UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Análisis de datos de temperatura y humedad de suelo procedentes de sensores IoT desplegados en un viñedo Documentación Técnica

Presentado por Gabriel Hernández Vallejo
en Universidad de Burgos — 28 de mayo de 2023
Tutores: Rubén Ruiz González, Alejandro Merino Gómez

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto

A.1. Introducción

En la sección se introduce la fase de planificación del proyecto, tanto la planificación temporal, como la viabilidad económica y legal.

A.2. Planificación temporal

La planificación del desarrollo del proyecto se realizó siguiendo la metodología ágil *Scrum*, aunque no en su definición completa, puesto que al ser una única persona en el desarrollo, no permite realizar todos los procedimientos recogidos dentro de los manuales de gestión de proyectos con metodologías ágiles [3], como las reuniones diarias "daily" para recoger diferentes cuestiones como cúales son los problemas que ralentizan el proceso de desarrollo del proyecto.

Por otro lado, la herramienta software que apoyaba su aplicación (ZenHub [4]) dejó de estar disponible de forma abierta, de forma que no pudo seguirse con el modelo *Canvas* que aplicaba la metodología recogiendo en un tablero con varias columnas, según fuera su estado, las tareas por cada uno de los *sprints*.

Aun así, se ha continuado aplicando la filosofía ágil en líneas generales:

- Se continuó con el desarrollo iterativo incremental mediante los *sprints*.

- La duración media de estas iteraciones fueron de dos semanas al comienzo del proyecto y durante la mayor parte de su realización, reduciendo el tiempo al final de este.
- En la terminación de los *sprints* se realizaba un incremento, manteniendo reuniones con los tutores para la planificación de la iteración entrante y la revisión de posibles errores en el desarrollo.
- Las tareas surgidas de la reunión se creaba, estimaban y priorizaban, en un inicio añadiéndolas al tablero *Canvas*.

La estimación fue realizada empleando los *story points* disponibles en ZenHub, de forma que estos iban de las tareas más sencillas de implementar y rápidas, a las más complejas y que requerían de mayor tiempo de desarrollo.

Sprint 0 (18/01/2023 - 31/01/2023)

En este *sprint* se presentó el proyecto, indicándose los primeros objetivos, lo que se enmarca en la fase de comprensión del negocio. Las primeras tareas se correspondían con la fase de comprensión de datos y a su vez la preparación de los mismos, puesto que el conjunto proporcionado contaba con diferentes casuísticas de las que en un comienzo se tenía constancia.

Entre los problemas encontrados previos al análisis en profundidad de los datos se encontraban lecturas incorrectas (valores erróneos) debido a problemas de *overflow* de variables en los controladores de cada sensor, de forma que cuando la batería bajaba de los 3.1V, las lecturas en el conjunto de datos se trataban de errores en su mayoría.

Además, en ciertos periodos de tiempo los sensores habían estado desconectados, por lo que se produjeron lecturas nulas (valores faltantes) que necesitaban ser tratadas.

Por otro lado, las lecturas en el pluviómetro instalado en las dependencias del viñedo en algunas de las muestras arrojaban valores inverosímiles debido a la forma en la que este se encontraba instalado, de manera que el balancín empleado para medir las diferencias de precipitaciones entre instantes se activaba artificialmente.

Durante el desarrollo de la iteración se comenzó con el tratamiento de los datos nulos, tomando la decisión de eliminar las entradas, pues se contaba con un número elevado de muestras y realizar la recuperación de estas no parecía viable debido a la naturaleza de los datos. Por otro lado, para los

problemas relacionados con el pluviómetro se solicitaron claves *API* para poder acceder a los registros de estaciones meteorológicas cercanas a la zona y poder comprobar los registros.

Por otro lado, se comenzó a realizar las gráficas de los datos para intentar encontrar correlaciones en los datos y poder hallar diferentes errores en el conjunto de datos.

Sprint 1 (31/01/2023 - 14/02/2023)

En este *sprint* se presentaron los avances de la primera iteración y se acordó la continuación del proceso de comprensión de datos y preparación de los mismos.

Los esfuerzos se centraron entonces en modificar las gráficas de visualización de datos para continuar con la inspección visual de los sensores y comenzar con los datos del pluviómetro. Con estas inspecciones se consigue encontrar anomalías en ciertos sensores como en el caso del segundo y cuarto.

Sprint 2 (14/02/2023 - 28/02/2023)

En la reunión de la iteración se propusieron diferentes objetivos:

- En el sensor 2 eliminar con variación excesiva en temperatura, corrigiendo los datos ruidosos.
- En el sensor 3 se encontró que el salto de humedad no era genuino.
- En el sensor 4 intentar realizar una media móvil para arreglar variables como la temperatura al emplear variaciones diarias en lugar de muestreos cada 5 minutos.

Durante el desarrollo del *sprint* se continuó con la selección de datos, encontrando que el efecto de la lluvia provocaba ciertos cambios bruscos en los datos como en el caso del sensor 5, pero estando algunos aparentemente no relacionados con este fenómeno. Se comenzó, por otro lado con la selección y limpieza de los datos mediante una columna adicional de validez en los ficheros correspondientes.

Además, se implementó la detección de *Outliers* mediante el rango intercuartílico, para eliminar el posible ruido del conjunto de datos de los sensores.

Sprint 3 (28/02/2023 - 14/03/2023)

En la reunión de la iteración se acordó realizar la recuperación de los datos del sensor 8 hasta la variabilidad excesiva de los valores de las variables observadas, así como el intento de mejorar la calidad del conjunto de datos en otros sensores realizando procesos similares. Por otro lado, se propuso emplear WeatherBit como *API* meteorológica para eliminar los datos que más distaran de las lecturas del pluviómetro desplegado.

Durante el desarrollo del *sprint* se modificaron los umbrales empleados en la detección y manejo de valores extremos, para eliminar la mayoría cantidad de ruido posible en todos los sensores. Por otro lado, se fueron finalizando las selecciones de datos de los sensores, de forma que se realizó la recuperación de los datos en sensores cuyas variabilidades se debían a factores externos (fueron sacados de sus posiciones originales).

En lo referente del sensor 8 se llegó a la conclusión de que buena parte de los datos eran irrecuperables debido a las variabilidades aleatorias con las que parecía contar.

Por otro lado, se comenzó con la limpieza de los datos de los sensores empleando los datos de varias *API*, en primer lugar se trató de emplear la mencionada anteriormente, siendo el *endpoint* ideal de licencia de pago. Probando con OpenWeatherMap se presentó el caso al soporte, ampliando la licencia de usuario para poder acceder a l utilidad requerida, sin embargo no se permitía retrotraerse en más de un año en las lecturas registradas.

Finalmente se decide emplear los servicios de la AEMET, debido principalmente a que permitía obtener los datos de hacía más de un año, sin embargo, contaba con inconvenientes, y es que no permitía obtener datos por hora, sino por día y la estación meteorológica más cercana se encontraba a unos 15 Kilómetros del despliegue.

Sprint 4 (14/03/2023 - 28/03/2023)

En la reunión del *sprint* se acuerda comenzar con la fase del modelado a la vista de la aparente correcta selección y limpieza de los datos proporcionados. Se plantea la creación de una matriz de correlación para observar cúales de los atributos seleccionar para la creación del modelo.

Durante el desarrollo de la iteración los esfuerzos se centran en la creación del entorno virtual con las dependencias correspondientes que permitan realizar los modelados, además de dejar el fichero de pre-procesamiento de datos comentado con las explciaciones pertinentes.

Por otro lado, se realizan avances sustanciales en la memoria del proyecto añadiendo información el la introducción, objetivos y técnicas y herramientas, además de realizar una reestructuración de ficheros y una modificación de la forma en la que se realizan las gráficas de los datos.

Se comienza con la creación tentativa del modelo neuronal en busca de conocimiento del dominio, que permita profundizar en los módulos empleados para tal propósito, realizando regresión simple. Se investiga sobre las redes neruonales recurrentes, más concretamente sobre la aplicación de modelos como *GRU* y *LSTM* en problemas similares (predicciones de tiempo atmosférico).

Sprint 5 (28/03/2023 - 11/04/2023)

En la reunión de la iteración se muestran los avances en memoria y en la creación de modelos, por otro lado, se plantea la forma de realizar la regresión empleando Keras.

Durante el *sprint* se modifica la manera en la que se crean los datos de entrada a los modelos, empleando medias diarias para tal propósito y dividiendo los datos en dos conjuntos: entrenamiento y validación. Además, se crean las gráficas de dispersión que permiten comparar visualmente las predicciones y valores reales.

Por otro lado, se continúa con el desarrollo de la memoria, añadiendo conceptos teóricos sobre el proceso de extracción de conocimiento de bases de datos (*KDD*), así como modificando algunos apartados.

Sprint 6 (11/04/2023 - 25/04/2023)

En la planificación se considera entrenar a los modelos neuronales únicamente con valores adecuados, pues hasta ahora pueden existir saltos temporales que pueden afectar a la regresión, realizando medias temporales cada cierto periodo de tiempo para eliminar los posibles problemas de *offset* de los muestreos originales.

En el desarrollo del *sprint*, se modificó la selección en los datos de entrada a los modelos, de forma que se realizaban las medias diarias de los diferentes atributos y se les aplicaba el filtro de Hodrick-Presscott, un filtro de tendencias, que permite obtener las tendencias en una serie temporal, de manera que los modelos sean capaces de generalizar de forma más adecuada.

Por otro lado, se continuó con la memoria, en concreto con los conceptos teóricos relacionados con la fase de modelado.

Sprint 7 (25/04/2023 - 09/05/2023)

En la reunión de la iteración se revisan los cambios realizados, llegando a la consideración de emplear varios dias previos como entrada a los modelos, utilizando una ventana deslizante para seleccionar datos, de manera que se eviten los saltos temporales y estudiar la incorporación de una componente temporal como entrada en la regresión.

