[ns](1), float64(11), int64(4)
```

- Linealización variables categóricas

Dado que las variables proporcionadas por el dispositivo de riego son objetos, debemos realizar una transformación de los datos de manera que podamos disponer para cada instante de tiempo de los valores numéricos de todas las variables.

Para ello se traspone dichos datos de manera que para cada bloque de instantes temporales se corresponda con una columna nombrada con el valor de la variable, todos los valores numéricos reflejados en la columna común valor.

Así mismo, en este punto aprovechamos para descartar los parámetros T1,T2, T3, T4, VG1 alarma de caudal, VG1 Estado RedePlan y VG1 Id de riego en memoria, ya que no aportan información adicional a nuestro modelado.

La activación/desactivación de cada toma individual viene determinada por la activación/desactivación de la válvula general, por lo que éstas son variables redundantes.

Así como, la alarma de válvula de la general en el caso de valores altos/bajos de caudal, el estado de RedePlan (Programa del dispositivo donde realizar agrupaciones de horarios de riego) y el identificador de riego programado carecen de valor de información de cara a los entrenamientos futuros, por lo que ya no se ha realizado el proceso de limpieza de dichos parámetros.

- Eliminación de muestras duplicadas

Una vez que ya disponemos de los datos en dataframes independientes por variables, realizamos la limpieza de estos, donde como primer paso se realiza la eliminación de valores duplicados en los mismos instantes temporales.

Para ello, de acuerdo con el código adjunto en el anexo de esta memoria, se crean las funciones:

- *remove_duplicates:* Dicha función se ejecuta con el objetivo de corregir valores duplicados de temperatura, batería y cobertura, donde se observa que únicamente en con esta última variable se realiza corrección de los datos.

  En el caso que exista valor de cobertura en el mismo instante temporal, elimina el índice del dataframe cuyo valor diste más de la media de dicho parámetro con el objetivo de minimizar el error cometido. Si ambos registros son iguales, se elimina la segunda muestra de cobertura.

- *remove_null:* Dicha función se ejecuta con el objetivo de corregir valores nulos de temperatura, batería y cobertura, donde se observa que únicamente en con esta última variable se realiza corrección de los datos.

  Dado que un registro de 0% de cobertura implicaría que el equipo no comunicara, se consideran muestras erróneas registradas por el dispositivo ya que pueden distorsionar la precisión del entrenamiento de los algoritmos predictivos y, por consiguiente, se elimina cualquier muestra nula del dataframe.

- *remove_duplicates_event:* Dicha función se ejecuta con el objetivo de corregir muestras duplicadas de los bits de válvula general, paso de caudal y riego activo. Se observa que en estos casos cuando se produce un cambio de evento, el dispositivo registra en el mismo instante temporal el paso del 0 al 1 lógico en la activación o la transición del 1 al 0 lógico en la desactivación.

  Por ese motivo, es necesario aplicar una corrección en los datos mediante dicha función, en la que se coteja con la muestra del instante anterior y el algoritmo se queda con la muestra del cambio de estado como valor correcto en ese instante.

- *remove_duplicates_flow:* Esta función contempla la eliminación de muestras duplicadas en el caso del caudal o la velocidad de caudal.

  En ambos casos, se valora inicialmente el estado del bit Paso de caudal ya que en la etapa de exploración de los datos se observó que el valor de caudal y velocidad registra valor nulo en el mismo instante temporal que registra un dato positivo cuando se produce el cierre de la electroválvula, es decir, en el momento del cambio de evento explicado en la anterior función.

  En base a ello, se determina como valor correcto el valor de caudal o velocidad de caudal acorde con el paso de caudal. Si no existe paso de caudal se elige la muestra nula y si existe, la muestra positiva.

  Así mismo, en el caso de existencia de paso de caudal y dos muestras positivas de la variable analizada, se descarta la muestra que más dista de la media de dicho parámetro.

Una vez que se dispone de datos correctos para las variables registradas por nuestro sistema, se realiza la concatenación de todos los parámetros registrados por el equipo de riego con los datos meteorológicos extraídos de la ubicación real del dispositivo.

Es fácil observar que, al no disponer de datos para todas las variables en los mismos instantes temporales, el dataframe obtenido presenta valores faltantes representados por *“nan”* que serán tratados en la siguiente fase del preprocesado.

- Corrección de valores faltantes:

En base a la problemática descrita anteriormente, es importante mencionar que eliminar filas o columnas de los datos faltantes no es óptimo ya que se pierden un buen número de datos útiles para nuestro análisis.

Por consiguiente, de acuerdo con el código adjunto en el anexo de esta memoria, se crean las funciones:

- *corregir_datos_faltantes_riego:*

  En el caso de las consignas de riego activo, apertura de la válvula general y paso de caudal, se tiene en cuenta si se dispone de valor de caudal, se toma la decisión de considerar los bits activos y, por tanto, sustituir los valores *“nan”* por un uno lógico. Y, en caso contrario que las consignas están desactivadas y se reemplazan por un cero lógico.

- *rellenar_datos_faltantes:*

  Esta función se aplica al resto de parámetros (cobertura, batería, temperatura del equipo, datos atmosféricos como pluviometría,…). Se toma la decisión de reemplazar los valores faltantes por la medida anterior registrada, ya que se supone que en un intervalo inferior a 15 min las muestras no varían considerablemente en dichos parámetros.

Finalmente, el dataset con el que se alimenta nuestros algoritmos de entrenamiento de los métodos predictivos se ha almacenado en el fichero *“dataset.xlsx”* adjuntado con este proyecto.

4.3. Análisis de variables

Una vez obtenido el conjunto de datos, es importante verificar la existencia o dependencia entre variables ya que incluir variables correlacionadas en el entrenamiento de los modelos predictivos no aportarán información adicional y ralentizarán el funcionamiento de estos.

Inicialmente, cabe mencionar que de acuerdo con lo planteado en el apartado de preprocesado se puede intuir que las variables Riego_act, Apertura_VG y Paso_Caudal están relacionadas.

En base a ello, tras visualizar la siguiente gráfica donde se representa la correlación entre estas, podemos concluir que están totalmente correladas y por ese motivo se descartan Apertura_VG y Paso_Caudal.

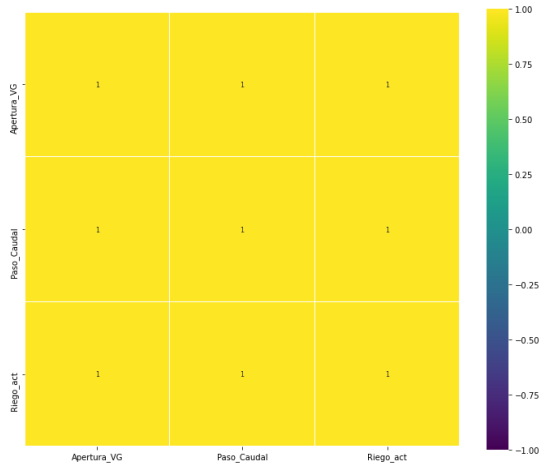

Figura 23: Matriz de correlación variables riego

Así mismo, mediante la librería "*matplotlib"* que ofrece Python, se ha graficado los distintos parámetros en función de Riego/No Riego, de acuerdo con el código adjunto en el apartado *9.2. Código Análisis de variables.*

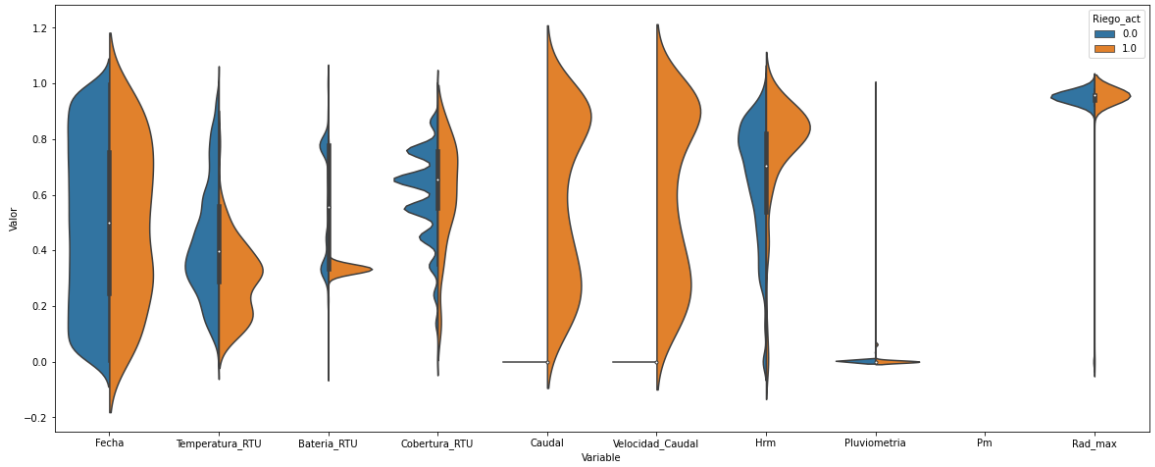

Figura 24: Representación variables 1 en relación con Riego_act

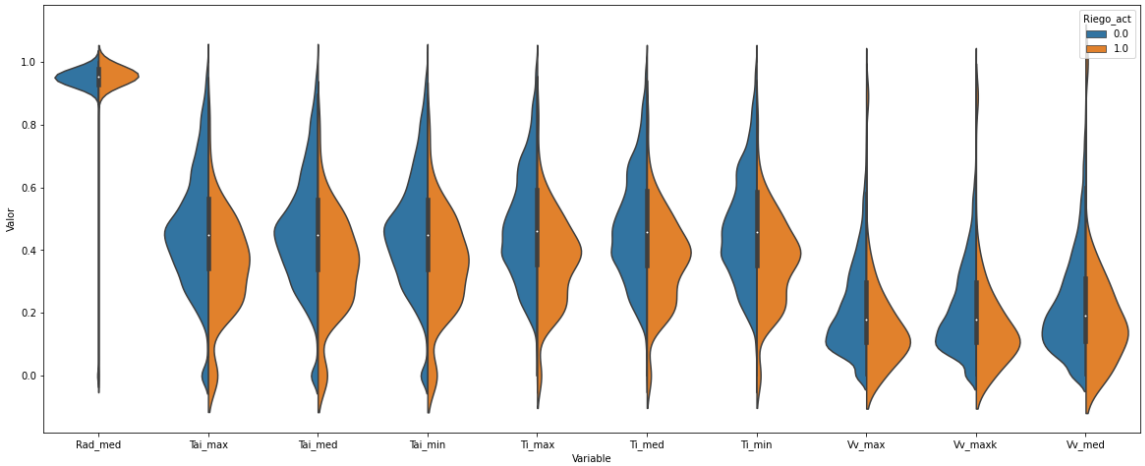

Figura 25: Representación variables 2 en relación con Riego_act

De acuerdo con las gráficas anteriores, se observa que existen varias variables que presentan una distribución uniforme entre regar o no regar, y por tanto, apenas aportan valor para discriminar en el modelo. Por ese motivo, descartamos las variables Pluviometría y presión atmosférica (Pm).

Adicionalmente, es fácil observar que tanto la velocidad de caudal con el caudal como las distintas medidas de temperatura entre sí y las velocidades del viento son variables dependientes y van a aportar la misma información a nuestros modelos.

Idealmente, el objetivo es incluir en el modelo todas las variables que aporten una componente explicativa alta a la variable objetivo y, por tanto, tengan una baja correlación entre ellas.

Es importante mencionar que, dos variables con una alta correlación entre sí, implica que una puede ser inferida a partir de la otra con un grado de error relativamente bajo, y por tanto aportarán información redundante al modelo.

Por este motivo, se analiza la matriz de correlación entre las variables, buscando aquellas con un nivel de correlación alto.

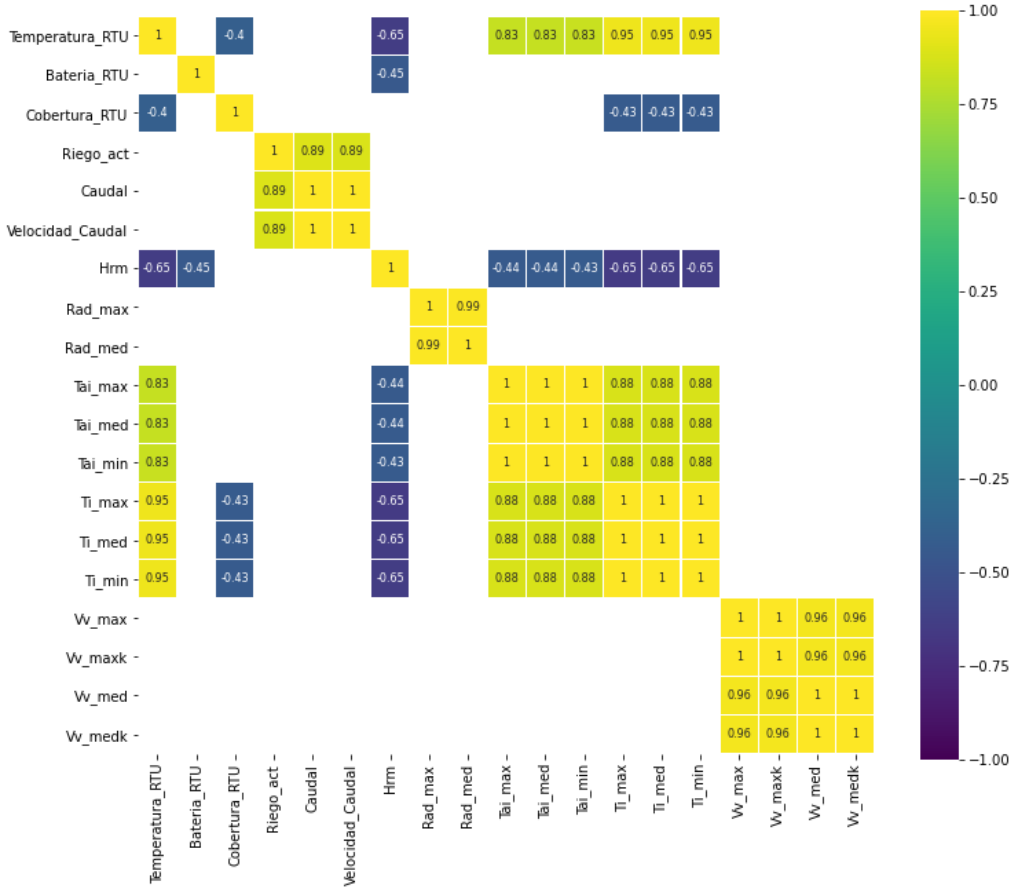

Figura 26: Matriz de correlación parámetros dataset

En la gráfica anterior se refleja la matriz de correlación entre los distintos parámetros registrados donde el índice entre 0 y 1 establece la dependencia total entre cada par de variables. En función de correlación positiva o negativa, dicha dependencia aumenta o disminuye en proporción constante.

En base a esto, se confirma lo observado en las gráficas anteriores, la correlación entre las temperaturas como era de esperar ya que las medidas por el equipo y las obtenidas mediante la estación meteorológica son de la misma ubicación, los registros del viento, etc. presentan valores altos de acuerdo con la interpretación de Pearson:

Coeficiente	Interpretación
0	Nula
>0.0 – 0.2	Muy baja
>0.2 – 0.4	Baja
>0.4 – 0.6	Moderada
>0.6 – 0.8	Alta
>0.8 – <1.0	Muy alta
