



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 — 24 de junio
de 2023

Tutores: Rubén Ruiz González, Alejandro
Merino Gómez

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	v
Apéndice A Introducción	1
Apéndice B Plan de Proyecto	3
B.1. Introducción	3
B.2. Planificación temporal	3
B.3. Estudio de viabilidad	10
Apéndice C Comprensión del negocio	15
C.1. Introducción	15
C.2. Objetivos del proyecto	15
C.3. Requisitos del proyecto	15
Apéndice D Comprensión de los datos	17
D.1. Introducción	17
D.2. Descripción de los datos	17
D.3. Visualización de los datos	19
Apéndice E Preparación de los datos	29
E.1. Introducción	29
E.2. Valores faltantes	30
E.3. Valores extremos	30
E.4. Valores erróneos	30

E.5. Visualización de los datos	31
Apéndice F Modelado	39
F.1. Introducción	39
F.2. Transformación del conjunto de datos	39
F.3. GRU	41
F.4. LSTM	41
F.5. MLP	42
Apéndice G Evaluación	45
G.1. Introducción	45
G.2. Resultados de GRU	47
G.3. Resultados de LSTM	51
G.4. Resultados de MLP	55
G.5. Comparativa entre modelos	58
Apéndice H Documentación técnica de programación	61
H.1. Introducción	61
H.2. Estructura de directorios	61
H.3. Manual del programador	62
H.4. Pruebas del sistema	66
Apéndice I Documentación de usuario	69
I.1. Introducción	69
I.2. Requisitos de usuarios	69
I.3. Instalación	69
I.4. Manual del usuario	70
Bibliografía	71

Índice de figuras

D.1. Datos no procesados: sensor 1	19
D.2. Datos no procesados: sensor 2	20
D.3. Datos no procesados: sensor 3	21
D.4. Datos no procesados: sensor 4	22
D.5. Datos no procesados: sensor 5	23
D.6. Datos no procesados: sensor 6	24
D.7. Datos no procesados: sensor 7	25
D.8. Datos no procesados: sensor 8	26
D.9. Datos no procesados: pluviómetro	27
E.1. Representación gráfica del método del rango intercuartílico . . .	30
E.2. Datos procesados: sensor 1	31
E.3. Datos procesados: sensor 2	32
E.4. Datos procesados: sensor 3	33
E.5. Datos procesados: sensor 4	34
E.6. Datos procesados: sensor 5	35
E.7. Datos procesados: sensor 6	36
E.8. Datos procesados: sensor 7	37
E.9. Datos procesados: sensor 8	38
F.1. Selección de la serie de tiempo	40
F.2. Representación celda GRU. Extraído de [1]	41
F.3. Representación celda LSTM. Extraído de [2]	41
F.4. Diagrama de entradas y salidas MLP	42
F.5. Función ReLU	43
G.1. Comparativa de número de capas para GRU	48
G.2. Comparativa de número de unidades para GRU	49

G.3. Comparativa de ratio de aprendizaje para GRU	49
G.4. Comparativa de ratio de aprendizaje para GRU (0.01 y 0.001)	50
G.5. Comparativa de número de capas para LSTM	52
G.6. Comparativa de número de unidades para LSTM	53
G.7. Comparativa de ratio de aprendizaje para LSTM	53
G.8. Comparativa de ratio de aprendizaje para LSTM (0.01)	54
G.9. Comparativa de número de capas para MLP	56
G.10.Comparativa de número de unidades para MLP	57
G.11.Comparativa de ratio de aprendizaje para MLP	57
G.12.Comparativa de número de capas	58
G.13.Comparativa de número de unidades	59
G.14.Comparativa de ratio de aprendizaje	59
G.15.Comparativa de ratio de aprendizaje ampliado	60
H.1. Búsqueda del entorno virtual	63
H.2. Selección del entorno virtual	63
H.3. Ejecución del servicio de Jupyter Notebook con GPU	65
H.4. Selección de GPU en Google Colab	66

Índice de tablas

B.1. Costes de Hardware	11
B.2. Costes de Hardware	11
B.3. Costes de Software	12
B.4. Costes varios	12
B.5. Costes totales	12
B.6. Licencias de las dependencias utilizadas	13
D.1. Atributos de los datos: sensores	18
D.2. Atributos de los datos: pluviómetro	18
G.1. Evaluación GRU	47
G.2. Evaluación LSTM	51
G.3. Evaluación MLP	55

Apéndice A

Introducción

Al no tratarse de un proyecto de desarrollo de software, sino que se encuentra enmarcado dentro del campo de *Machine Learning*, se seguirá una estructura en los anexos algo diferente a la propuesta original. Se mantienen los apartados clave como son el plan de proyecto, el manual de programador y usuario y se sustituyen el resto por las diferentes fases de descubrimiento de conocimiento en bases de datos (*KDD*).