De esta forma, en el desarrollo se aplicó la ventana deslizante para la selección de datos de entrada en el modelo, de manera que se pudieran evitar los saltos temporales.

Sprint 8 (09/05/2023 - 16/05/2023)

En la planificación se acordó intentar rebajar los periodos de tiempo, y que se emplearan datos medios por hora en lugar de diarios y probar los hiperparámetros de las redes, intentando variar las neuronas por capa, el número de estas, el ratio de aprendizaje, etc. para observar el comportamiento de los modelos y determinar tanto si continúan generalizando como si es necesario realizar cambios en el proceso de conocimiento de datos y preparación de estos.

De esta manera, se redujo la frecuencia de las medias de los datos, realizando la integración de los datos de los sensores en un único fichero, para poder entrenar los modelos empleando un único conjunto. Esto obligó a cambiar la forma en la que se graficaban los resultados, mostrando las gráficas de dispersión por sensor y atributo.

Por otro lado, se dividió el conjunto de datos en tres subconjuntos diferentes: un conjunto para entrenamiento, otro para validación y el restante para test, permitiendo la parametrización en la división de estos.

Sprint 9 (16/05/2023 - 23/05/2023)

En la reunión de la iteración se acordó tratar de predecir tiempos más largos en lugar de un único instante como realizaban los modelos hasta el momento; previsiblemente el error sería mayor, por lo que se mencionó permitir parametrizar este valor.

Ante el objetivo principal planteado hubo que realizar diferentes modificaciones en la selección de los datos de salida y los parámetros, penalizando el rendimiento general de los modelos.

Se avanzó en la memoria en los conceptos teóricos y se realizaron algunas correcciones.

Sprint 10 (23/05/2023 - 30/05/2023)

En la planificación básicamente se indicó la continuación en los esfuerzos de realización de memoria y anexos. De esta manera, este *sprint* se centró en la finalización de los mismos.

A.3. Estudio de viabilidad

En esta sección se desglosarán la viabilidad económica y legal del proyecto, en cuanto a la primera indicará los costes derivados del desarrollo en un enotorno real. En lo referente la segunda, se presentarán las licencias empleadas en el proyecto y su implicación con librerías de terceros.

Viabilidad económica

En términos de viabilidad económoca es necesario hacer una diferenciación entre los costes y los beneficios que lleva realizar el proyecto.

Costes

Los costes que pueden surgir del proyecto en un entorno empresarial pueden desglosarse en los siguientes:

Costes de personal:

El desarrollo se ha realizado con un única persona empleada a tiempo completo en aproximadamente 4 meses, de forma que se consideran los siguientes costes:

Concepto	**Coste**
Salario mensual neto	1.080€
Retención I.R.P.F(12 %)	185,46€
Seguridad Social(31,1 %)	480,64€
Salario mensual bruto	1.545€
Total 4 meses	6.181,92€

Tabla A.1: Costes de Hardware

El porcentaje mensual de aporte a la Segurida Social se calcula como el 0,2 % al Fondo de Garantía Salarial (FOGASA), más el 0,6 %, el 6,7 % de la prestación por desempleo y el 23,6 % de contingencias comunes [5].

Costes de hardware:

El hardware empleado tiene un tiempo de amortización aproximado de 4 años.

Concepto	**Coste**	**Coste amortizado**
Ordenador portátil	900€	45€
Total	900€	45€

Tabla A.2: Costes de Hardware

Costes de software:

En cuanto a los costes de software, hay ciertos programas o herramientas empleadas durante el desarrollo que requieren de licencia de pago, estos contarán con un tiempo de amortización estimado de 2 años.

Concepto	**Coste**	**Coste amortizado**
Windows 10 Home	148€	18,5€
Total	148€	18,5€

Tabla A.3: Costes de Software

Costes varios:

Al igual que en otras situaciones empresariales, en los desarrollos de software surgen tanto costes inesperados, como fijos. Se reflejarán aquellos costes variados que aparentemente se tratarían de costes fijos.

Concepto	**Coste**
Memoria y anexos	50€
Alquiler de espacio de trabajo	592€
Internet	120€
Total	762€

Tabla A.4: Costes varios

Costes totales:

Los costes del desarrollo del proyecto son los siguientes:

Concepto	**Coste**
Personal	6.181,92€
Hardware	900€
Software	148€
Varios	762€
Total	7.991,92€

Tabla A.5: Costes totales

Beneficios

En cuanto a los beneficios generados, en el *scope* del proyecto no había cabida a un despliegue en los meses de desarrollo de los modelos, por lo que a corto plazo no habría ningún beneficio aparente.

Sin embargo, la forma de obtener los beneficios entraría dentro de una segunda fase, cuando se realice el despliegue del resultados en algún producto.

Viabilidad legal

En esta sección se desglosarán los temas relacionados con las licencias de uso de los productos software.

En primer lugar se analizará la licencia más conveniente en el desarrollo del proyecto, teniendo en cuenta las dependencias empleadas y sus respectivas licencias, puesto que la selección estará limitada por las restricciones de estas principalmente.

Dependencia	**Versión**	**Descripción**	**Licencia**
TensorFlow	2.11.0	Biblioteca de aprendizaje automático	Apache v2.0
Pandas	1.5.3	Biblioteca especializada en manipulación y análisis de datos	BSD
Matplotlib	2.11.0	Biblioteca para generación de gráficos	BSD
IPython Kernel for Jupyter	6.21.3	Biblioteca para manipulación de Python Notebooks	BSD
Statsmodel	0.13.5	Biblioteca con modelos y funciones estadísticas	BSD

Tabla A.6: Licencias de las dependencias utilizadas

De esta forma, se debe seleccionar una licencia compatible con Apache v2.0 y BSD, siendo la primera la más restrictiva de las licencias, que requiere la conservación del aviso de derecho de autor y un descargo de responsabilidad, sin embargo, no es *copyleft* [6].

Por su versatilidad se escoge una licencia MIT, de forma que el se trata de una licencia permisiva de software libre, imponiendo muy poocas restricciones y otorgando una buena compatibilidad con otras licencias.

Apéndice B

Comprensión del negocio

B.1. Introducción

En la fase de comprensión del negocio se pretende analizar los objetivos y requisitos del proyecto.

Estos quedarán bien marcados por las necesidades establecidas en las descripciones del TFG, quedando reflejadas en la sección de introducción de la memoria.

A continuación se resumen tanto los requisitos como los objetivos.

B.2. Objetivos del proyecto

Entre los principales objetivos del proyecto se encuentran la realización de un análisis de los datos proporcionados de forma que se pueda facilitar la comprensión de los datos recogidos mediante representaciones gráficas, buscando, de esta forma correlaciones y para poder encontrar modelos predictivos para anticipar la evolución futura de las diferentes variables en un plazo de tiempo determinado.

B.3. Requisitos del proyecto

Entre los rquisitos del proyecto se encontraría la creación de modelos adecuados a los datos que predigan correctamente las diferentes variables objetivo.

Apéndice C

Comprensión de los datos

C.1. Introducción

En la etapa de comprensión de los datos se creará el conjunto inicial y se comprobará si este es adecuado, para, en caso contrario, poder continuar recopilando.

Los plazos del proyecto impiden realizar esta fase en toda su extensión, puesto que a pesar de ser el proceso de análisis de datos un proceso iterativo, se encuentra sujeto a unos plazos inamovibles.

C.2. Descripción de los datos

Se proporcionan un conjunto de datos compuesto por varios sensores (en concreto 8 sensores IoT) y un pluviómetro desplegados en un viñedo.

Los datos de los sensores proporcionados cuentan con los siguientes atributos:

Atributo	**Descripción**	**Medida**
ts	*Timestamp* de la muestra	ms
fecha	Fecha de la muestra	Fecha en formato yy/mm/dd, hh:mm:ss
batería	Nivel de batería del sensor	V
t_ext	Temperatura ambiental	ºC
h_ext	Humedad ambiental	%
t_C_cal	Temperatura superficial a 20cm.	ºC
h_C_cal	Humedad superficial a 20cm.	%
t_L_cal	Temperatura a profundidad mayor a t_C_cal	ºC
h_L_cal	Humedad superficial a profundiad mayor a h_C_cal	%
h_C	Humedad superficial sin calibrar	Valor relacionado con capacitancias
h_L	Humedad profunda sin calibrar	Valor relacionado con capacitancias

Tabla C.1: Atributos de los datos: sensores

Los datos del pluviómetro cuentan con los siguientes atributos:

Atributo	**Descripción**	**Medida**
ts	*Timestamp* de la muestra	ms
fecha	Fecha de la muestra	Fecha en formato yy/mm/dd, hh:mm:ss
batería	Nivel de batería del sensor	V
pluv	Contador del incremento de activación del balancín	N/A
pluv_delta	Incremento de activación del balancín	N/A
pluv_deltaMM	Precipitaciones registradas desde la última lectura	litros/m^2

Tabla C.2: Atributos de los datos: pluviómetro

C.3. Visualización de los datos

Para dar una idea del estado de los datos se proporcionan unas gráficas de dispersión con las lecturas realizadas en cada uno de los sensores y el pluviómetro. En estas podrán observarse los valores **sin procesar** de cada atributo en función del tiempo.

En el caso de los sensores, los atributos *h_C* y *h_L* se obviarán debido a que se tratan de datos redundantes con las respectivas humedades calibradas. Además, los datos de batería y fecha tampoco se emplearán en la creación de las gráficas, puesto que no aportan información útil para los objetivos del proyecto más allá de posibles errores de muestreo.