1.0	Perfecta

Figura 27: Interpretación correlación

Posteriormente, se crea un nuevo dataframe donde se incluyen solamente las características que contribuyen a la discriminación de riego/no riego cuya matriz de correlación se refleja de acuerdo con la siguiente gráfica.

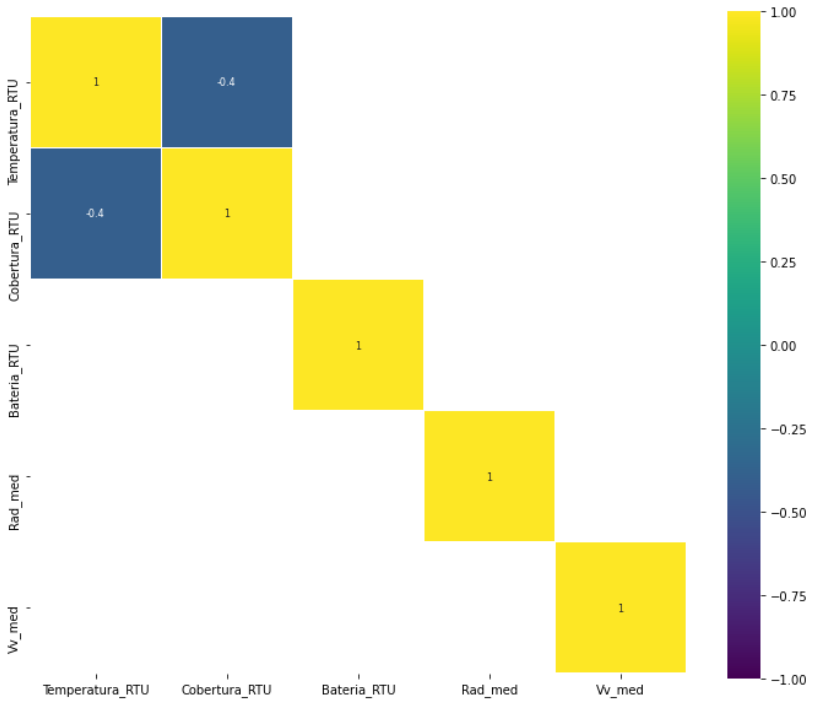

Figura 28: Matriz correlación características principales

Por tanto, las características principales que se tendrán en cuenta en nuestros modelos predictivos serán:

[['Fecha','Riego_act','Temperatura_RTU','Cobertura_RTU','Bateria_RTU','Rad_med','Vv_med']]

Por otro lado, cabe mencionar que adicionalmente de conocer la necesidad de regar o no, puede aportar gran información de cara a los agricultores el conocimiento de cuánto regar.

En base a ello, se han transformado los datos de partida en otro dataframe donde se registra la fecha y el acumulado diario, considerando éste como la suma de los

caudales registrados por el periodo o duración de dichos registros en cada intervalo diario.

Así mismo, detallar de acuerdo con el conocimiento obtenido en la exploración de los datos, que se observó que la cantidad regada en un instante/intervalo exacto del tiempo es muy imprecisa.

Esto implicaría la no convergencia de los algoritmos, por lo que tiene mucho más sentido valorar el análisis del acumulado diario en otro entrenamiento que se reflejará en el apartado de simulación.

4.5. Aplicación métodos predictivos

En este proyecto se aplicará un aprendizaje automático supervisado ya que se dispone de un gran número de muestras de entrada y sus respectivas respuestas en nuestro sistema donde la variable objetivo a predecir es Riego_act, es decir, la decisión de regar o no, así como Acumulado en el último entrenamiento.

En los distintos modelos se dividirá un 70% de las muestras como datos de entrenamiento y el 30% restante como muestras test.

El sistema será capaz de predecir de acuerdo con las muestras de entrenamiento una respuesta a los datos test y de esta manera, replicar con mayor o menor precisión en función del modelo utilizado el comportamiento del sistema de riego.

De acuerdo con el código adjunto en el apartado *9.3. Código Análisis métodos predictivos* de los Anexos, se implementa distintos algoritmos predictivos tanto de clasificación como de regresión con el objetivo de estimar cuál es el más apropiado u ofrece una mejor estimación o precisión.

Mediante la librería Scikit-learn de Python es relativamente sencillo declarar los distintos algoritmos ya que dicha biblioteca ya incluye distintos modelos de aprendizaje automático de modelos de clasificación o regresión.

En nuestro programa se han evaluado los siguientes algoritmos de regresión y de clasificación dado que son los más habituales en el caso de aprendizaje automático supervisado:

- Algoritmos de regresión:
    - Regresión lineal múltiple:
    - Modelo de vectores de soporte de regresión
    - Árbol de decisión de regresión
    - Bosques aleatorios de regresión
- Algoritmos de clasificación:
    - Modelo de regresión logística
    - K vecinos más cercanos
    - Máquina de vectores de soporte de clasificación
    - Naive Bayes
    - Árboles de decisión de clasificación
    - Bosques aleatorios de clasificación

A continuación, en el siguiente capítulo se observarán los resultados obtenidos de acuerdo con varios entrenamientos realizados.

5. Simulación y pruebas

En la fase de simulación, se ha analizado la ingesta de distintos parámetros de entrada en varios modelos predictivos con el objetivo de validar y estimar la exactitud de la salida de acuerdo con los entrenamientos que describiremos posteriormente.

Inicialmente, es importante mencionar que el proceso de validación está basado en una comparación cuantitativa mediante el indicador de precisión, definido como el porcentaje de riego clasificado correctamente; así como la evaluación de la "*accuracy*" o exactitud, que representa el porcentaje de predicciones correctas frente al total. Por tanto, es el cociente entre los casos bien clasificados por el modelo (verdaderos positivos y verdaderos negativos), con respecto al total de todas las muestras.

Adicionalmente, el análisis de la matriz de confusión en el caso de los modelos de clasificación aporta gran cantidad de información. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real.

Uno de los beneficios de la matriz de confusión es que facilita el análisis si el sistema está confundiendo dos clases. Si en los datos de entrada el número de muestras de clases diferentes varía mucho la tasa de error del clasificador, no es representativa de una buena estimación del clasificador.

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 29: Matriz de Confusión

Así mismo cabe mencionar que, para el entrenamiento de los algoritmos se destinó el 70% de las muestras y el 30% se usó para su validación.

A continuación, se mostrarán los resultados obtenidos de acuerdo con cuatro entrenamientos distintos realizados en nuestro sistema y que se reflejan en el código adjunto en el apartado de Anexos *9.4. Código Entrenamientos.*

5.1. Entrenamiento 1

El primer entrenamiento realizado consistió en evaluar la dependencia de las variables registradas del equipo y las condiciones meteorológicas en la decisión de irrigación.

Por ese motivo se separan en la variable dependiente "X" los parámetros: 'Temperatura_RTU','Cobertura_RTU','Bateria_RTU','Rad_med','Vv_med' y en la variable independiente "y" la variable que determina la existencia de riego o no, 'Riego_act'.

Los datos obtenidos tras esta primera simulación son los siguientes:

lr_multiple	svr	adr	bar	r_log	K_neig	svm	naive_bayes	adc	bac
0’02	-2’25	0’09	0’35	0’04	0’5	0’04	0’47	0’55	0’65

Figura 30: Comparativa modelos Entrenamiento 1

De acuerdo con los resultados de esta primera simulación donde se tienen en cuenta variables meteorológicas, se observa que los algoritmos de regresión no son capaces de estimar la decisión de irrigación con una predicción muy buena.

Por otro lado, si evaluamos los algoritmos de clasificación, el que proporciona una mayor precisión es bosques aleatorios.

El algoritmo de regresión logística, tal y como se observa en los resultados presenta una predicción muy baja de acierto en estimar riego, en cambio tiene gran asertividad al estimar no riego, lo que proporciona una exactitud del modelo de 0’66 a pesar de proporcionar una predicción inferior al 1%.

	Precisión	Recall	F1-score	Support
0	1	0’66	0’79	3410
1	0’74	0’92	0’07	51
Accuracy			0’99	3461

Figura 31: Report LogisticRegression E1

Así mismo, este mismo comportamiento se aprecia en el modelo supervisado máquina de vectores de soporte, cuya predicción ronda el mismo orden descrito en el modelo de regresión.

Cabe mencionar que, el bajo valor de muestras de riego=1 supone un peor comportamiento en la estimación de la precisión de estos algoritmos con respecto a otros que analizaremos, por lo que para evitar una precisión nula, se ha balanceado el dataset asignándole más peso a las muestras con predicción de riego=1.

En el caso del modelo de K vecinos más cercanos, devuelve un resultado de precisión de un 50 % con una exactitud del 99%, es decir, acierta riego igual que falla por lo que no se ajusta correctamente a nuestro dataset.

	Precisión	Recall	F1-score	Support
0	0’99	0’99	0’99	3410
1	0’50	0’35	0’41	51
Accuracy			0’99	3461

Figura 32: Report KNeighborsClassifier E1

En el caso de Naive Bayes al igual que en los modelos de regresión logística y máquina de vectores de soporte se obtiene una predicción muy baja, dispone de una buena exactitud de no riego pero no es capaz de estimar la necesidad de irrigación. Este algoritmo está penalizado por la baja cantidad de muestras de no riego con respecto a riego.

En el caso del modelo de árboles de decisión la predicción de riego se incrementa a un 55%, si bien es verdad que no aporta más que un comportamiento aleatorio, ya que predice riego correctamente la mitad de las veces y por tanto, no podemos considerar una buena predicción del comportamiento.

Finalmente, el algoritmo de bosques aleatorios de clasificación es el que ofrece una mayor precisión de acuerdo con los datos obtenidos.

	Precisión	Recall	F1-score	Support
0	0’99	0’99	0’99	3410
1	0’65	0’65	0’65	51
Accuracy			0’99	3461

Figura 33: Report RandomForestClassifier E1

Este algoritmo es uno de los métodos más eficientes de predicción ya que construye diferentes conjuntos de entrenamiento y test sobre los mismos datos, de modo que se generan distintos árboles de decisión cuya unión forma un bosque aleatorio, creando un modelo más robusto.

Si bien es verdad que dicho modelo ofrece la mejor estimación posible entre los analizados en este primer entrenamiento, una predicción del 65% aproximadamente, es mejorable de cara a obtener un modelo predictivo robusto y fiable, dada la sensibilidad del producto en el caso de no realizar un riego adecuado por disponer nuestro sistema de mucho sesgo.

Tal y como se observa en la matriz de confusión, se estima no regar erróneamente cuando sí correspondería en 18 ocasiones, de la misma forma que se indica al sistema regar en las mismas ocasiones cuando el evento debería ser no riego.