Estas se corresponderán con el conocimiento del negocio, conocimiento de datos, preparación de datos, modelado y evaluación de los resultados.

Apéndice *B*

Plan de Proyecto

B.1. Introducción

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

B.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 desarrollando, 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 atender diferentes cuestiones como cuáles son los problemas que ralentizan su avance.

Por otro lado, la herramienta software que apoyaba su aplicación (ZenHub [4]) dejó de estar disponible de forma abierta, de modo 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 fue 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 creaban, 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 tareas iniciales se correspondían con la fase de comprensión de datos y a su vez con la preparación de estos, 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 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, el pluviómetro en las dependencias del viñedo en algunas de las muestras arrojaba valores inverosímiles debido a la forma en la que se encontraba instalado, de manera que el balancín empleado para medir las diferencias de precipitaciones entre instantes se activaba artificialmente por acción del viento.

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 su naturaleza. 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 comprobar los valores obtenidos en los conjuntos.

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

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 para continuar con la inspección visual de los sensores y comenzar con las muestras del pluviómetro. Con estas inspecciones se consigue, de esta forma, 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 la variación excesiva en temperatura, corrigiendo el ruido presente.
- En el sensor 3 se encontró que el salto de humedad no era genuino, por lo que había que realizar un estudio para determinar su viabilidad.
- 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 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 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 en el 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 de los conjuntos de otros sensores realizando procesos similares. Por otro lado, se propuso emplear WeatherBit como *API* meteorológica para eliminar las muestras 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 mayor cantidad de ruido posible en todos los sensores. Por otro lado, se fueron finalizando las selecciones de los datos, realizando de igual forma una recuperación de valores 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 era irrecuperable debido a las variabilidades aleatorias con las que parecía contar.

Así mismo, se comenzó con la limpieza de las muestras del pluviómetro empleando los datos de varias *API*, en primer lugar, se trató de utilizar la mencionada anteriormente, estando el *endpoint* ideal bajo licencia de pago. Probando con OpenWeatherMap se presentó el caso al soporte, consiguiendo ampliar los privilegios de usuario para poder acceder a la 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 acceder a datos de más de hace un año, sin embargo, no permitía obtener registros 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 realizada. Se plantea la implementación de una matriz de correlación para observar cuáles de los atributos seleccionar para el entrenamiento.

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 documentado con las explicaciones pertinentes.

Por otro lado, se realizan avances sustanciales en la memoria del proyecto añadiendo información en 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 neuronales 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 seleccionan los datos de entrada a los modelos, empleando medias diarias para tal propósito y dividiendo 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 afecten a la regresión, realizando medias 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, que permite obtener las tendencias de una serie temporal, de modo que los modelos sean capaces de generalizar de manera más precisa.

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 días 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 el método mencionado para la selección de datos del modelo, de modo que se pudiera evitar la situación anterior.

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

En la planificación se acordó intentar rebajar los periodos de tiempo, emplear valores 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 continuaban generalizando como si es necesario realizar cambios en el proceso de conocimiento de datos y preparación.

De esta manera, se redujo la frecuencia de las medias de los datos, realizando su integración 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.

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 los conceptos teóricos de la memoria y se realizaron algunas correcciones.

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

En la planificación se indicó la continuación en los esfuerzos de realización de memoria y anexos, de forma que este *sprint* se centró en la finalización de los mismos.

Sprint 11 (30/05/2023 - 06/06/2023)

En la iteración se acordó la realización de más pruebas cambiando el número de predicciones en las salidas de los modelos, empleando para ello herramientas como Google Colab, de manera que pudieran agilizarse los procesos para obtener los resultados.

Se propuso, por otro lado, la utilización nativa de librerías de *IA* empleando *CUDA* como forma de agilizar el proceso de entrenamiento.

De este modo, además de las ejecuciones ya realizadas, se estableció añadir las verificaciones con 2 y 3 predicciones en la salida de los modelos.

Sprint 12 (06/06/2023 - 13/06/2023)

En la reunión de la iteración se acordó comenzar a realizar las correcciones de la memoria y anexos con los comentarios proporcionados por los docentes.