Sensor 1

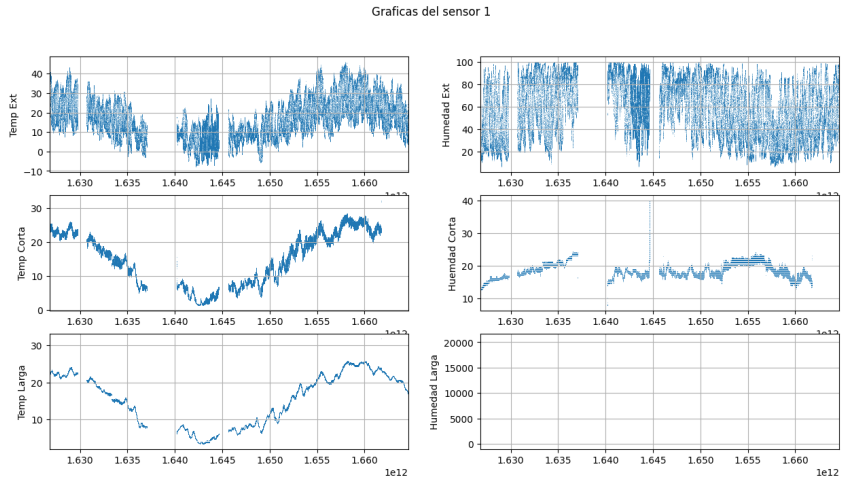

Figura C.1: Datos no procesados: sensor 1

En el primer sensor, como puede observarse en la Figura C.1, se encuentran varias situaciones que a *priori* son sencillas de solucionar, y es que existe una cantidad importante de muestras para las que no contamos con datos, además de existir valores claramente inverosímiles (como en el caso de la humedad más profunda, o el aumento tan repentino de la humedad superficial). A pesar de esto, el resto parecen estar dentro de la normalidad, por lo que el trabajo en el sensor mencionado será uno de los menos tediosos.

Las subidas tan repentinas como las observadas en la imágen posteriormente se demostró que eran debidas al agotamiento de la batería del

respetivo sensor, que poco antes de producirse arrojaban lecturas artificiales para luego realizar lecturas nulas.

Sensor 2

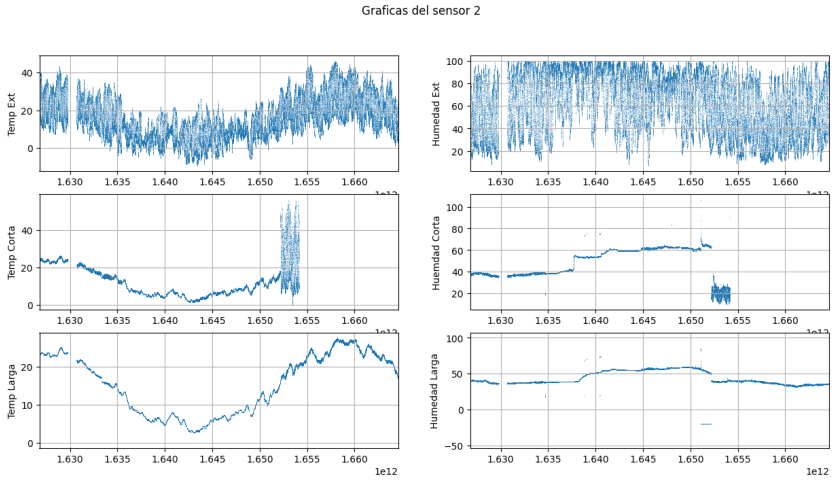

Figura C.2: Datos no procesados: sensor 2

En el segundo caso, además de las lecturas para las que no contemos con valores, el la Figura C.2 se observa cómo llegado un instante las sondas de temperatura y humedad superficiales comienzan a realizar lecturas muy variadas a lo largo del tiempo, para posteriormente dejar de producir datos.

Además, puede observarse la presencia de cierto ruido durante los muestreos previos al suceso mencionado en las sondas de humedad, lo que posiblemente sea debido a las capacitancias de estas.

Sensor 3

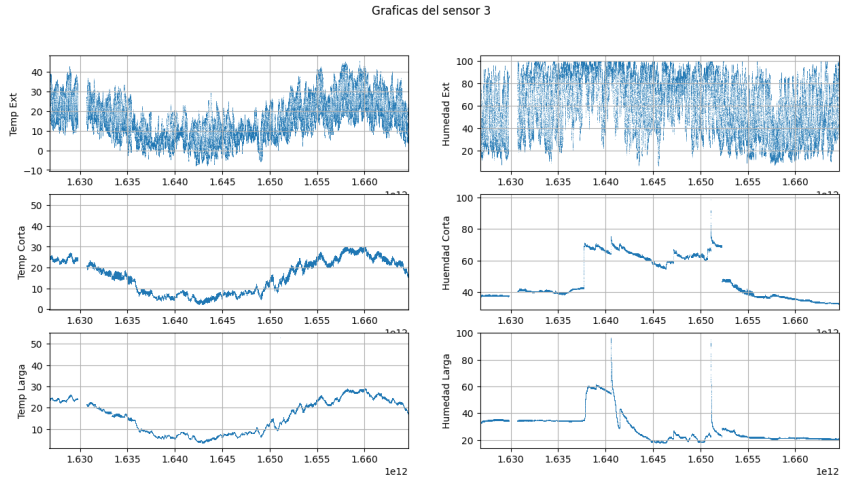

Figura C.3: Datos no procesados: sensor 3

En el este sensor, como puede observarse en la Figura C.3, además de los muestras para los que no tenemos valores (que será la tónica habitual en el conjunto de datos), existen unas subidas remarcables de humedades, con unas bajadas que pueden considerarse aceleradas o repentinas.

La mayor parte de estas son genuinas y debidas a las lluvias contrastadas esos días, sin embargo, la última de las subidas se observa contextualmente errónea, provocando un cambio brusco en ambas humedades del suelo.

Sensor 4

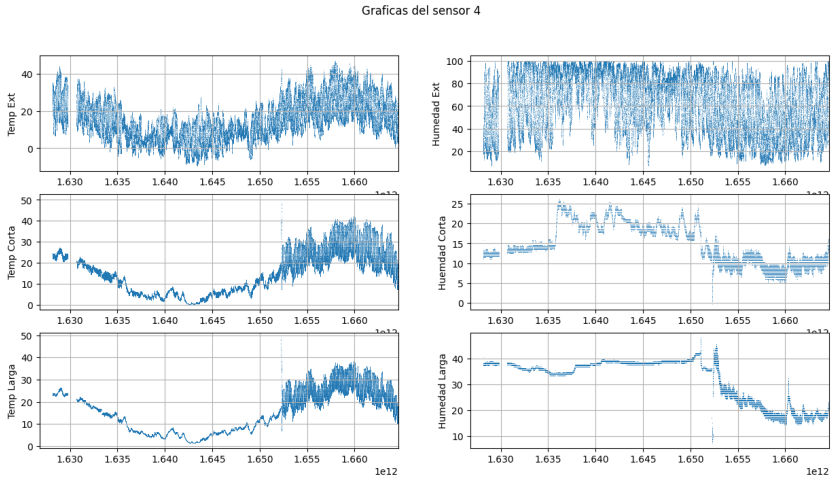

Figura C.4: Datos no procesados: sensor 4

En este caso, pueden observarse en la Figura C.4, diferentes situaciones. La primera de ellas, como en el resto de sensores son los valores faltantes; la segunda es el ruido existente en ciertos periodos de tiempo.

Tras este ruido instantáneo en los datos, puede observarse cómo la variabilidad de las muestras aumenta de forma homogénea en todas y cada una de las sondas, a salvedad de las que recogen datos ambientales.

Las posteriores investigaciones indicaron que de nuevo el sensor había sido desplazado de su situación inicial.

Sensor 5

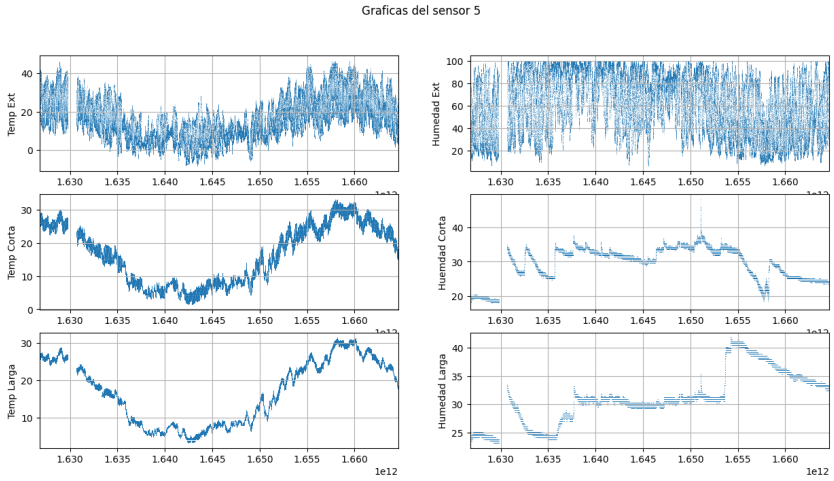

Figura C.5: Datos no procesados: sensor 5

En la Figura C.5 puede observarse diferentes situaciones. Una de ellas es la repetida falta de valores para algunos instántes de tiempo; la segunda es las aparentes subidas repentinas de humedades.

Estas siguen el mismo patrón en todas las situaciones, aumento que puede clasificarse como repentino, y un descenso moderado a lo largo del tiempo. De esta forma, todas se contrastaron como genuinas observando ciertas situaciones como el tiempo atmosférico en los instántes de subida.

Este sensor se trata de uno con los datos más limpios de forma predefinida de todo el conjunto de datos, sino el que en mejor estado se encuentra.