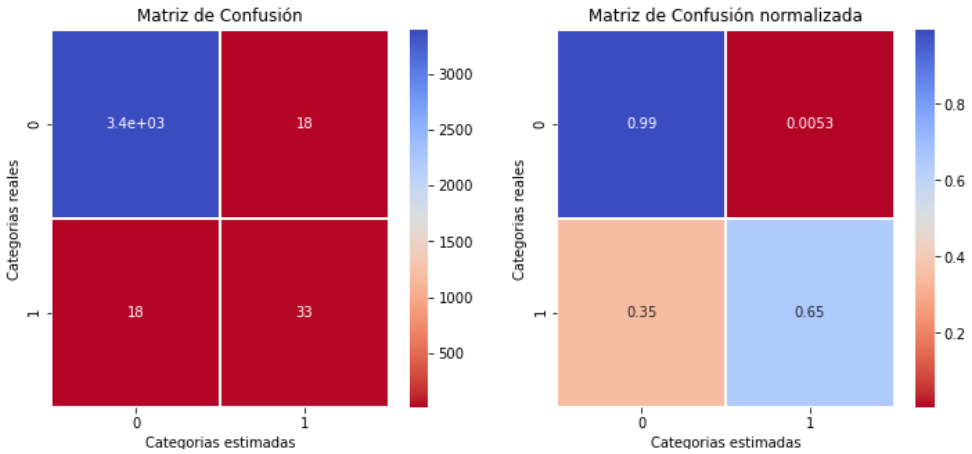

Figura 34: Matriz Confusión RandomForestClassifier E1

Por tanto, dichos resultados nos llevan a suponer que el agricultor no ha programado los riegos de acuerdo con un patrón meteorológico, en base a esto en los sucesivos entrenamientos se ha tenido en cuenta el intervalo temporal.

Así mismo, hay que indicar que dicha conclusión responde a lo que se podía intuir en el análisis de datos dado que para conjuntos de parámetros prácticamente idénticos (si no se considera el momento del riego) se obtienen muestras con valor riego igual 1 y otras con valor igual a 0, por lo cual muchos algoritmos acaban acertando a base de predecir el valor que se repite más, pero no son capaces de discriminar lógicamente una decisión u otra en base a este conjunto de variables.

5.2. Entrenamiento 2

De acuerdo con lo supuesto, en este segundo entrenamiento adicionalmente a las características principales, se incluye la hora.

En este caso, se verifica que aportando el intervalo temporal del riego los distintos algoritmos son capaz de predecir un patrón que se refleja en una mejora considerable de la precisión con respecto al primer entrenamiento realizado.

lr_multiple	svr	adr	bar	r_log	K_neig	svr	naive_bayes	adc	bac
0’04	-2’02	0’71	0’87	0’07	0’38	0’08	0’13	0’31	0’71

Figura 35: Comparativa modelos Entrenamiento 2

Tal y como podemos observar en la tabla anterior, es notable la mejora con respecto a los algoritmos de regresión donde la precisión de bosques aleatorios alcanza el 87% mientras que en el entrenamiento 1 no alcanzaba el 40%.

Así mismo, si observamos los algoritmos de clasificación y sus respectivas matrices de confusión podemos apreciar ese incremento con respecto a la primera simulación.

Si bien es verdad que, los modelos de regresión logística, máquina de vectores de soporte y Naive Bayes no consiguen modelar el comportamiento de irrigación debido a la escasez de muestras de riego en comparación con el volumen de datos de no riego, como era de esperar el modelo de bosques aleatorios es el más preciso de clasificación.

	Precisión	Recall	F1-score	Support
0	0’99	0’99	0’99	3414
1	0’71	0’64	0’67	47
Accuracy			0’99	3461

Figura 36: Report RandomForestClassifier E2

Así mismo, con respecto al entrenamiento 1 se observa mejora en los falsos positivos y negativos de dicho modelo como se observa en la gráfica siguiente.

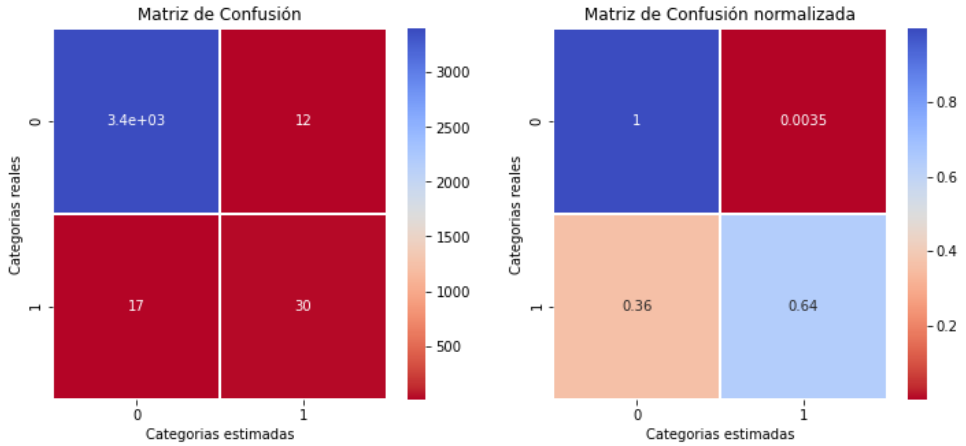

Figura 37: Matriz Confusión RandomForestClassifier E2

Análogamente al primer entrenamiento realizado, cabe mencionar que para conjuntos de parámetros prácticamente idénticos (como se vuelve a observar en este caso), muchos algoritmos acaban acertando a base de predecir el valor que se repite más, pero no son capaces de discriminar lógicamente una decisión u otra en base a este conjunto de variables.

5.3. Entrenamiento 3

En esta tercera simulación se introduce al sistema únicamente los intervalos temporales de riego para analizar la dependencia en ausencia de los parámetros meteorológicos.

En la siguiente tabla resumen y matrices de confusión, podemos comprobar que con respecto a la simulación anterior los algoritmos mejoran en ausencia de parámetros meteorológicos y de los datos del equipo de riego.

lr_multiple	svr	adr	bar	r_log	K_neig	svr	naive_bayes	adc	bac
0’04	-2’18	0’68	0’82	0’08	0’78	0’08	0’54	0’83	0’83

Figura 38: Comparativa modelos Entrenamiento 3

Tal y como se observa en los datos anteriores, KNeighborsClassifier, DecisionTreeClassifier y RandomForestClassifier incrementan notablemente su precisión con respecto al entrenamiento anterior.

- K vecinos más cercanos: predice casi un 80% de los riegos correctamente con una exactitud del modelo de 99%

	Precisión	Recall	F1-score	Support
0	1	1	1	3409
1	0’78	0’81	0’79	52
Accuracy			0’99	3461

Figura 39: Report KNeighborsClassifier E3

Su matriz de confusión está representada por el siguiente mapa de calor, donde se observa claramente el incremento de precisión con respecto a los anteriores entrenamientos realizados.

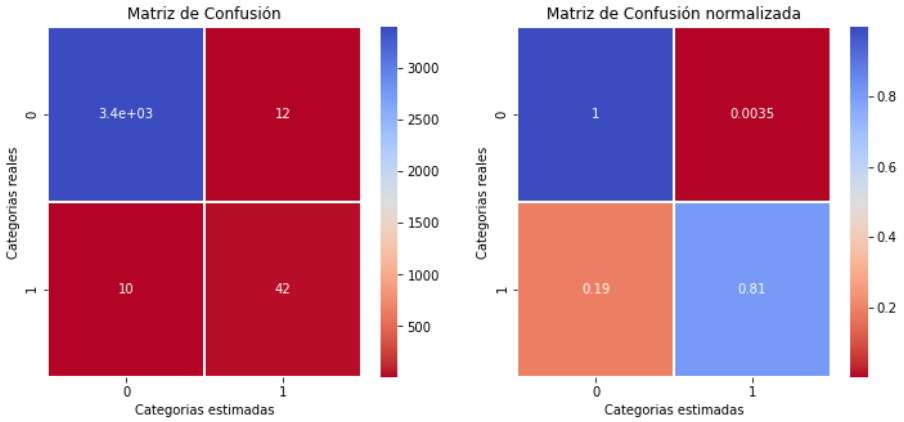

Figura 40: Matriz Confusión KNeighborsClassifier E3

- Árboles de decisión: predice correctamente un 83% de las muestras de riego con exactitud del 0’99.

	Precisión	Recall	F1-score	Support
0	1	1	1	3409
1	0’83	0’83	0’83	52
Accuracy			0’99	3461

Figura 41: Report KNeighborsClassifier E3

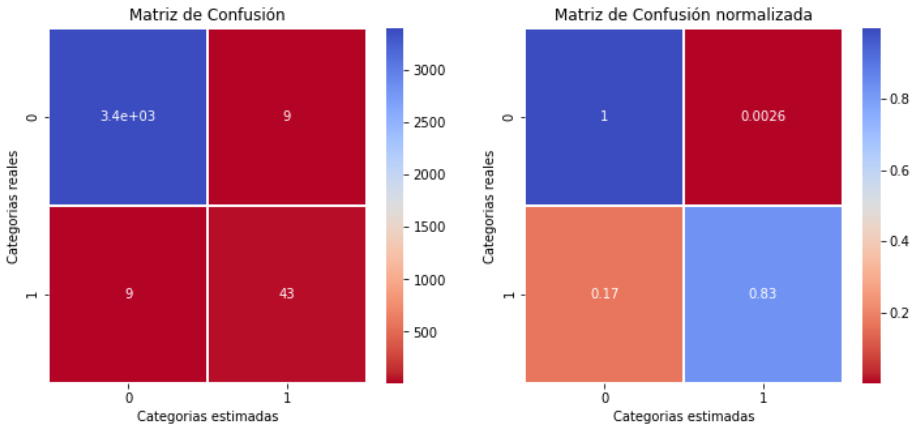

Figura 42: Matriz Confusión DecisionTreeClassifier E3

En base al análisis y simulaciones realizadas, tras entrenar al sistema mediante la entrada de distintos parámetros parece claro suponer que regar/no regar tiene dependencia con el intervalo temporal definido por el agricultor o regante más que con las características medioambientales en este dataset concreto utilizado.

Por tanto, podemos decir que por el momento dichos entrenamientos solo han servido para concluir algo que se podría conocer de antemano de forma sencilla, es decir, que el riego se ha programado con independencia de los parámetros medioambientales.

Así mismo, es importante mencionar que en el caso de poder aplicar este entrenamiento sobre un dataset correspondiente a un riego "más inteligente", donde se tengan en cuenta las condiciones medioambientales, o en el mismo en un periodo de tiempo más extenso, el entrenamiento sería igualmente válido y muy posiblemente las conclusiones serían diferentes, ya que los algoritmos serían capaces de detectar esos patrones de comportamiento más allá de las franjas horarias, y la metodología sería correcta.

Adicionalmente, por medio del cuarto entrenamiento se puede aprender a partir de este dataset la cantidad de riego, dato que posiblemente los propios agricultores no sean conscientes de ello, pero que han implementado, y haberlo medido permite ahora generar un modelo que les aporte valor.

5.4. Entrenamiento 4

Finalmente, en este cuarto entrenamiento se ha estimado el caudal acumulado diario, entrenando al sistema únicamente con el intervalo temporal y cuya respuesta sea el caudal acumulado diario.

En base a esto, aplicando algoritmos de regresión para poder obtener la cantidad o volumen de riego diario se observa que el algoritmo de bosques aleatorios presenta como era de esperar la mejor precisión de los modelos analizados.

lr_multiple	svr	adr	bar
0'21	-0'69	0'68	0'86

Figura 43: Comparativa modelos Entrenamiento 4

Como dato adicional, la media de los datos reales del acumulado diario registrado fue de 5373 litros y la predicción realizada por el modelo de Bosques aleatorios es de 4771 litros, estimando de esta manera aproximadamente un 88% del volumen correctamente.

A modo de resumen, se ha recopilado en la siguiente gráfica el resultado de los 4 entrenamientos realizados y la precisión observada en cada modelo supervisado con el objetivo de apreciar claramente la evolución del incremento de precisión en relación con el entrenamiento realizado.

E	lr_multiple	svr	adr	bar	r_log	K_neig	svm	naive_bayes	adc	bac
1	2%	0%	9%	35%	4%	50%	4%	47%	55%	65%
2	4%	0%	71%	87%	7%	38%	8%	13%	61%	71%
3	4%	0%	68%	82%	8%	78%	8%	54%	83%	83%
4	21%	0%	68%	86%						

Figura 44: Comparativa resultados

Por tanto, podemos decir que utilizando un modelo supervisado basado en bosques aleatorios, como hemos visto a lo largo de las simulaciones realizadas, el sistema reproduce con una buena precisión el comportamiento establecido en el calendario de riegos configurado en el equipo de telecontrol.

6. Conclusiones

A lo largo de este proyecto se ha llevado a cabo un análisis de los datos reales registrados por un equipo de telecontrol de riego facilitado por la Comunidad de Regantes de Najerilla, el cual ha proporcionado indicadores predictivos atendiendo a los entrenamientos presentados.

En base a esto, se implementaron como alternativas 4 modelos supervisados de regresión y 6 de clasificación en etapas de entrenamiento y validación con el conjunto de datos registrados por el equipo, en adición a parámetros meteorológicos proporcionados por el Gobierno de La Rioja en dicha ubicación de riego.

Tras evaluar los resultados de todas las simulaciones realizadas y comparar las predicciones en ambas clases, se observó que todos los clasificadores presentaron más dificultad en predecir la decisión de irrigación que de no realizar riego.

Cabe mencionar que, esto puede ser debido al alto número de falsos positivos (en algunos modelos más que en otros), derivado de un entrenamiento con una cantidad de muestras de riego activo insuficientes en relación con el volumen registrado de no irrigación.

Así mismo, de acuerdo con las simulaciones realizadas, se ha extraído la necesidad de incluir el intervalo temporal como parámetro de entrada en los modelos de aprendizaje supervisado, con el fin de determinar un patrón en la decisión de irrigación.

Adicionalmente, hay que indicar que podemos destacar la elección de implementación en nuestro sistema, del algoritmo de bosques aleatorios ya que construye diferentes conjuntos de entrenamiento y test sobre los mismos datos, de modo que se generan distintos árboles de decisión cuya unión forma un bosque aleatorio, creando un modelo más robusto y proporcionando mejores resultados de acuerdo con la investigación realizada.