Por otro lado, se realizaron las investigaciones para preparar el entorno de ejecución acelerada, comenzando con Google Colab y siguiendo con la implementación de un contenedor de Docker que virtualiza un servicio de Jupyter Notebook que permite emplear la GPU para el entrenamiento.

Además, se realizan las primeras correcciones basadas en los comentarios recibidos en la primera parte de la memoria.

Sprint 13 (13/06/2023 - 20/06/2023)

En el *sprint* se acordó continuar con las correcciones de memoria y anexos, comenzando, a su vez, con la preparación de la presentación del proyecto. Por otro lado, se indica la problemática con el contenedor de

Docker implementado (era preciso instalar una dependencia en el contenedor proporcionado por TensorFlow), acordando realizar una investigación más extensa en el campo para poder realizar la instalación y poder emplear el servicio adecuadamente, pudiendo, de esta manera, ejecutar el entrenamiento con la GPU.

De este modo, se continuó la investigación de la virtualización del servicio mencionado en la iteración anterior, modificando el *script* correspondiente del contenedor para que previamente instalara la dependencia necesaria.

Por otra parte, se comienza la corrección de los anexos en base a los comentarios de los docentes y se elabora un índice preliminar de los puntos a tratar en la presentación del proyecto, iniciando, de este modo, la presentación de diapositivas.

Sprint 14 (20/06/2023 - 27/06/2023)

En la reunión de la iteración se acuerda la introducción de sistemas más visuales para presentar los resultados obtenidos, así como la comparativa entre ellos, siendo necesario la modificación del anexo de evaluación en el presente documento y la elaboración de las gráficas mencionadas. Por otro lado, se indica la grabación de vídeos que muestren la ejecución de los entrenamientos debido al poco tiempo disponible en la defensa del proyecto en comparación con los conceptos y metodologías que precisan de una explicación más extensa.

Durante el desarrollo del *sprint* se comenzó la elaboración de las gráficas mencionadas con los diferentes resultados obtenidos de las ejecuciones de las pruebas.

B.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 entorno 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 económica es necesario hacer una diferenciación entre los costes y los beneficios que conlleva 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 una 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 B.1: Costes de Hardware

El porcentaje mensual de aporte a la Seguridad 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 B.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 B.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 B.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 B.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, al emplear los resultados en algún producto o servicio posterior.

Por otro lado, a pesar de que no existan unas ganancias directas cuantificables en términos monetarios en el propio desarrollo del proyecto, podría mejorar tanto la comprensión de las influencias de las humedades y temperaturas en los cultivos, como la calidad de los productos resultantes al permitir, por ejemplo, planificar en base a las predicciones obtenidas o los conocimientos extraídos las estrategias de riegos o abonados.

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 en 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 B.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, requiriendo la conservación del aviso de derecho de autor y un descargo de responsabilidad, sin embargo, no es *copyleft* [6].

Por su versatilidad se escoge la opción MIT, puesto que se trata de una licencia permisiva de software libre, imponiendo muy pocas restricciones y otorgando una buena compatibilidad con otras licencias como la mencionada anteriormente.

Apéndice C

Comprensión del negocio

C.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.

C.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 manera, correlaciones y poder encontrar modelos predictivos para anticipar la evolución futura de las diferentes variables en un plazo de tiempo determinado.

C.3. Requisitos del proyecto

Entre los requisitos del proyecto se encontraría la creación de modelos adecuados a los datos que predigan correctamente a corto o medio plazo las diferentes variables objetivo. Estas estarían compuestas por las humedades y temperaturas del suelo de los distintos sensores.

Apéndice D

Comprensión de los datos

D.1. Introducción

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

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 entregas inamovibles.

D.2. Descripción de los datos

Se proporciona un conjunto de datos compuesto por varios sensores (en concreto 8 sensores *IoT*) y un pluviómetro desplegado en un viñado.

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

Atributo	Descripción	Unidades
ts	<i>Timestamp</i> de la muestra	ms
fecha	Fecha de la muestra	Fecha en formato yy/m-m/dd, hh:mm:ss
batería	Nivel de batería del sensor	V
t_ext	Temperatura ambiental	°C
h_ext	Humedad ambiental	%
t_C_cal	Temperatura superficial a 20cm.	°C
h_C_cal	Humedad superficial a 20cm.	%
t_L_cal	Temperatura a profundidad mayor a t_C_cal	°C
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 D.1: Atributos de los datos: sensores

Los datos del pluviómetro cuentan con los siguientes atributos:

Atributo	Descripción	Unidades
ts	<i>Timestamp</i> de la muestra	ms
fecha	Fecha de la muestra	Fecha en formato yy/m-m/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 D.2: Atributos de los datos: pluviómetro