Sensor 6

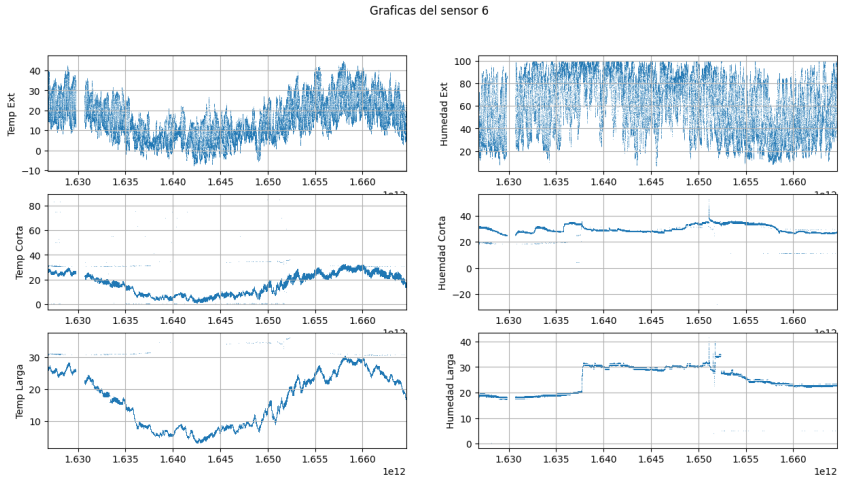

Figura C.6: Datos no procesados: sensor 6

En el conjunto de datos que se puede observar en la Figura C.6, al igual que en los anteriores sensores, pueden observarse valores faltantes, además de otras situaciones.

Una de ella es la presencia generalizada de ruido en el conjunto de ejemplos en las lecturas de las sondas que miden variables del suelo.

Por otro lado, en un instante de tiempo las humedades aumentan de forma repentina, posiblemente por la lluvia, sin embargo en la humedad del suelo a mayor profundidad, hay una variabilidad que se consideró como irrecuperable por su estado.

Sensor 7

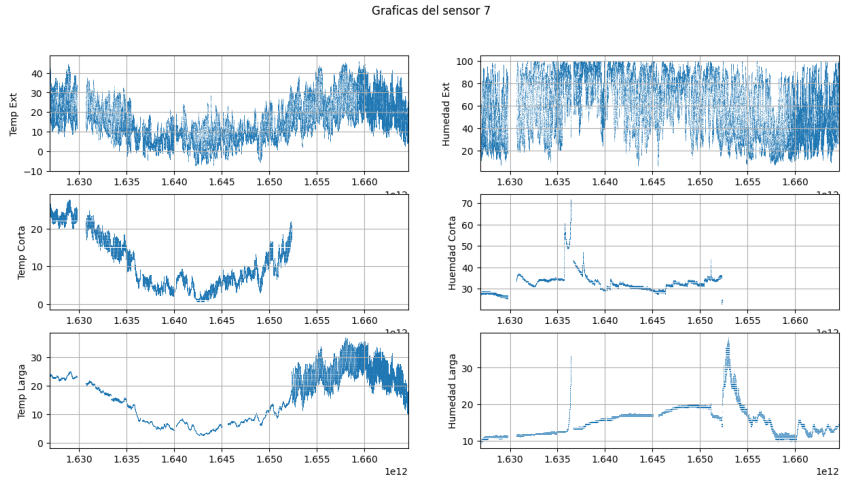

Figura C.7: Datos no procesados: sensor 7

En este caso pueden observarse en la Figura C.7 diferentes situaciones, además de la falta ya mencionada en el resto de sensores de valores en algunas muestras.

En primer lugar, en las sondas de humedad no ambientales se comienza a producir un aumento que podría considerarse como exponencial en el tiempo debido al agotamiento de la batería.

En segundo lugar, como en el sensor 4, se produce un aumento significativo de la variabilidad de la temperatura del suelo, produciéndose, además, un aumento de la sonda de humedad más profunda.

De igual forma que en el sensor mencionado, las labores en los cultivos habían producido un cambio en el emplazamiento original del sensor, lo que producía estas situaciones mencionadas. Con respecto a la falta de los datos de humedad y temperatura más superficiales se comprobará que fue debida a la rotura de las respectivas sondas.

Sensor 8

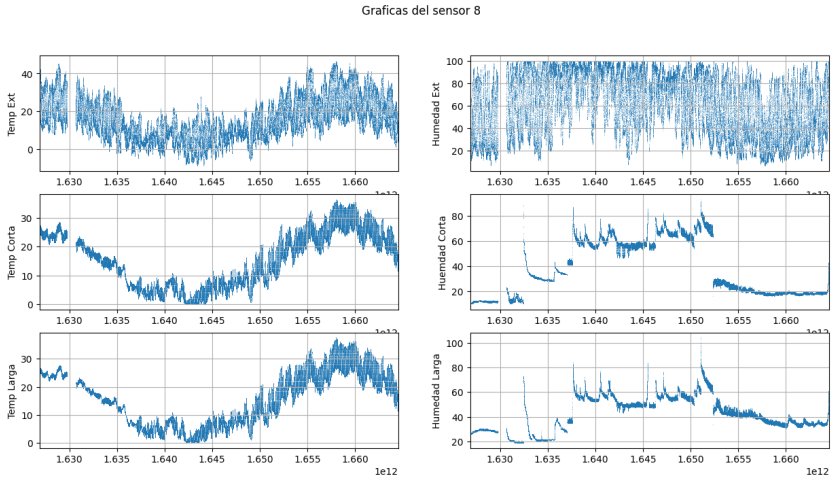

Figura C.8: Datos no procesados: sensor 8

Como puede observarse en la figura C.8, en este caso existen múltiples problemas en el conjunto de datos.

En primer lugar, los ya mencionados valores faltantes en diferentes instantes de tiempo. En segundo lugar, los aumentos de temperatura debidos al agotamiento de la batería y en tercer lugar la alta variabilidad en las temperaturas, que por otro lado, se asemejan en gran medida, lo que indica que de nuevo el sensor había sido desplazado de su emplazamiento original.

Este será uno de los sensores con el conjunto de datos con más problemas de todos los expuestos anteriormente.

Pluviómetro

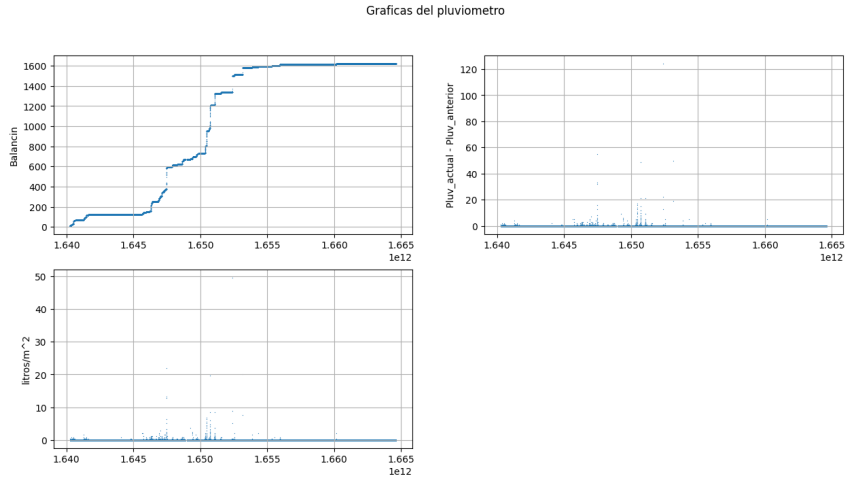

Figura C.9: Datos no procesados: pluviómetro

En el caso del pluviómetro, los datos estarán sujetos a muestras erróneas debido a la forma de fijación del mismo, puesto que con las ráfagas de viento se producían oscilaciones en el poste al que se encontraba anclado, de forma que se realizaban lecturas incorrectas de las precipitaciones.

Apéndice D

Preparación de los datos

D.1. Introducción

En la fase de preparación de los datos se realiza la limpieza de los mismos para que posteriormente puedan ser empleados por los modelos y obtener resultados adecuados para ser utilizados en un posterior despliegue.

Esta fase está compuesta por la detección y el tratamiento de varias casuísticas en los datos, como los valores faltantes, que se tratan de valores de los que no tenemos registros por diferentes motivos y los valores extremos, que son aquellos estadísticamente lejanos al resto, lo que no quiere decir que sean erróneos.

Por otro lado, también se pueden encontrar valores erróneos, que pueden ser estadísticamente correctos, pero son datos que no tienen sentido contextual, en este caso concreto se tratará de subidas o bajadas inverosímiles de temperaturas y/o humedades del suelo, que se encuentran dentro de los valores naturales, pero que no se pueden producir a la velocidad a la que tienen lugar. En este caso, se empleará una exploración visual para realizar su tratamiento.

Cabe destacar que a todos los conjuntos de datos se les ha aplicado un filtro de tendencias (en este caso el filtro de Hodrick-Presscot [7]), puesto que no existe un interés específico en los datos concretos, sino en las posibles tendencias de las diferentes variables, que nos permitan obtener una predicción adecuada en cada circunstancia. De esta manera, se ha establecido la tendencia diaria de los datos.

D.2. Valores faltantes

En cuanto a los valores faltantes, la decisión fue elimiar las muestras que tuvieran indeterminados en alguna de sus columnas, puesto que la recuperación empleando técnicas como la imputación de valores era prácticamente inviable y por otro lado, se contaban con ejemplos suficientes como para poder suprimir parte de estos.

D.3. Valores extremos

En cuanto a la detección y el tratamiento de los valores faltantes, se empleó el rango intercuartílico para realizar la primera de las tareas, sustituyendo las variables del atributo concreto de los ejemplos detectados con la mediana diaria.

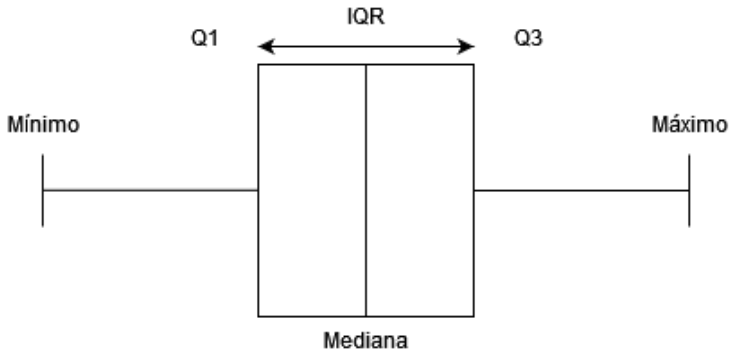

Figura D.1: Representación gráfica del método del rango intercuartílico

De esta manera se eliminaba el ruido presente en alguno de los sensores, a la par que no se producían cambios bruscos en el conjunto de datos gracias a emplear la mediana del grupo.