Desde el punto de vista de producto, si bien es verdad que el modelo no atiende en gran medida a los aspectos meteorológicos como se ha observado, es importante destacar que modela correctamente el comportamiento humano por lo que automatiza y facilita al regante la gestión de la irrigación.

Como línea a futuro me gustaría valorar la posibilidad de alimentar al sistema con un entrenamiento ajustado a las necesidades de la planta, con el fin de observar la diferencia, no solo en la estimación de la clasificación riego/no riego sino poder evaluar el acumulado necesario para ello y analizar cuánto dista del modo de riego que se realiza por temporización.

En base a ello, poniendo en valor el producto desarrollado podremos orientar al pequeño agricultor hacia el IOT al observar que mediante modelos supervisados de aprendizaje automático es posible predecir el menor volumen de agua necesario y, por tanto, valorar el impacto positivo que obtendrá en su negocio.

Por tanto, podemos concluir que la utilización de algoritmos predictivos en la agricultura no solo predice el comportamiento del regante correctamente, sino que es un producto que favorece la automatización, el ahorro de costes y la gestión de un uso eficiente del agua y, por consiguiente, el impacto medioambiental.

7. Glosario

IOT	Internet Of Things
TIC	Tecnologías de la Información y las Comunicaciones
BBDD	Base de datos
WNS	Wireless sensor networks
MEMS	Sistemas microelectromecánicos
SW	Software
RTU	Remote Terminal Unit
HW	Hardware
OPC	OLE for Process Control
FAO	Food and Agriculture Organization
CCRR	Comunidad de Regantes
UNE	Asociación Española de Normalización
RTC	Real Time Connection
SPI	Serial Peripherical Interface.
M2M	Machine to Machine

8. Bibliografía

[1]https://es.unesco.org/themes/water-security/wwap/wwdr/2020#:~:text=El%20Informe%20Mundial%20de%20las,mejorando%20la%20gesti%C3%B3n%20del%20agua.
http://www.fao.org/news/story/es/item/1186505/icode/

[2] https://depurtotal.es/optimizacion-del-consumo-de-agua-para-el-riego/

[3] Historia ancestral del riego agrícola, TRACXO (2010)

[4] Análisis de la distribución de agua en sistemas de riego por aspersión, por Jesús Montero Martínez (2000)

[5] https://www.redagricola.com/cl/pivotes-la-revolucion-circular-del-riego/

[6] https://www.agroptima.com/es/blog/riego-inteligente

[7] Smart city and IoT, T Kim, C Ramos, S Mohammed - 2017

[8] Redes inalámbricas de sensores: teoría y aplicación práctica/ Roberto Fernández Martínez…[et al.] (Integrantes del Grupo de Investigación EDMANS)-[Logroño]: Universidad de la Rioja,2009

[9] OPC unified architecture, W Mahnke, SH Leitner, M Damm - 2009

[10]http://crea.uclm.es/crea/descargas/_files/El_Riego_y_sus_Tecnologias.pdf

[11] http://www.fao.org/3/ai128s/ai128s02.pdf

[12] La densidad aparente y sus implicaciones agrícolas en el proceso expansión/contracción del suelo, Américo J. Hossne G. (2007)-> http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0187-57792008000300001

[13] El riego y sus tecnologías, Luis Santos Pereira (2010)

[14] https://www.apd.es/que-es-machine-learning

Machine learning: a probabilistic perspective, KP Murphy (2012)

[15] Siegel, Eric (2013). Predictive Analytics. The power to predict who will click, buy, lie, or die.

[16] J. A. Freitez, “Desarrollo de un Modelo Predictivo del Brote de la Sigatoka Negra para las Plantaciones de Plátanos al Sur del Lago de Maracaibo,” vol. 9, no. 1, pp. 191–198, 2009.

[17] https://www.iartificial.net/regresion-lineal-con-ejemplos-en-python/

[18] https://www.revistaseden.org/files/14-cap%2014.pdf

[19] Breve aproximación a la técnica de árbol de decisiones, C.Zuniga, N. Abgar (2011)

[20] Redes Neuronales: Conceptos básicos y aplicaciones, Damián Jorge Matich (2011)

[21] Modelos de Clasificación basados en Máquinas de Vectores Soporte, L. González Abril (2003)

[22] Evaluación del clasificador basado en los "k" vecinos más cercanos para la localización de la zona en falla en los sistemas de potencia, Juan José Mora Flórez, Germán Morales España, Rene Barrera Cárdenas, Ingeniería e Investigación, ISSN 0120-5609, ISSN-e 2248-8723, Vol. 28, N.º. 3, 2008

[23] https://enviraiot.es/

[24] risiberia.es/blog/nuevo_sistema_avanzado_de_telecontrol_de_riego/

[25] https://orionis-iot.com/

[26] https://www.xatakahome.com/domotica/blossom-el-sistema-de-riego-inteligente-que-utiliza-las-predicciones-meteorologicas

[27] https://www.hunterindustries.com/es/que-es-un-programador-inteligente

[28] https://www.tysmagazine.com/bynse-la-solucion-big-data-la-agricultura/

[29]https://www.vyrsa.com/es/catalogo/categorias/control-riego/sistemas-de-tele-gestion/skydrop/

[30] https://www.odins.es/proyecto-iotas/

[31] https://www.une.org/

[32] A. Ruiz-Canales[1], M. J. Oates[1], J.J. Pérez Solano[2], J.M. Molina-Martínez[3] [1] Departamento de Ingeniería, Escuela Politécnica Superior de Orihuela (EPSO-UMH)[2] Departamento de Informática, Escuela Técnica Superior de Ingeniería, Universidad de Valencia [3] Grupo de Investigación de Ingeniería Agromótica y del Mar (UPCT)

[33] Diseño e Implementación de un Sistema de Riego Inteligente basado en Sensores y Módulos de Radiofrecuencia para Transmisión y Sistema de Control, José Andrés Muñoz Arcentales, Vanessa Calero Bravo, Ignacio Marín-García, Nov. 2013

https://www.researchgate.net/publication/258441493_Diseno_e_Implementacion_de_un_Sistema_de_Riego_Inteligente_basado_en_Sensores_y_Modulos_de_Radiofrecuencia_para_Transmision_y_Sistema_de_Control

[34] Contribución a las redes de sensores inalámbricas. Estudio e implementación de soluciones hardware para agricultura de precisión, López Riquelme, Juan Antonio, 2011. URI: http://hdl.handle.net/10317/2244

[35] Electronic system design based upon XBee modules for transmitting voice signals, Ing. Alexis Castellanos Rodríguez, Ing. Franklin Padrón Quindemil, Ing. Frank Martínez Suárez, Dr. C. Ángel Damián Bárzaga Varela, Ing. Luis Alejandro Madruga Milanés, 2015

[36] https://es.rs-online.com/web/p/arduino/7154081/?cm_mmc=ES-PLA-DS3A-_-google-_-CSS_ES_ES_Raspberry_Pi_%26_Arduino_y_M%C3%B3dulos_de_Desarrollo_Whoop-_-(ES:Whoop%21)+Arduino-_-7154081&matchtype=&pla-339391921421&gclid=EAIaIQobChMItfH93oWW7QIVApntCh2khAX9EAQYBSABEgJMJPD_BwE&gclsrc=aw.ds

[37] https://es.rs-online.com/web/p/sensores-de-humedad-y-temperatura/1594685/?cm_mmc=ES-PLA-DS3A-_-google-_-CSS_ES_ES_Semiconductores_Whoop-_-(ES:Whoop!)+Sensores+de+Humedad+y+Temperatura-_-1594685&matchtype=&aud-821594433763:pla-321775116422&gclid=EAIaIQobChMI597EzI6W7QIVFJ7VCh1EgAAeEAQYAyABEgL0ZvD_BwE&gclsrc=aw.ds

[38] https://blog.330ohms.com/2020/02/16/como-conectar-un-sensor-de-temperatura-lm35-a-arduino/

[39] https://es.aliexpress.com/item/32562744759.html?src=google&albch=shopping&acnt=439-079-4345&isdl=y&slnk=&plac=&mtctp=&albbt=Gploogle_7_shopping&aff_atform=google&aff_short_key=UneMJZVf&gclsrc=aw.ds&&albagn=888888&&ds_e_adid=475881929441&ds_e_matchtype=&ds_e_device=c&ds_e_network=u&ds_e_product_group_id=743612850954&ds_e_product_id=es32562744759&ds_e_product_merchant_id=108379374&ds_e_product_country=ES&ds_e_product_language=es&ds_e_product_channel=online&ds_e_product_store_id=&ds_url_v=2&ds_dest_url=https://es.aliexpress.com/item/32562744759.html?&albcp=11492995081&albag=117794251848&gclid=EAIaIQobChMIgeD_hZGW7QIVC7DtCh3--AirEAQYAyABEgJepfD_BwE

[40] https://www.luisllamas.es/arduino-humedad-suelo-fc-28/

[41] https://www.tiendatec.es/arduino/modulos/400-modulo-zs-042-reloj-en-tiempo-real-rtc-basado-en-ds3231-at24c32-arduino-y-raspberry-pi-8404001180013.html

[42] https://programarfacil.com/blog/arduino-blog/reloj-con-arduino-rtc/

[43] https://www.amazon.es/dp/B01MR7B3TT?tag=tarjetasdememoria.info-21&linkCode=ogi&th=1&psc=1

[44] https://www.tarjetasdememoria.info/micro-sd-para-arduino/

[45] https://es.rs-online.com/web/p/herramientas-de-desarrollo-inalambricas-y-de-comunicacion/1359730/?cm_mmc=ES-PLA-DS3A-_-google-_-PLA_ES_ES_Raspberry_Pi_%26_Arduino_y_M%C3%B3dulos_de_Desarrollo_Whoop-_-(ES:Whoop!)+Herramientas+de+Desarrollo+Inal%C3%A1mbricas+y+de+Comunicaci%C3%B3n-_-1359730&matchtype=&aud-813230962291:pla-338945524988&gclid=EAIaIQobChMIqtzlm-aW7QIVhfdRCh1KbwDMEAYYASABEgKWmPD_BwE&gclsrc=aw.ds

[46] https://aprendiendoarduino.wordpress.com/2016/11/16/zigbeexbee/#:~:text=ZigBee%20es%20el%20nombre%20de,personal%20area%20network%2C%20WPAN).

[47] https://proyectosconarduino.com/curso/maneras-de-alimentar-el-arduino-uno/

[48] https://www.luisllamas.es/arduino-dht11-dht22/

[49] https://www.luisllamas.es/medir-presion-del-aire-y-altitud-con-arduino-y-barometro-bmp180

[50] https://www.luisllamas.es/arduino-lluvia/

[51] https://www.instructables.com/SIM900-GSM-GPRS-SHIELD-CON-ARDUINO-UNO

[52] https://ticnegocios.camaravalencia.com/servicios/tendencias/caminar-con-exito-hacia-la-industria-4-0-capitulo-11-infraestructuras-i-redes-inalambricas/#1530172407572-55ff5094-ef11

[53] https://www.efor.es/sites/default/files/tecnologias-de-comunicacioon-para-iot.pdf

[54] https://www.larioja.org/emergencias-112/es/meteorologia/datos-actuales-rioja/detalle-estacion?homepage=12&cod_muni=3

9. Anexos

9.1. Código Preprocesado de datos

a) Lectura del fichero de datos

#Cargamos datasets