D.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. De igual manera, los valores 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

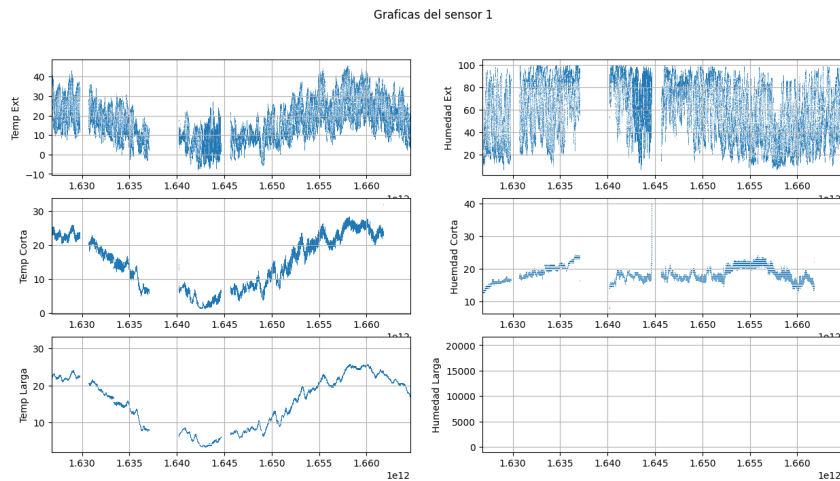


Figura D.1: Datos no procesados: sensor 1

En el primer sensor, como puede observarse en la Figura D.1, se encuentran varias situaciones que *a priori* son sencillas de solucionar, puesto que existe una cantidad importante de muestras para las que no se cuentan con datos, además de 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 encontrarse dentro de la normalidad, por lo que las labores en el sensor mencionado serán de las menos tediosas.

Los aumentos tan repentinos como los observados en la imagen posteriormente se demostraron que eran debidos al agotamiento de la batería,

que poco antes de producirse provocaba que las sondas arrojaran valores artificiales para luego realizar lecturas nulas.

Sensor 2

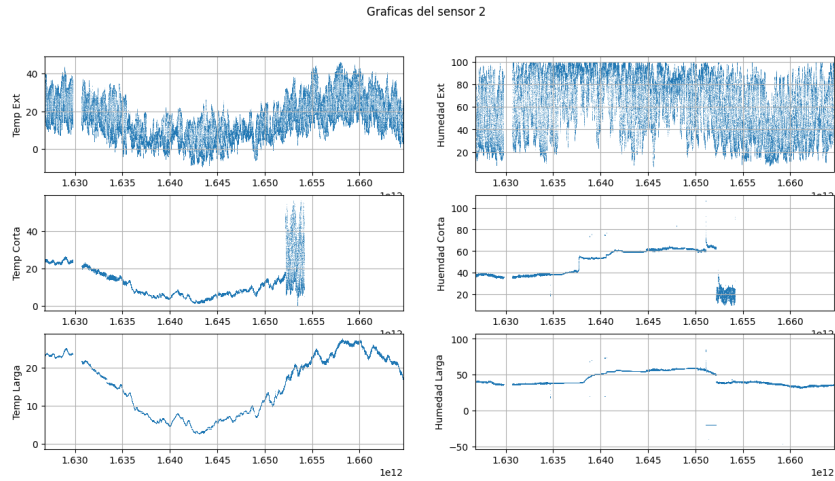


Figura D.2: Datos no procesados: sensor 2

En el segundo caso, además de las muestras para las que no se cuenta con valores, en la Figura D.2 se observa cómo llegado un instante en las sondas de temperatura y humedad superficiales se 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 sus capacitancias.