D.4. Valores erróneos

En lo referente a los valores erróneos se trata de una de las tareas más arduas del proyecto, puesto que parte de este se desarrolló de forma visual diréctamente sobre el conjunto de datos de los diferentes sensores y el pluviómetro, siendo, de esta manera, en su mayoría un proceso manual.

Para tal propósito se estableció una columna adicional al conjunto de datos original que indicaba la validez de la muestra concreta.

D.5. Visualización de los datos

En esta sección se explicará el tratamiento realizado en cada uno de los sensores, y, de esta forma, poner en contexto cómo se han resuelto los problemas detectados en la fase anterior.

Sensor 1

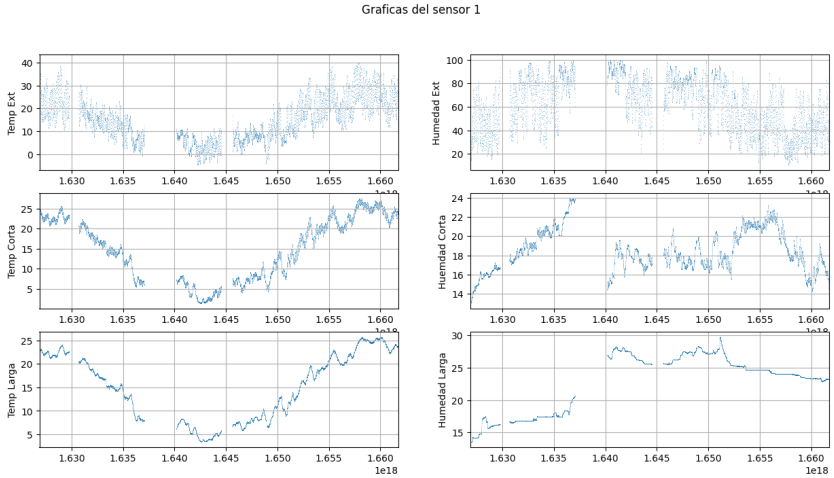

Figura D.2: Datos procesados: sensor 1

Como puede observarse en la Figura D.2, se ha tratado el ruido presente en el conjunto de datos original, sustituyendo los respectivos valores por la mediana diaria.

Por otro lado, se ha determinado la supresión de las muestras que indicaban una variabilidad brusca de las humedades del suelo debidas a el agotamiento de la batería. Esta decisión estuvo motivada por la pérdida moderada de datos, además de la falta de las muestras siguientes.

Sensor 2

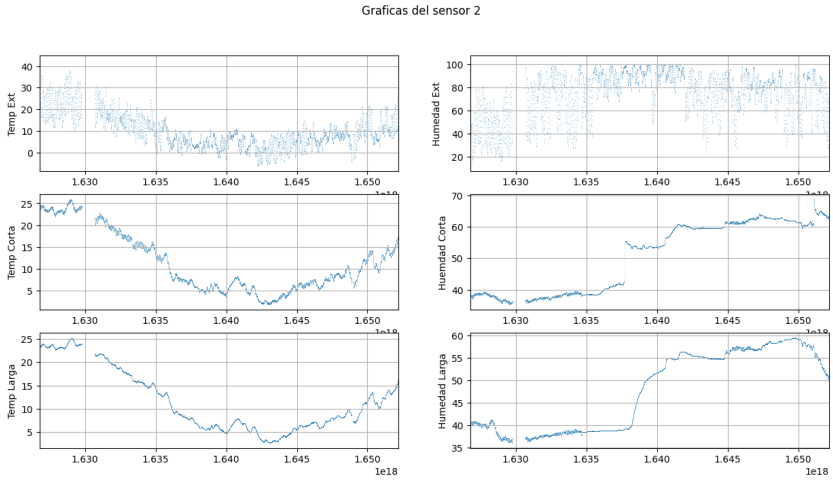

Figura D.3: Datos procesados: sensor 2

En este caso, como puede observarse en la Figura D.3, se aplicó de nuevo la detección y tratamiento de ruido empleando la mediana como valor sustitutorio, además, ante la aparente imposibilidad de obtener una adecuada recuperación de los datos con alta variabilidad, se optó por no utilizarlos.

Sensor 3

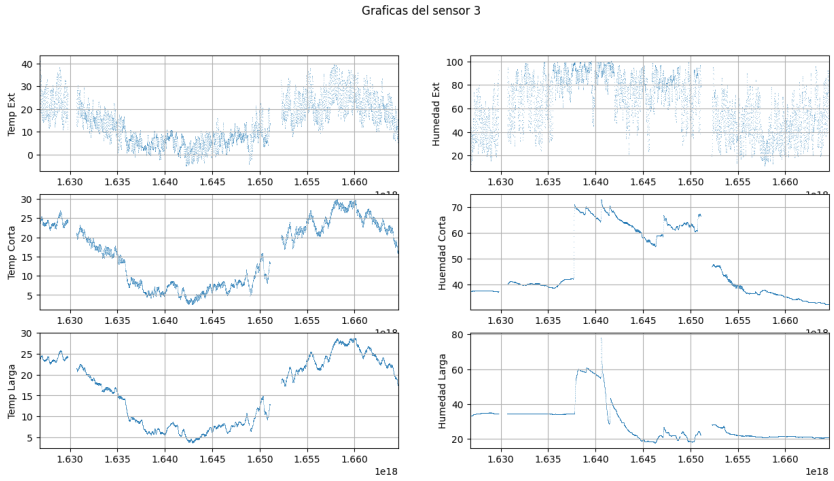

Figura D.4: Datos procesados: sensor 3

Como puede observarse en la Figura D.4, en este caso concreto, el conjunto de datos se modifica, en cuanto a la invalidez de ejemplos se refiere, visiblemente menos que los sensores anteriores y los siguientes.

En este caso solo fue necesario retirar del conjunto de datos una porción moderada de las muestras totales que había resultado del agotamiento de la batería del sensor.

Sensor 4

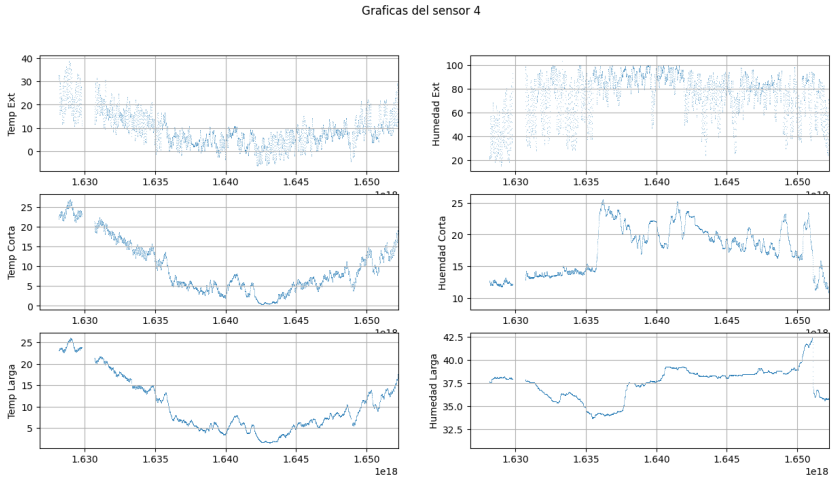

Figura D.5: Datos procesados: sensor 4

En este sensor se optó, como refleja la Figura D.5, por la eliminación de los datos a partir del comienzo de las lecturas que ya se conocían que eran incorrectas, lo que dejaba un conjunto de datos preparado para ser empleado por los modelos.

Sensor 5

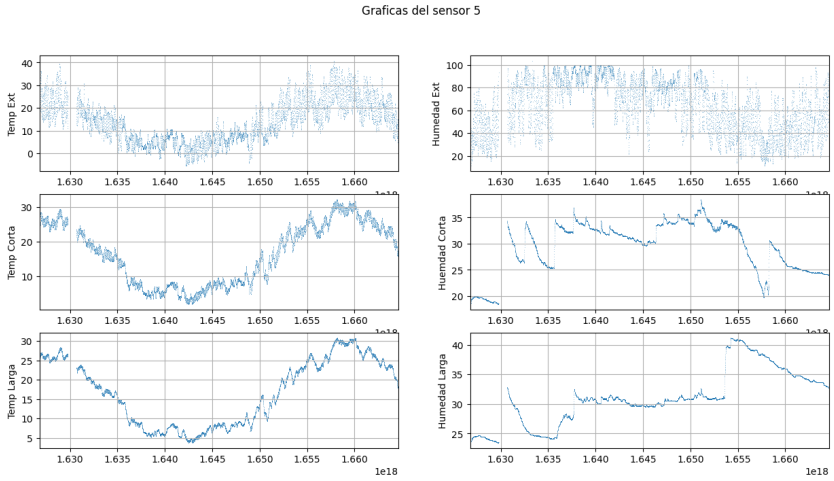

Figura D.6: Datos procesados: sensor 5

En este caso, a diferencia del resto de sensores, no fue necesario una modificación tan exhaustiva del conjunto de datos, puesto que los únicos tratamientos fueron la detección de *outliers* o valores extremos y la eliminación de los ejemplos con valores faltantes.