```
dataset_RTU=pd.read_excel('Sigma5_T63.xlsx')

dataset_weather=pd.read_excel('Datos_metereológicos_Haro.xlsx')
```

#Ordenar por fecha

```
dataset_RTU=dataset_RTU.sort_values(by=['Fecha'],ascending=[True])
```

b) Análisis del tipo de variables que lo componen

```
dataset_RTU.head() # Comprobamos las primeras filas

dataset_RTU.info() #RangeIndex: 9212 entries, 0 to 9211// Variable gráfica y Unidades son variables categoricas y Fecha datetime64 y Valor float64

dataset_weather.head() #RangeIndex: 8797 entries, 0 to 8796

dataset_weather.info() #No hay variables categóricas solo enteros, float y datetime
```

c) Linealizamos las variables categóricas para poder entrenar los modelos

```
cabecera_weather=['Fecha','Hrm','Pluviometria','Pm','Rad_max','Rad_med','Tai_max','Tai_med','Tai_min','Ti_max','Ti_med','Ti_min','Vv_max','Vv_maxk','Vv_med','Vv_medk']

dataset_weather.columns=cabecera_weather

dataset_weather=dataset_weather.rename({'Fecha-Hora':'Fecha',},axis=1)

dataset_RTU.set_index("Variable gráfica",inplace=True)

cobertura=dataset_RTU.loc['T-63-Nivel señal GPRS'].rename({'Valor':'Cobertura_RTU',},axis=1).iloc[:,:-1]

cobertura.reset_index(level=0,inplace=True)

cobertura=cobertura[['Fecha','Cobertura_RTU']]

temperatura=dataset_RTU.loc['T-63 -temperatura'].rename({'Valor':'Temperatura_RTU',},axis=1).iloc[:,:-1]

temperatura.reset_index(level=0,inplace=True)

temperatura=temperatura[['Fecha','Temperatura_RTU']]

bateria=dataset_RTU.loc['T-63-Nivel de batería'].rename({'Valor':'Bateria_RTU',},axis=1).iloc[:,:-1]

bateria.reset_index(level=0,inplace=True)

bateria=bateria[['Fecha','Bateria_RTU']]
```

```
VG1_riego_act=dataset_RTU.loc['VG1- Riego
activo'].rename({'Valor':'Riego_act',},axis=1).iloc[:,:-1]