Sensor 6

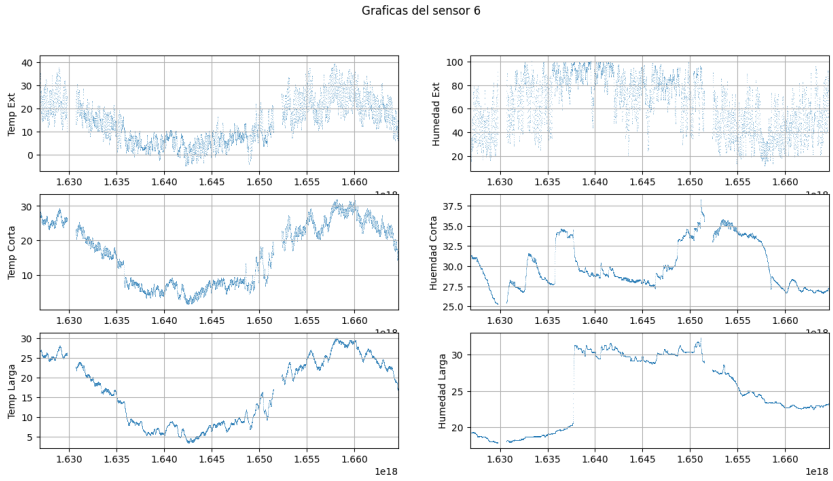

Figura D.7: Datos procesados: sensor 6

En el caso del sensor 6, como puede apreciarse en la Figura D.7, se ha tratado y eliminado el ruido presente mediante la detección de valores extremos. Además, la anomalía de la humedad ha sido eliminada al tratarse de una situación que no podía corregirse, por ejemplo, con los métodos mencionados.

Sensor 7

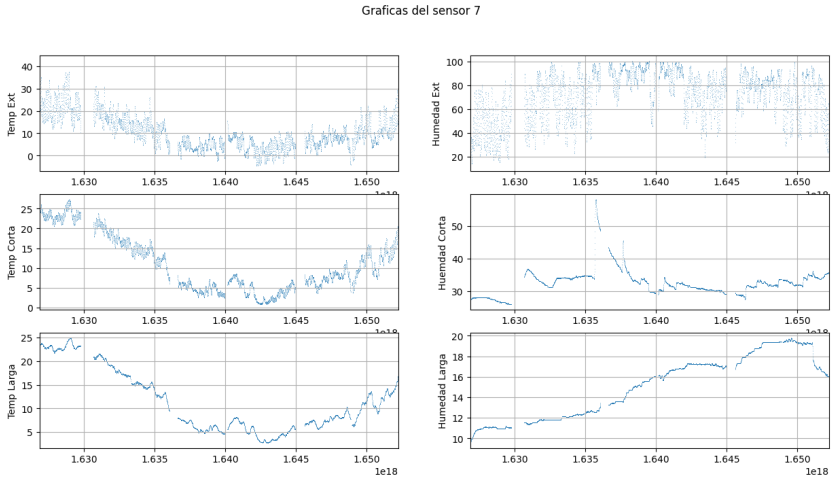

Figura D.8: Datos procesados: sensor 7

En este caso existían dos problemas principales en el conjunto de datos original: las sondas de temperatura y humedad comenzaban a realizar lecturas con variaciones muy elevadas para su profundidad en cortos periodos de tiempo y a la par existían atributos que no contaban con valores, además, al igual que en otros casos, el agotamiento de la batería producía lecturas incorrectas en las humedades.

Ante estos problemas las soluciones tomadas fueron las de eliminar los ejemplos afectados. En el primer caso porque se trataban de periodos de tiempo moderados y en el segundo porque se había producido el desplazamiento del sensor de su posición original.

Sensor 8

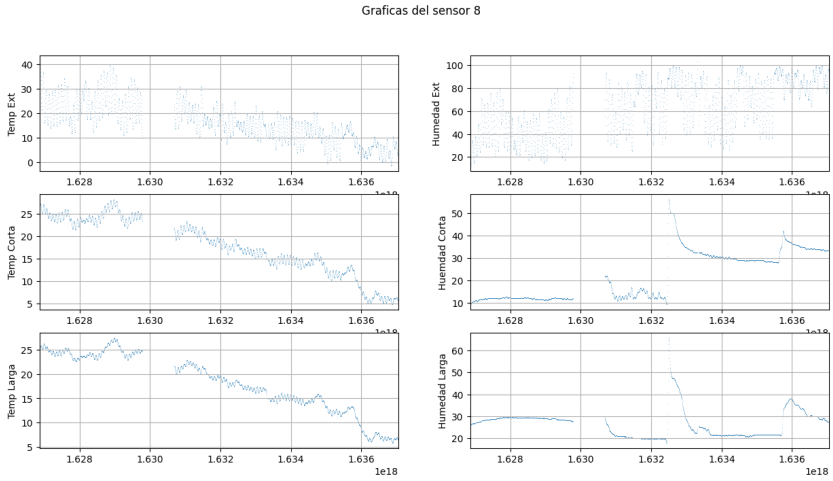

Figura D.9: Datos procesados: sensor 8

Como puede observarse en la Figura D.9, en este caso no ha sido posible recuperar los datos de gran parte del periodo muestreado. Los aumentos de humedad mostrados en la figura indicada están contrastados como provocados por las altas precipitaciones.

Hablándo en términos de sesgos, es posible que al realizar la recuperación de estos valores en los sensores mencionados, puedan haberse sesgado en cierta medida los conjuntos de datos, sin embargo, los datos no dejarán de estar más cercanos a los valores reales y, por tanto, contener un error más reducido que las muestras originales con esta casuística.

Apéndice E

Modelado

E.1. Introducción

En la fase del modelado, como su nombre indica, se crearán los modelos, de forma que al ser este proceso (*KDD*) iterativo, se podrá unir con la etapa anterior, pues existen herramientas que aunan estos procedimientos.

En este caso, se toman por separado, teniendo en cuenta que los resultados en una fase afecta de forma directa al resto.

Se estableció el desarrollo de tres redes neuronales diferentes, *MLP*, *LSTM* y *GRU*, tratándose las dos últimas de modelos recurrentes.

En esta sección se explicará la transformación del conjunto de datos para poder realizar el entrenamiento de los modelos y las respectivas predicciones, así como las diferentes características de cada uno de los modelos.

E.2. Transformación del conjunto de datos

El primer problema que es necesario afrontar es que los modelos deberán aceptar varias entradas de ejemplares, es decir, se necesitarán n muestras sucesivas para realizar las predicciones.

Por otro lado, también deben ser capaces de realizar m predicciones, de forma que se deben crear modelos con n muestras de entrada y m predicciones diferentes.

El segundo problema deriva de la falta de datos en instantes concretos por el procesamiento realizado previamente, tanto por la supresión de muestras con valores erróneos, como la de ejemplares con valores faltantes.

Por tanto, es necesario recorrer los datos en busca de los saltos temporales, para obtener los subconjuntos completos, y de esta manera, asegurar que dentro de estos estas situaciones no se producen.

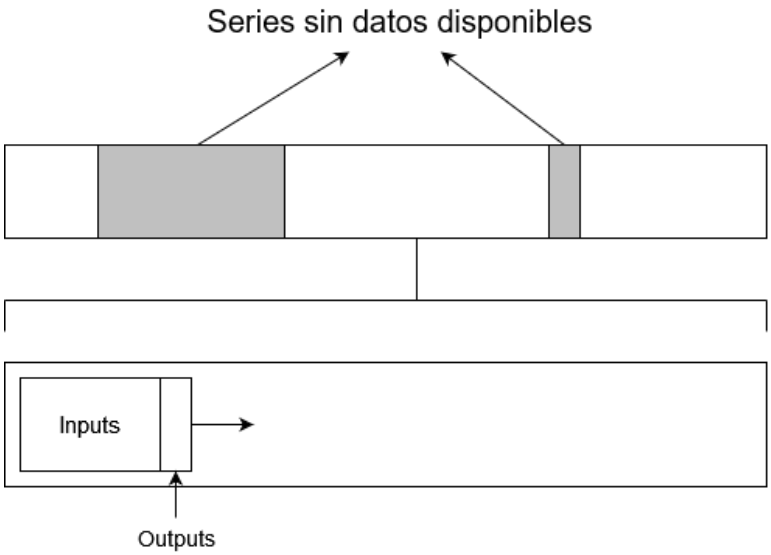

Figura E.1: Selección de la serie de tiempo

En la Fiugura E.1 puede observarse el proceso esquematizado. De esta manera, cada uno de los subconjuntos que no tienen saltos, se recorrerá mediente ventanas deslizantes formando los datos de entrada (*inputs*) a los modelos y las salidas (*targets*).

Cabe destacar que tanto los subconjuntos como las ventanas deberán tener datos suficientes, es decir, $n + m$ muestras.

E.3. GRU

En el caso de *GRU*, las celdas de las que se encuentra compuesto el modelo emplearán como funciones de activación la tangente hiperbólica y funciones sigmoides, es decir, se emplearán los modelos por defecto como puede observars en la Figura E.2.

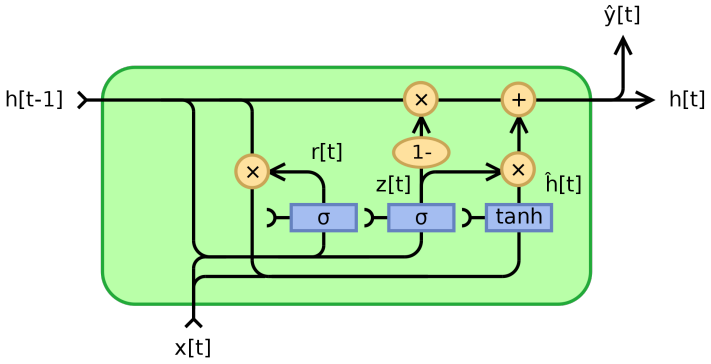

Figura E.2: Representación celda GRU. Extraido de [1]

E.4. LSTM

Al igual que en *GRU*, *LSTM* empleará funciones sigmoides y la tangente hiperbólica como funciones de activación eb las celdas (Figura E.3), empleando, de igual manera, modelos por defecto.

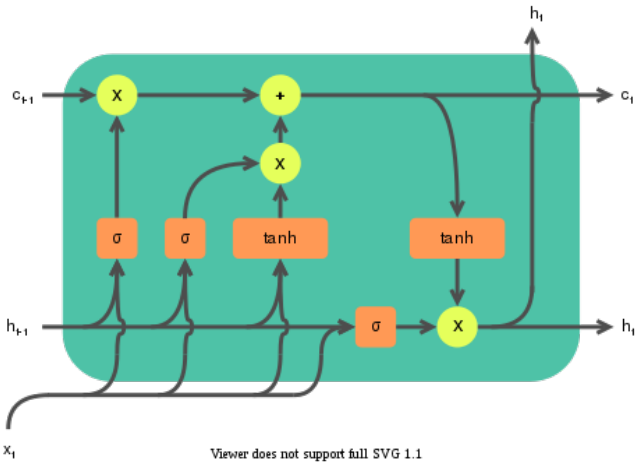

Figura E.3: Representación celda LSTM. Extraido de [2]

A diferencia de *GRU* en cada celda cuenta con 3 puertas diferentes, la puerta de olvido (*Forget Gate*), de entrada (*Input Gate*), de salida (*Output Gate*) y un estado oculto (*Cell Satate*), mientras que el modelo mencionado únicamente cuenta con dos (*Reset Gate* y *Update Gate*). De esta forma, en los modelos *LSTM*, se necesitarán realizar más operaciones por celda que en *GRU*, de manera que el tiempo de entrenamiento será mayor.

E.5. MLP

El perceptrón multicapa, a diferencia de los otros dos modelos, no acepta la entrada de secuencias variables, por lo que se establecerá una neurona de entrada por cada uno de los atributos de las muestras, es decir, la red contendrá $a * n$ neuronas de entrada, siendo a el número de atributos de las muestras y n el número de muestras de entrada.

De forma similar, contará con $a * m$ número de neuronas en la capa de salida, siendo m el número de muestras de salida.

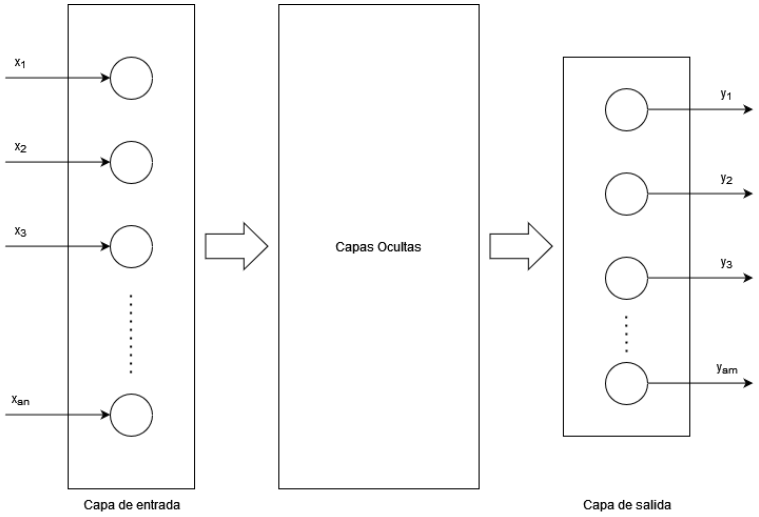

Figura E.4: Diagrama de entradas y salidas MLP

Se establece una función *ReLU* (Figura E.5) como función de activación en las neuronas del modelo.

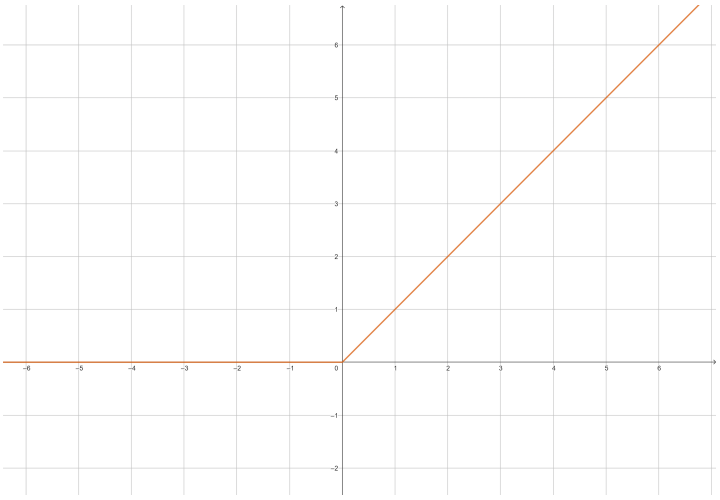

Figura E.5: Función ReLU

A pesar de que el modelo no acepta entradas secuenciales y que por tanto requiere tantas neuronas de entrada como número de valores hay en la secuencia de entrada, el tiempo de entrenamiento de estos modelos será menor generalmente que el de los modelos recurrentes, puesto que se realizan menos operaciones por término medio, además de necesitar de menos memoria para almacenar los pesos sinápticos.

Apéndice F

Evaluación

F.1. Introducción

En esta fase se estima el rendimiento de los diferentes modelos, y, en su caso, se reconsideran los objetivos del proyecto, de manera que si estos son poco efectivos, se vuelve a la primera fase.

En todos los modelos se ha empleado validación cruzada externa y, además, se realizarán diferentes ejecuciones con diversos parámetros para conocer el rendimiento de estos, a la par que poder eliminar en cierta medida el efecto del azar en los modelos neuronales resultantes.

Para poder validarlos se realizarán pruebas variando el números de capas ocultas, así como el número de celdas en cada una de estas y el ratio de aprendizaje. De esta manera, se realizarán pruebas con 1, 2 y 3 capas ocultas en cada uno de los modelos, así como 32, 16 y 8 celdas/neuronas y 0.1, 0.01 y 0.001 de ratio de aprendizaje.

Por otro lado, se ha establecido el conjunto de datos de entrenamiento al 75 % de los disponibles por sensor, 15 % para el conjunto de validación siendo el restante para test.

Con respecto al número de muestras de entrada se emplerarán 6 horas previas para realizar una predicción a una hora vista, empleando un tamaño de bloque de 256 ejemplos (los modelos no se entranan con todos los datos en todas las iteraciones, sino que se emplean un subconjunto).

Por otro lado, se establece un planificador de la tasa de aprendizaje, de forma que en la época 50 disminuye a razón de $lr * e^{-0,1}$, para de esta manera, centrar los esfuerzos en la región explorada.

Para cada una de las pruebas se realizarán 5 ejecuciones diferentes con 100 iteraciones cada una, tomando como medida comparativa el error cuadrático medio de validación.

F.2. Resultados de GRU

En la Tabla F.1 se presentan los errores cuadráticos medios que se obtendrán por término medio con el modelo.

Capas	**Tamaño capa**	**Learning rate**	**MSE GRU**
1	8	0.001	1.72183E-05
1	8	0.01	3.37952E-06
1	8	0.1	0.00551
1	16	0.001	9.1907E-06
1	16	0.01	9.55771E-07
1	16	0.1	0.00474
1	32	0.001	4.50204E-06
1	**32**	**0.01**	**5.96471E-07**
1	32	0.1	0.00297
2	8	0.001	1.56769E-05
2	8	0.01	1.65277E-06
2	8	0.1	0.00965
2	16	0.001	5.07987E-06
2	**16**	**0.01**	**5.66822E-07**
2	16	0.1	0.01148
2	32	0.001	2.18717E-06
2	**32**	**0.01**	**2.69205E-07**
2	32	0.1	0.01727
3	8	0.001	1.61179E-05
3	8	0.01	1.42658E-06
3	8	0.1	0.00706
3	16	0.001	5.18860E-06
3	**16**	**0.01**	**4.66034E-07**
3	16	0.1	0.02418
3	32	0.001	2.36664E-06
3	32	0.01	8.90246E-07
3	32	0.1	0.00976

Tabla F.1: Evaluación GRU

Como puede observarse, los mejores resultados en términos generales les obtendremos con un *learning rate* de 0.01.

F.3. Resultados de LSTM

Capas	**Tamaño capa**	**Learning rate**	**MSE LSTM**
1	8	0.001	
1	8	0.01	
1	8	0.1	
1	16	0.001	
1	16	0.01	
1	16	0.1	
1	32	0.001	
1	32	0.01	
1	32	0.1	
2	8	0.001	
2	8	0.01	
2	8	0.1	
2	16	0.001	
2	16	0.01	
2	16	0.1	
2	32	0.001	
2	32	0.01	
2	32	0.1	
3	8	0.001	
3	8	0.01	
3	8	0.1	
3	16	0.001	
3	16	0.01	
3	16	0.1	
3	32	0.001	
3	32	0.01	
3	32	0.1	

Tabla F.2: Evaluación LSTM

F.4. Resultados de MLP

Capas	**Tamaño capa**	**Learning rate**	**MSE MLP**
1	8	0.001	
1	8	0.01	
1	8	0.1	
1	16	0.001	
1	16	0.01	
1	16	0.1	
1	32	0.001	
1	32	0.01	
1	32	0.1	
2	8	0.001	
2	8	0.01	
2	8	0.1	
2	16	0.001	
2	16	0.01	
2	16	0.1	
2	32	0.001	
2	32	0.01	
2	32	0.1	
3	8	0.001	
3	8	0.01	
3	8	0.1	
3	16	0.001	
3	16	0.01	
3	16	0.1	
3	32	0.001	
3	32	0.01	
3	32	0.1	

Tabla F.3: Evaluación MLP

Apéndice G

Documentación técnica de programación

G.1. Introducción

En esta sección se incluyen la documentación técnica del programador, incluyendo la estructura de directorios del proyecto, junto con el manual para realizar la correcta instalación y ejecución del mismo.

G.2. Estructura de directorios

El proyecto cuenta con la siguiente estructura de directorios:

- **/data/:** directorio que contiene los diferentes datos del proyecto, tanto procesados, sin procesar y los datos integrados que se emplearán en el modelado.

  /data/raw/: directorio con los datos sin procesar (úncamente con la selección previa de validez).

  /data/processed/: directorio con los datos procesados.

  /data/integrated/: directorio con los datos integrados en un único fichero.

- **/img/graphics/:** directorio con las diferentes gráficas resultado de la ejecución de los scripts de graficado.