VG1_riego_act.reset_index(level=0,inplace=True)

VG1_riego_act=VG1_riego_act[['Fecha','Riego_act']]

VG1=dataset_RTU.loc['VG1'].rename({'Valor':'Apertura_VG',},axis=1).iloc[:,:-1]

VG1.reset_index(level=0,inplace=True)

VG1=VG1[['Fecha','Apertura_VG']]

Paso_Caudal=dataset_RTU.loc['VG1- Paso de
caudal'].rename({'Valor':'Paso_Caudal',},axis=1).iloc[:,:-1]

Paso_Caudal.reset_index(level=0,inplace=True)

Paso_Caudal=Paso_Caudal[['Fecha','Paso_Caudal']]

VG1_Caudal=dataset_RTU.loc['VG1- Caudal'].rename({'Valor':'Caudal',},axis=1).iloc[:,:-1]

VG1_Caudal.reset_index(level=0,inplace=True)

VG1_Caudal=VG1_Caudal[['Fecha','Caudal']]

Velocidad_Caudal=dataset_RTU.loc['VG1- Velocidad
(Q/S)'].rename({'Valor':'Velocidad_Caudal',},axis=1).iloc[:,:-1]

Velocidad_Caudal.reset_index(level=0,inplace=True)

Velocidad_Caudal=Velocidad_Caudal[['Fecha','Velocidad_Caudal']]
```

#Se descartan T1,T2, T3, T4, VG1 alarma de caudal, VG1 Estado RedePlan y VG1 Id de riego en memoria

d) Limpieza de datos

```
def remove_duplicates(samples,name_param): #Eliminación de muestras duplicadas en
variables independientes

    fin=samples.loc[0,'Fecha']

    df_param=samples.copy()

    media_param=samples[name_param].mean()

    for i in range(1,len(samples)):

        if samples.loc[i,'Fecha']==samples.loc[i-1,'Fecha'] and samples.loc[i,'Fecha']>fin:

            dif_media_i=abs(samples.loc[i,name_param]-media_param)

            dif_media_anterior=abs(samples.loc[i-1,name_param]-media_param)

            if dif_media_i<dif_media_anterior:

                df_param.drop([i-1],inplace=True)
```

```
        else:
            df_param.drop([i],inplace=True)
    df_param=df_param.sort_values(by=['Fecha'],ascending=[True])
    df_param.reset_index(inplace=True)
    df_param.drop(columns='index',inplace=True)
    return df_param


def remove_null (samples,name_param): #Eliminación de muestras nulas en variables independientes
    df_param=samples.copy()
    for i in range(1,len(samples)):
        if samples.loc[i,name_param]==0:
            df_param.drop([i],inplace=True)
    df_param=df_param.sort_values(by=['Fecha'],ascending=[True])
    df_param.reset_index(inplace=True)
    df_param.drop(columns='index',inplace=True)
    return df_param


def remove_duplicates_event (samples,name_param):  #Eliminación de muestras duplicadas, cotejamos con la muestra de la hora anterior y nos quedamos con el cambio de evento
    fin=samples.loc[0,'Fecha']
    df_param=samples.copy()
    for i in range(1,len(samples)):
        if samples.loc[i,'Fecha']==samples.loc[i-1,'Fecha'] and samples.loc[i,'Fecha']>fin:
            print(i)
            if samples.loc[i,name_param]==samples.loc[i-1,name_param]:
                df_param.drop([i],inplace=True)
            elif samples.loc[i-1,name_param]==samples.loc[i-2,name_param]:
                df_param.drop([i-1],inplace=True)
            elif samples.loc[i,name_param]==samples.loc[i-2,name_param]:
                df_param.drop([i],inplace=True)
```

```
    return df_param


def merge_datasets (dataset1,dataset2):
    df=pd.merge(dataset1,dataset2,on="Fecha",how="outer")
    df=df.sort_values(by=['Fecha'],ascending=[True])
    df.reset_index(inplace=True)
    df.drop(columns='index',inplace=True)
    return df


def remove_duplicates_flow (df1,df2,name_param1,name_param2):
    media_param=df1[name_param1].mean()
    fin=df2.loc[0,'Fecha']
    df_param=df2.copy()
    for i in range(1,len(df2)):
        if df2.loc[i,'Fecha']==df2.loc[i-1,'Fecha'] and Paso_Caudal_Caudal.loc[i,'Fecha']>fin:
            if int(df2.loc[i,name_param1]) != 0:
                if int(df2.loc[i-1,name_param1]) != 0:
                    if int(df2.loc[i-1,name_param1])==int(df2.loc[i,name_param1]):
                        df_param.drop([i-1],inplace=True)
                    else:
                        dif_media_i=abs(df2.loc[i,name_param1]-media_param)
                        dif_media_anterior=abs(df2.loc[i-1,name_param1]-media_param)
                        if dif_media_i<dif_media_anterior:
                            df_param.drop([i-1],inplace=True)
                        else:
                            df_param.drop([i],inplace=True)
                elif int(df2.loc[i,name_param2]) == 1:
                    df_param.drop([i-1],inplace=True)
                else:
                    df_param.drop([i],inplace=True)
```

```
            elif int(df2.loc[i,name_param2]) == 1:
                df_param.drop([i],inplace=True)
            else:
                df_param.drop([i-1],inplace=True)
    df_param=df_param.sort_values(by=['Fecha'],ascending=[True])
    df_param.reset_index(inplace=True)
    df_param.drop(columns='index',inplace=True)
    return df_param

def compare_flow (df1,name_param1,name_param2):
    for i in range(1,len(df1)):
        if int(df1.loc[i,name_param1])!= 0:
            df1.at[i,name_param2]=1
        else:
            df1.at[i,name_param2]=0
    return df1

def corregir_datos_faltantes_riego(df_riego):
    df_riego_aux=df_riego.copy()
    for i in range(1,len(df_riego_aux)):
        if  math.isnan(df_riego_aux.loc[i,'Caudal'])==False:
            if int (df_riego_aux.loc[i,'Caudal']) > 0:
                df_riego.at[i,'Riego_act']=1
                df_riego.at[i,'Apertura_VG']=1
            else:
                df_riego.at[i,'Riego_act']=0
                df_riego.at[i,'Apertura_VG']=0
        elif math.isnan(df_riego_aux.loc[i,'Riego_act'])==False:
            if int (df_riego_aux.loc[i,'Riego_act']) == 0:
                df_riego.at[i,'Caudal']=0
```

```
            df_riego.at[i,'Velocidad_Caudal']=0
            df_riego.at[i,'Paso_Caudal']=0
            df_riego.at[i,'Apertura_VG']=0
        else:
            df_riego.at[i,'Caudal']=0
            df_riego.at[i,'Velocidad_Caudal']=0
            df_riego.at[i,'Paso_Caudal']=0
            df_riego.at[i,'Riego_act']=0
            df_riego.at[i,'Apertura_VG']=0
    else:
        df_riego.at[i,'Caudal']=0
        df_riego.at[i,'Velocidad_Caudal']=0
        df_riego.at[i,'Paso_Caudal']=0
        df_riego.at[i,'Riego_act']=0
        df_riego.at[i,'Apertura_VG']=0
  return df_riego


def rellenar_datos_faltantes(df,name_param):
  while df[name_param].isnull().sum()>0:
    df[name_param] = df[name_param].fillna(method='pad', limit=1)
  return df
#Limpieza Cobertura
df_cobertura=remove_duplicates(cobertura, 'Cobertura_RTU')
df_cobertura=remove_null(df_cobertura, 'Cobertura_RTU')
#Limpieza Temperatura
df_temperatura=remove_duplicates(temperatura, 'Temperatura_RTU')
df_temperatura=remove_null(df_temperatura, 'Temperatura_RTU')
#Limpieza Bateria
df_bateria=remove_duplicates(bateria, 'Bateria_RTU')
df_bateria=remove_null(df_bateria, 'Bateria_RTU')
```

#Limpieza Riego activo

df_VG1_riego_act=remove_duplicates_event(VG1_riego_act, 'Apertura_VG')

#Limpieza Válvula General

df_VG1=remove_duplicates_event(VG1, 'Apertura_VG')

#Limpieza Paso Caudal

df_Paso_Caudal=remove_duplicates_event(Paso_Caudal, 'Paso_Caudal')

#Limpieza Caudal de la Válvula General para eliminar los duplicados de caudal se debe tener en cuenta si está activo el paso de caudal porque en el cierre podemos tener caudal x y 0 y en la apertura 0 y x, por el cambio de evento. Por tanto, unimos Caudal y flag de Paso de Caudal

Paso_Caudal_Caudal=merge_datasets(df_Paso_Caudal, VG1_Caudal)

df_Caudal_Paso=remove_duplicates_flow(VG1_Caudal,Paso_Caudal_Caudal,'Caudal','Paso_Caudal')

df_Caudal=df_Caudal_Paso[['Fecha','Caudal']]

df_Caudal_Paso=compare_flow(df_Caudal_Paso,'Caudal','Paso_Caudal')

#Limpieza Velocidad caudal de la Válvula General para eliminar los duplicados de velocidad caudal se debe tener en cuenta si está activo el paso de caudal porque en el cierre podemos tener velocidad caudal x y 0 y en la apertura 0 y x, por el cambio de evento. Por tanto, unimos Velocidad Caudal y flag de Paso de Caudal

Paso_Caudal_Velocidad=merge_datasets(df_Paso_Caudal, Velocidad_Caudal)

df_Velocidad_Paso=remove_duplicates_flow(Velocidad_Caudal,Paso_Caudal_Velocidad,'Velocidad_Caudal','Paso_Caudal')

df_Velocidad_Caudal=df_Velocidad_Paso[['Fecha','Velocidad_Caudal']]

df_Velocidad_Paso=compare_flow(df_Velocidad_Paso,'Velocidad_Caudal','Paso_Caudal')

#Unir datasets

df_RTU=merge_datasets(df_Caudal_Paso, df_Velocidad_Caudal)

df_RTU=merge_datasets(df_VG1, df_RTU)

df_RTU=merge_datasets(df_VG1_riego_act, df_RTU)

df_RTU=merge_datasets(df_cobertura, df_RTU)

df_RTU=merge_datasets(df_bateria, df_RTU)

df_RTU=merge_datasets(df_temperatura, df_RTU)

df=merge_datasets(df_RTU, dataset_weather)

#Limpieza de datos faltantes