- **/scripts/:** directorio con los scripts para la instalación de los entornos virtuales de Python junto con los requerimientos para ejecutar todos los ficheros fuente del proyecto.
- **/src/:** directorio con los diferentes ficheros fuente y variables de entorno y globales.
- **/models/:** directorio con los diferentes modelos obtenidos en el proceso final. Uno subdirectorio para cada diferente modelo neuronal implementado.

G.3. Manual del programador

En esta subsección se explicará cómo realizar una correcta descarga e instalación de los entornos necesarios para llevar a cabo la ejecución del proyecto.

Para descargar todo el contenido es necesario tener instalado en el sistema **Git**. Es posible clonar el repositorio introduciendo en la consola de git: **git clone https://github.com/GabiHV/TFG22-23**

De igual forma, para poder llevar a cabo la ejecución e instalación del resto de las dependencias es necesario tener instalado Python 3.9.13.

Para instalar el intérprete del lenguaje empleado en el proyecto es necesario acudir a la página web oficial de los desarrolladores e instalar el ejecutable de instalación oficial. La instalación puede realizarse en el siguiente enlace [8]. En la fuente mencionada se pueden escoger diferentes formas de instalación. Dependiendo del sistema operativo instalado en la máquina en la que se ejecutará el proyecto se debe seleccionar una u otra y seguir los pasos establecidos.

Durante el desarrollo del proyecto se empleó como entorno de programación Visual Studio Code [9], sin embargo para su ejecución podemos emplear otros entornos como Anaconda Navigator [10]. Se explicará la ejecución con el editor mencionado, puesto que simplifica el trabajo al disponer de scripts que realizan de forma automática la instalación de las dependencias. Los ficheros mencionados se encuentran en el directorio **/scripts/**.

Para ejecutar el script correspondiente al entorno de PowerShell de Windows se necesita establecer la política que permita ejecutarlo. Para ello se debe abrir la terminal mencionada como administradores del sistema e introducir:

```
Set-ExecutionPolicy Unrestricted
```

Tras esto, se puede introducir para iniciar el proceso:

```
./Virtual_env.ps1
```

Para ejecutar el script en el CMD de Windows se introduce:

```
virtual_env.bat
```

De forma similar en Linux Bash:

```
chmod +x virtual_env.sh && ./virtual_env.sh
```

Una vez finalice el proceso de instalación de todas las dependencias se podrá ejecutar los diferentes ficheros fuente de Python Notebook abriendo el proyecto en Visual Studio Code y estableciendo el Kernel de ejecución al entorno configurado. Está definido que el entorno virtual se denomine **.venv**, por lo que será necesario buscar entre los diferentes instalados haciendo click en la parte superior derecha del notebook (en el botón para la selección del intérprete de ejecución).

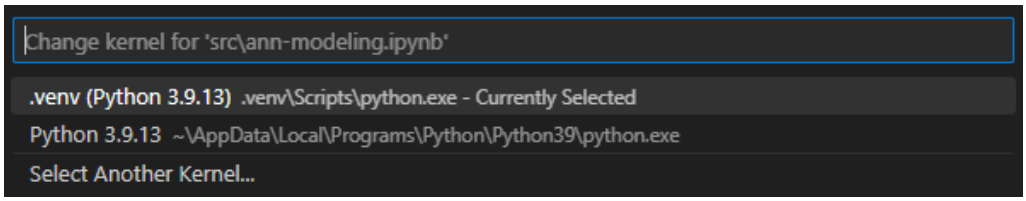

Figura G.1: Búsqueda del entorno virtual

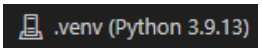

Figura G.2: Selección del entorno virtual

Posteriormente podrá ejecutarse cualquier fichero Pyhton Notebook en el botón de "Execute All".

En lo referente a los ficheros fuente de Python, con el entorno virtual instalado abriendo una consola en el sistema operativo en **/.venv/bin/** en Linux y **/.venv/Scripts/** en Windows, podremos ejecutar la activación del entorno virutal con los scripts incluidos en el directorio (ejecutando

activate.bat y **Activate.ps1** en Windows dependiendo del terminal empleado). Esta operación se realizará automáticamente al realizar la instalación de los módulos de Python incluidos en el fichero de requerimientos, por lo que se puede aprovechar la terminal en ejecución para este propósito.

G.4. Pruebas del sistema

En esta subsección se presentará la forma de realizar las modificaciones en los hiperparámetros de los modelos, de forma que estos puedan variar de acuerdo a los nuevos requerimientos introducidos.

Los modelos contienen los siguientes hiperparámetros:

- **learning_rate:** ratio de aprendizaje empleado en la variación de los pesos en los modelos neuronales.
- **batch_size:** tamaño del conjunto de datos que se emplea en una única iteración en el proceso de aprendizaje.
- **epochs:** cantidad de épocas que se entrenará cada modelo.
- **window_size_inputs:** tamaño de la ventana de datos que se introduce como datos de entrada al modelo (se corresponde con el número de horas previas para realizar X predicciones).
- **window_size_targets:** tamaño de la ventana de datos que se emplean como datos a predecir.
- **train_frac:** fracción del conjunto total de datos que se empleará para entrenar los modelos.
- **val_frac:** fracción del conjunto total de datos que se empleará para validar los modelos, siendo el $1 - train\_frac - val\_frac$ la fracción del conjunto de test.

Por cada uno de los diferentes modelos se proporcionará una gráfica del error de entrenamiento y validación durante el proceso de entrenamiento de la red correspondiente, además de las gráficas comparativas de los valores predichos y los reales por sensor y atributo, así como el error máximo total para todos los sensores en cada uno de estos, para dar una idea de los valores en los que ronda el error en cada una de las variables.

Apéndice H

Documentación de usuario

H.1. Introducción

En esta sección se presentará la forma de cargar los modelos resultantes en un fichero Python para poder ser desplegados en un producto software, así como la forma que deberá tener el tensor de entrada a la red y la que tendrán los datos de salida.

H.2. Requisitos de usuarios

En cuanto a los requisitos del usuario, se deberá tener instalado **Python 3.9.13** [8], junto con la versión 2.11.0 de **TensorFlow** [11], de forma que se puedan cargar los modelos almacenados empleando la función de la *API* **Keras** correspondiente.

H.3. Instalación

En cuanto a la instalación de la versión concreta del intérprete de Python puede realizarse en [8], mientras que para instalar la dependencia de la *API*, se puede introducir el comando:

```
pip3 install tensorflow=2.11.0
```

Para cargar un modelo con la librería mencionada, se debe emplear [12]:

```
from tensorflow import keras
keras.models.load_model('<path_del_modelo>')
```

H.4. Manual del usuario

Para realizar predicciones con el modelo pertinente, las entradas deben tener una estructura concreta que dependerá de cómo se haya entrenado a cada una de las diferentes redes neuronales. Es decir, el tamaño del número de muestras de entrada dependerá de la cantidad de "*backtracking*" que se haya establecido en el entrenamiento.

En este caso se debe introducir un tensor bidimensional de 6 muestras (se han obtenido modelos que aceptan 6 horas) con 7 variables de entrada cada una que se corresponde a:

- **t_ext:** temperatura exterior media en una hora (-50, 50).
- **h_ext:** humedad exterior media en una hora (0, 100).
- **t_C_cal:** temperatura media de la sonda de temperatura más superficial en una hora (-50, 50).
- **h_C_cal:** humedad media de la sonda de humedad más superficial en una hora (0, 100).
- **t_L_cal:** temperatura media de la sonda de temperatura interna en una hora (-50, 50).
- **h_L_cal:** humedad media de la sonda de humedad interna en una hora (0, 100).
- **sensor:** sensor al que se corresponde los datos (0, 7).

Por otro lado, las variables deberán estar normalizadas en el rango 0-1, con los máximos y mínimos especificados anteriormente.

En el caso del modelo *MLP*, en lugar de un tensor tridimensional, se deberá modificar para que se corresponda con un vector del número de variables por el de muestras.

La salida será igualmente un tensor bidimensional del número de predicciones establecidas en el entrenamiento del modelo con 6 variables cada una que se corresponden a las mencionadas anteriormente a excepción del número de sensor. De forma inversa, el modelo proporcionará datos normalizados, por lo que para obtener cada atributo en un rango correcto deberá de denormalizarse.

Bibliografía

[1] W. Commons, "File:gated recurrent unit, base type.svg — wikimedia commons, the free media repository," 2022, [Online; accessed 27-May-2023]. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:Gated_Recurrent_Unit,_base_type.svg&oldid=654928160

[2] ——, "File:lstm cell.svg — wikimedia commons, the free media repository," 2022, [Online; accessed 27-May-2023]. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:LSTM_cell.svg&oldid=713650305

[3] M. Palacio, *Scrum Master*. Iubaris Info 4 Media SL, 2021, ch. 1, p. 11.

[4] "Zenhub." [Online]. Available: https://www.zenhub.com/

[5] "Seguridad social: Cotización / recaudación de trabajadores." [Online]. Available: https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537#36538

[6] "Welcome to the apache software foundation!" [Online]. Available: https://www.apache.org/licenses/LICENSE-2.0

[7] Wikipedia, "Filtro de hodrick-prescott — wikipedia, la enciclopedia libre," 2023, [Internet; descargado 3-febrero-2023]. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Filtro_de_Hodrick-Prescott&oldid=149041721

[8] "Python." [Online]. Available: https://www.python.org/downloads/release/python-3913/

[9] Microsoft, "Visual studio code - code editing. redefined," Nov 2021. [Online]. Available: https://code.visualstudio.com/

[10] "Anaconda navigator." [Online]. Available: https://anaconda.org/anaconda/anaconda-navigator

[11] "Tensorflow." [Online]. Available: https://www.tensorflow.org/?hl=es-419

[12] "Guardando y serializando modelos con tensorflow keras | tensorflow core." [Online]. Available: https://www.tensorflow.org/guide/keras/save_and_serialize?hl=es-419