```
df=corregir_datos_faltantes_riego(df)
df=rellenar_datos_faltantes(df,'Cobertura_RTU')
df=rellenar_datos_faltantes(df,'Bateria_RTU')
df=rellenar_datos_faltantes(df,'Temperatura_RTU')
df=rellenar_datos_faltantes(df,'Caudal')
df=rellenar_datos_faltantes(df,'Velocidad_Caudal')
df=rellenar_datos_faltantes(df,'Hrm')
df=rellenar_datos_faltantes(df,'Pluviometria')
df=rellenar_datos_faltantes(df,'Pm')
df=rellenar_datos_faltantes(df,'Rad_max')
df=rellenar_datos_faltantes(df,'Rad_med')
df=rellenar_datos_faltantes(df,'Tai_max')
df=rellenar_datos_faltantes(df,'Tai_med')
df=rellenar_datos_faltantes(df,'Tai_min')
df=rellenar_datos_faltantes(df,'Ti_max')
df=rellenar_datos_faltantes(df,'Ti_med')
df=rellenar_datos_faltantes(df,'Ti_min')
df=rellenar_datos_faltantes(df,'Vv_max')
df=rellenar_datos_faltantes(df,'Vv_maxk')
df=rellenar_datos_faltantes(df,'Vv_med')
df=rellenar_datos_faltantes(df,'Vv_medk')
df.dropna(inplace=True)
```

#Guardar dataset conjunto en fichero

```
df.to_excel('dataset.xlsx')
```

9.2. Código Análisis de variables

a) Graficar variables para ver relación o aporte de información

Elimino Apertura_VG y Paso_Caudal porque como vimos en la limpieza son iguales a Riego_act

df.drop(columns=['Apertura_VG','Paso_Caudal'],inplace=True)

y=df.Riego_act # Separamos el resultado (Riego/No Riego)

x=df.drop(['Riego_act'],axis=1) # Quitamos la variable Riego_act,Paso_Caudal y Apertura_VG

Para facilitar la visualización de resultados, debido al alto número de variables, crearemos las gráficas

x_scaled=(x-x.min())/(x.max()-x.min())

sub_df1=pd.concat([y,x_scaled.iloc[:,0:10]],axis=1)

sub_df2=pd.concat([y,x_scaled.iloc[:,10:20]],axis=1)

sub_df11=pd.melt(sub_df1,id_vars="Riego_act",var_name="Variable",value_name='Valor')

sub_df22=pd.melt(sub_df2,id_vars="Riego_act",var_name="Variable",value_name='Valor')

plt.figure(figsize=(20,8))

sns.violinplot(x="Variable",y="Valor",hue="Riego_act",data=sub_df11, split=True)

plt.figure(figsize=(20,8))

sns.violinplot(x="Variable",y="Valor",hue="Riego_act",data=sub_df22, split=True)

#Observamos que muchas variables la distribución uniforme entre riego y no riego es uniforme, y por tanto apenas aportar valor para discriminar en el modelo. Por ello, discriminamos Pluviometria y Pm. Se crea el nuevo DataFrame con solo las características que contribuyen a la discriminación de riego/no riego

df_features=df.drop(['Pluviometria','Pm'],axis=1)

b) Normalización de datos

#Para que no distorsionen el entrenamiento del modelo, se identifican las columnas cuyo valor mínimo es menor que -1 y valor máximo mayor que 1 y se normalizan

tt = df_features.describe().transpose()

tt[(tt['max']>1) & (tt['min']< -1)]# No es necesario normalizar

c) Correlación entre las variables

#De forma ideal buscamos incluir en el modelo todas las variables que aporten una componente explicativa alta a la variable objetivo y que tengan una baja correlación entre ellas. Dos variables con una alta correlación significan que, con un grado de error bajo, una puede ser inferida a partir de la otra, y por tanto aportarán información redundante al modelo. Analizamos para ello la matriz de correlacción entre las variables, buscando aquellas variables con un nivel de correlación alto.

corr_base = df_features.corr() #generamos la matriz de correlación

plt.figure(figsize=(12, 10))

```
sns.heatmap(corr_base[(corr_base >= 0.5) | (corr_base <= -0.4)],

        cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,

        annot=True, annot_kws={"size": 8}, square=True);
```

Se observa que todas las temperaturas medidas por la estación meteorológica están muy correlacionadas con la temperatura de la RTU, y por tanto las eliminamos. Así mismo, la velocidad del viento (4 variables, están muy correlacionadas, nos quedamos con Vvmed). Rad_max y Rad_min están correlacionadas, me quedo con Rad_med y Riego_act está correlacionada con el caudal.

#Asi mismo , 'Velocidad_Caudal' está correlacionada con Caudal por lo que la descartamos también.

```
df_features2=df_features[['Fecha','Temperatura_RTU','Cobertura_RTU','Bateria_RTU','Rad_me
d','Vv_med']]

corr_base2 = df_features2.corr()  #generamos la matriz de correlación

plt.figure(figsize=(12, 10))

sns.heatmap(corr_base2[(corr_base2 >= 0.5) | (corr_base2 <= -0.4)],

        cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,

        annot=True, annot_kws={"size": 8}, square=True);

df_features2.to_excel('Características_principales.xlsx')

df_param=df_features[['Fecha','Riego_act','Temperatura_RTU','Cobertura_RTU','Bateria_RTU','
Rad_med','Vv_med']]
```

d) Relación acumulado diario

Función que formatea la fecha de un dataset e incorpora columnas con día, mes, año...

```
def format_fecha (df,df_aux):

   i=0

   for fecha in df.Fecha:

      test = df_aux.loc[df_aux.index[i],'Fecha']

      year = pd.to_numeric(test.strftime('%Y'))

      month = pd.to_numeric(test.strftime('%m'))

       day = pd.to_numeric(test.strftime('%d'))

      hour = pd.to_numeric(test.strftime('%H'))

      minute = pd.to_numeric(test.strftime('%M'))

      second = pd.to_numeric(test.strftime('%S'))
```

```
        df_aux.loc[df_aux.index[i], 'WEEK_DAY'] = date.datetime(year,month,day).weekday()
        df_aux.loc[df_aux.index[i], 'MONTH_DAY'] = day
        df_aux.loc[df_aux.index[i], 'MONTH'] = month
        df_aux.loc[df_aux.index[i], 'YEAR'] = year
        df_aux.loc[df_aux.index[i], 'HOUR'] = hour
        df_aux.loc[df_aux.index[i], 'MINUTE'] = minute
        df_aux.loc[df_aux.index[i], 'SECOND'] = second
        i=i+1
    return df_aux
```

Función que crea un nuevo dataframe con el acumulado diario y la fecha

```
def dataframe_acumulado (df,df_aux):
    format_fecha (df,df_aux)
    j=0
    Acumulado=0
    df_acumulado=pd.DataFrame(columns=('WEEK_DAY','MONTH_DAY',  'MONTH',  'YEAR', 'ACUMULADO'))
    fin=df_aux.loc[len(df_aux)-1,'Fecha']
    for i in range(1,len(df_aux)):
        if df_aux.loc[i,'MONTH_DAY']==df_aux.loc[i-1,'MONTH_DAY']:
            if df_aux.loc[i,'Fecha']<fin:
                if df_aux.loc[i,'MONTH']==df_aux.loc[i-1,'MONTH']:
                    if df_aux.loc[i,'YEAR']==df_aux.loc[i-1,'YEAR']:
                        if int(df_aux.loc[i,'Caudal'])>0 and df_aux.loc[i+1,'Fecha']<fin:
                            horas=abs(int(df_aux.loc[df_aux.index[i+1], 'HOUR'])-
int(df_aux.loc[df_aux.index[i],'HOUR']))
                            minutos=abs(int(df_aux.loc[df_aux.index[i+1], 'MINUTE'])-
int(df_aux.loc[df_aux.index[i],'MINUTE']))
                            segundos=abs(int(df_aux.loc[df_aux.index[i+1], 'SECOND'])-
int(df_aux.loc[df_aux.index[i],'SECOND']))
                            duracion=3600*horas+60*minutos+segundos
                            caudal=df_aux.loc[i,'Caudal']
                            Acumulado=Acumulado+duracion*caudal
```

```
        else:
            df_acumulado.loc[j, 'ACUMULADO'] = Acumulado
            df_acumulado.loc[j, 'WEEK_DAY'] = int(df_aux.loc[i-1,'WEEK_DAY'])
            df_acumulado.loc[j, 'MONTH_DAY'] = int(df_aux.loc[i-1,'MONTH_DAY'])
            df_acumulado.loc[j, 'MONTH'] =int(df_aux.loc[i-1,'MONTH'])
            df_acumulado.loc[j, 'YEAR'] = int(df_aux.loc[i-1,'YEAR'])
    else:
        df_acumulado.loc[j, 'ACUMULADO'] = Acumulado
        df_acumulado.loc[j, 'WEEK_DAY'] = int(df_aux.loc[i-1,'WEEK_DAY'])
        df_acumulado.loc[j, 'MONTH_DAY'] = int(df_aux.loc[i-1,'MONTH_DAY'])
        df_acumulado.loc[j, 'MONTH'] =int(df_aux.loc[i-1,'MONTH'])
        df_acumulado.loc[j, 'YEAR'] = int(df_aux.loc[i-1,'YEAR'])
        j=j+1
        Acumulado=0
  return df_acumulado
```

9.3. Código Análisis métodos predictivos

```
def plot_confusion_matrix(y_test, pred): #Función que imprime la matriz de confusión y el reporte de clasificación
  y_test_legit = y_test.value_counts()[0]
  y_test_fraud = y_test.value_counts()[1]
  cfn_matrix = confusion_matrix(y_test, pred)
  cfn_norm_matrix = np.array([[1.0 /
y_test_legit,1.0/y_test_legit],[1.0/y_test_fraud,1.0/y_test_fraud]])
  norm_cfn_matrix = cfn_matrix * cfn_norm_matrix
  fig = plt.figure(figsize=(12,5))
  ax = fig.add_subplot(1,2,1)
  sns.heatmap(cfn_matrix,cmap='coolwarm_r',linewidths=0.5,annot=True,ax=ax)
  plt.title('Matriz de Confusión')
  plt.ylabel('Categorias reales')
  plt.xlabel('Categorias estimadas')
  ax = fig.add_subplot(1,2,2)
```

```
    sns.heatmap(norm_cfn_matrix,cmap='coolwarm_r',linewidths=0.5,annot=True,ax=ax)
    plt.title('Matriz de Confusión normalizada')
    plt.ylabel('Categorias reales')
    plt.xlabel('Categorias estimadas')
    plt.show()
    print('---Matriz de Confusión---')
    print(cfn_matrix)
    print()
    print('---Report de clasificación---')
    print(classification_report(y_test,pred))


def print_precision_model (precision,algoritmo,i):
    if i==1:
        print('DATOS DEL MODELO LINEAL MULTIPLE')
        print()
        print('Valor de las pendientes o coeficientes "a":')
        print(algoritmo.coef_)
        print()
        print('Valor de las pendientes o coeficientes "b":')
        print(algoritmo.intercept_)
      if i==2:
        print('DATOS DEL MODELO VECTORES DE SOPORTE REGRESIÓN')
    if i==3:
        print('DATOS DEL MODELO ÁRBOLES DE DECISIÓN REGRESIÓN')
    if i==4:
        print('DATOS DEL MODELO BOSQUES ALEATORIOS REGRESIÓN')
    if i==5:
        print('DATOS DEL MODELO REGRESIÓN LOGÍSTICA')
    if i==6:
        print('DATOS DEL MODELO K VECINOS MAS CERCANOS')
```

```
    if i==7:
        print('DATOS DEL MODELO MÁQUINA VECTORES SOPORTE CLASIFICACIÓN')
    if i==8:
        print('DATOS DEL MODELO NAIVE BAYES')
    if i==9:
        print('DATOS DEL MODELO ÁRBOLES DE DECISIÓN CLASIFICACIÓN')
    if i==10:
        print('DATOS DEL MODELO BOSQUES ALEATORIOS CLASIFICACIÓN')
    print()
    print('Precisión del modelo:')
    print(precision)
    print()
def analisis_algoritmos_regresion (X,y):
    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
    for i in range (1,5):
        #ALGORITMOS DE REGRESION
        if i==1: #REGRESIÓN LINEAL MULTIPLE
            lr_multiple=linear_model.LinearRegression()
            lr_multiple.fit(X_train,y_train)
            Y_pred_multiple=lr_multiple.predict(X_test)
            precision_mul=lr_multiple.score(X_train,y_train)
            print_precision_model(precision_mul,lr_multiple,i)
        if i==2: #MODELO VECTORES DE SOPORTE REGRESIÓN
            svr=SVR(C=1.0,epsilon=0.2)
            svr.fit(X_train,y_train)
            Y_pred_svr=svr.predict(X_test)
            precision_svr=svr.score(X_train,y_train)
            print_precision_model(precision_svr,svr,i)
        if i==3: #ARBOL DE DECISION REGRESION
            adr=DecisionTreeRegressor(max_depth=5)
```

```
            adr.fit(X_train,y_train)

            Y_pred_adr=adr.predict(X_test)

            precision_adr=adr.score(X_train,y_train)

            print_precision_model(precision_adr,adr,i)

        if i==4: #BOSQUES ALEATORIOS REGRESION

            bar=RandomForestRegressor(n_estimators=300,max_depth=8)

            bar.fit(X_train,y_train)

            Y_pred_bar=bar.predict(X_test)

            precision_bar=bar.score(X_train,y_train)

            print_precision_model(precision_bar,bar,i)

    return
Y_pred_multiple,precision_mul,Y_pred_svr,precision_svr,Y_pred_adr,precision_adr,Y_pred_ba
r,precision_bar

def analisis_algoritmos_clasificacion (X,y):

    X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)

    for i in range (5,11):

        if i==5: #MODELO REGRESIÓN LOGÍSTICA

            escalar=StandardScaler()

            X_train_esc=escalar.fit_transform(X_train)

            X_test_esc=escalar.transform(X_test)

            r_log=LogisticRegression(class_weight='balanced')

            r_log.fit(X_train_esc,y_train)

            Y_pred_logist=r_log.predict(X_test_esc)

            precision_r_log=precision_score(y_test, Y_pred_logist)

            print_precision_model(precision_r_log,r_log,i)

            plot_confusion_matrix(y_test,Y_pred_logist)

        if i==6: #K VECINOS MÁS CERCANOS

            K_neig=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)      #      distancia
Euclidiana

            K_neig.fit(X_train,y_train)

            y_pred_neig=K_neig.predict(X_test)
```

```
    precision_neig=precision_score(y_test, y_pred_neig)
    print_precision_model(precision_neig,K_neig,i)
    plot_confusion_matrix(y_test,y_pred_neig)
if i==7: #MÁQUINA VECTORES SOPORTE CLASIFICACIÓN
    svr_clas=SVC(kernel='linear', class_weight='balanced')
    svr_clas.fit(X_train,y_train)
    y_pred_svr_class=svr_clas.predict(X_test)
    precision_svr_class=precision_score(y_test, y_pred_svr_class)
    print_precision_model(precision_svr_class,svr_clas,i)
    plot_confusion_matrix(y_test,y_pred_svr_class)
if i==8: #NAIVE BAYES
    naive_bayes=GaussianNB()
    naive_bayes.fit(X_train,y_train)
    y_pred_naive=naive_bayes.predict(X_test)
    precision_naive=precision_score(y_test, y_pred_naive)
    print_precision_model(precision_naive,naive_bayes,i)
    plot_confusion_matrix(y_test,y_pred_naive)
if i==9: #ÁRBOLES DE DECISIÓN CLASIFICACIÓN
    adc=DecisionTreeClassifier(criterion='entropy')
    adc.fit(X_train,y_train)
    y_pred_adc=adc.predict(X_test)
    precision_adc=precision_score(y_test, y_pred_adc)
    print_precision_model(precision_adc,adc,i)
    plot_confusion_matrix(y_test,y_pred_adc)
if i==10: #BOSQUES ALEATORIOS CLASIFICACIÓN
    bac=RandomForestClassifier(n_estimators=10,criterion='entropy')
    bac.fit(X_train,y_train)
    y_pred_bac=bac.predict(X_test)
    precision_bac=precision_score(y_test, y_pred_bac)
    print_precision_model(precision_bac,bac,i)
```

```
        plot_confusion_matrix(y_test,y_pred_bac)

    return
Y_pred_logist,precision_r_log,y_pred_neig,precision_neig,y_pred_svr_class,precision_svr_clas
s,y_pred_naive,precision_naive,y_pred_adc,precision_adc,y_pred_bac,precision_bac
```

9.4. Código Entrenamientos

```
print('---------------------------------------------------------------------------------------------')

print()

print('############################ENTRENAMIENTO 1
#####################################################')

print()

print('---------------------------------------------------------------------------------------------')

print()

# Entrenamiento características principales

X=df_param.copy()

X.drop(columns=['Fecha','Riego_act'],inplace=True)

y=df_param['Riego_act']

(Y_pred_multiple1,precision_mul1,Y_pred_svr1,precision_svr1,Y_pred_adr1,precision_adr1,Y_
pred_bar1,precision_bar1)= analisis_algoritmos_regresion (X,y)

(Y_pred_logist1,precision_r_log1,y_pred_neig1,precision_neig1,y_pred_svr_class1,precision_s
vr_class1,y_pred_naive1,precision_naive1,y_pred_adc1,precision_adc1,y_pred_bac1,precision
_bac1)=analisis_algoritmos_clasificacion(X,y)

print('---------------------------------------------------------------------------------------------')

print()

print('############################ ENTRENAMIENTO 2
#####################################################')

print()

print('---------------------------------------------------------------------------------------------')

print()

# Entrenamiento características principales incluyendo la hora

X=df_param.copy()

X=format_fecha (df,X)

X.drop(columns=['Fecha','Riego_act','WEEK_DAY','MONTH_DAY','MONTH','YEAR'],inplace=Tr
ue)
```

```
y=df_param['Riego_act']

(Y_pred_multiple2,precision_mul2,Y_pred_svr2,precision_svr2,Y_pred_adr2,precision_adr2,Y_
pred_bar2,precision_bar2)=analisis_algoritmos_regresion (X,y)

(Y_pred_logist2,precision_r_log2,y_pred_neig2,precision_neig2,y_pred_svr_class2,precision_s
vr_class2,y_pred_naive2,precision_naive2,y_pred_adc2,precision_adc2,y_pred_bac2,precision
_bac2)=analisis_algoritmos_clasificacion(X,y)

print('----------------------------------------------------------------------------------------')

print()

print('############################ ENTRENAMIENTO 3
#########################################################')

print()

print('----------------------------------------------------------------------------------------')

print()

# Entrenamiento características principales incluyendo la hora y día de la semana

X=df_param.copy()

X=format_fecha (df,X)

X.drop(columns=['Fecha','Riego_act','Temperatura_RTU','Cobertura_RTU','Bateria_RTU','Rad_
med','Vv_med','MONTH_DAY','MONTH','YEAR'],inplace=True)

y=df_param['Riego_act']

(Y_pred_multiple3,precision_mul3,Y_pred_svr3,precision_svr3,Y_pred_adr3,precision_adr3,Y_
pred_bar3,precision_bar3)=analisis_algoritmos_regresion (X,y)

(Y_pred_logist3,precision_r_log3,y_pred_neig3,precision_neig3,y_pred_svr_class3,precision_s
vr_class3,y_pred_naive3,precision_naive3,y_pred_adc3,precision_adc3,y_pred_bac3,precision
_bac3)=analisis_algoritmos_clasificacion(X,y)

print('----------------------------------------------------------------------------------------')

print()

print('############################ ENTRENAMIENTO 4
#########################################################')

print()

print('----------------------------------------------------------------------------------------')

print()

# Entrenamiento con acumulado y día

df_aux=df[['Fecha','Caudal']]

df_acumulado=dataframe_acumulado (df,df_aux)
```

```
X=df_acumulado.copy()

X.drop(columns=['ACUMULADO'],inplace=True)

y=df_acumulado['ACUMULADO']

(Y_pred_multiple4,precision_mul4,Y_pred_svr4,precision_svr4,Y_pred_adr4,precision_adr4,Y_
pred_bar4,precision_bar4)=analisis_algoritmos_regresion (X,y)

df_precision=pd.DataFrame(columns=('REGRESIÓN LINEAL MULTIPLE','VECTORES DE
SOPORTE REGRESIÓN', 'ARBOL DE DECISION REGRESION', 'BOSQUES ALEATORIOS
REGRESION','REGRESIÓN LOGÍSTICA','K VECINOS','MÁQUINA VECTORES SOPORTE
CLASIFICACIÓN','NAIVE BAYES','ÁRBOLES DE DECISIÓN CLASIFICACIÓN','BOSQUES
ALEATORIOS CLASIFICACIÓN'))

df_precision.loc[0]=[precision_mul1,precision_svr1,precision_adr1,precision_bar1,precision_r_l
og1,precision_neig1,precision_svr_class1,precision_naive1,precision_adc1,precision_bac1]

df_precision.loc[1]=[precision_mul2,precision_svr2,precision_adr2,precision_bar2,precision_r_l
og2,precision_neig2,precision_svr_class2,precision_naive2,precision_adc2,precision_bac2]

df_precision.loc[2]=[precision_mul3,precision_svr3,precision_adr3,precision_bar3,precision_r_l
og3,precision_neig3,precision_svr_class3,precision_naive3,precision_adc3,precision_bac3]

df_precision.loc[3]=[precision_mul4,precision_svr4,precision_adr4,precision_bar4,'N/A','N/A','N/
A','N/A','N/A','N/A']

df_precision.to_excel('precision.xlsx')
```