Sensor 3

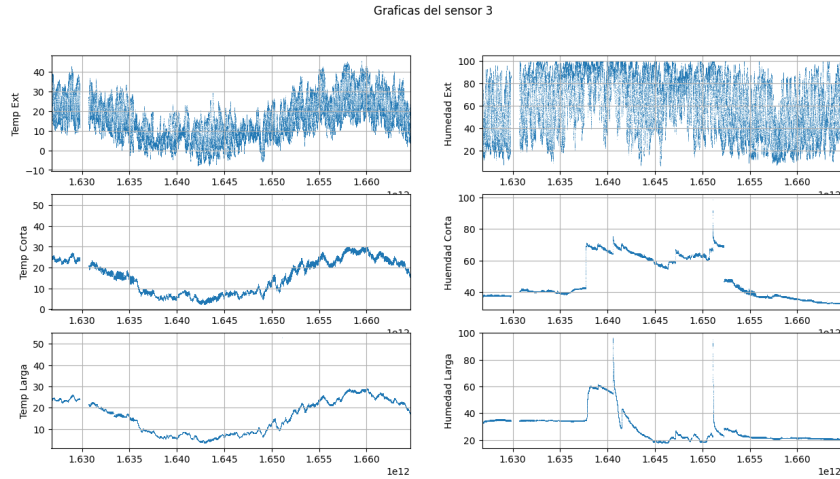


Figura D.3: Datos no procesados: sensor 3

En este sensor, como puede apreciarse en la Figura D.3, además de las muestras para los que no tenemos valores (que será la tónica habitual en el conjunto de datos), existen aumentos remarcables en las 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

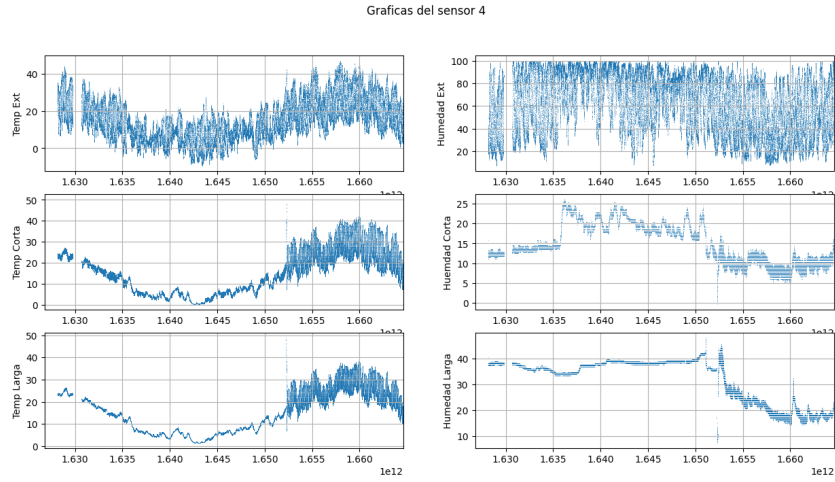


Figura D.4: Datos no procesados: sensor 4

En este caso, pueden observarse en la Figura D.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 esta situación, puede apreciarse cómo la variabilidad de las muestras aumenta de forma homogénea en todas y cada una de las sondas, a salvedad de las que recogen valores ambientales.

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

Sensor 5

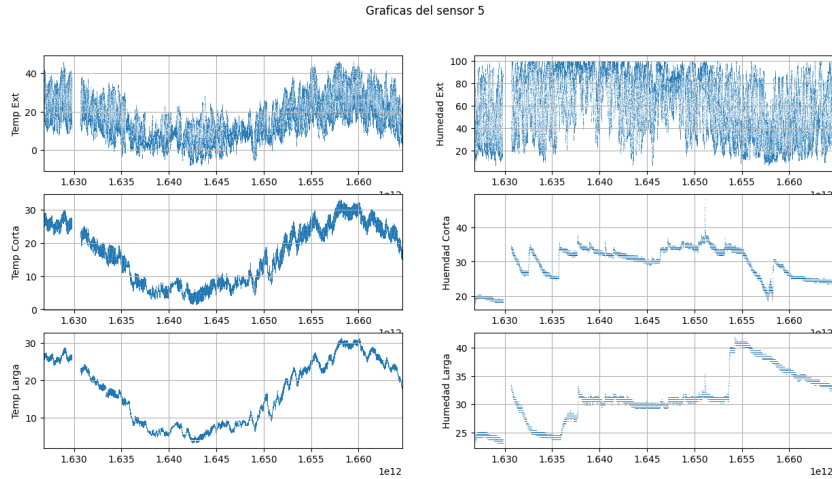


Figura D.5: Datos no procesados: sensor 5

En la Figura D.5 puede observarse diferentes situaciones. Una de ellas es la repetida falta de valores para algunos instantes de tiempo; la segunda son los aparentes aumentos repentinos de humedades.

Estos siguen el mismo patrón en todas las situaciones: incremento que puede clasificarse como repentino, y un descenso moderado a lo largo del tiempo. De esta forma, todos se contrastaron como genuinos observando ciertos parámetros como el tiempo atmosférico en los instantes de subida.

Este sensor cuenta, en términos generales, con uno de los conjuntos de datos con menos singularidades de todos los analizados, formando parte de los conjuntos con menor transformación de todos.

Sensor 6

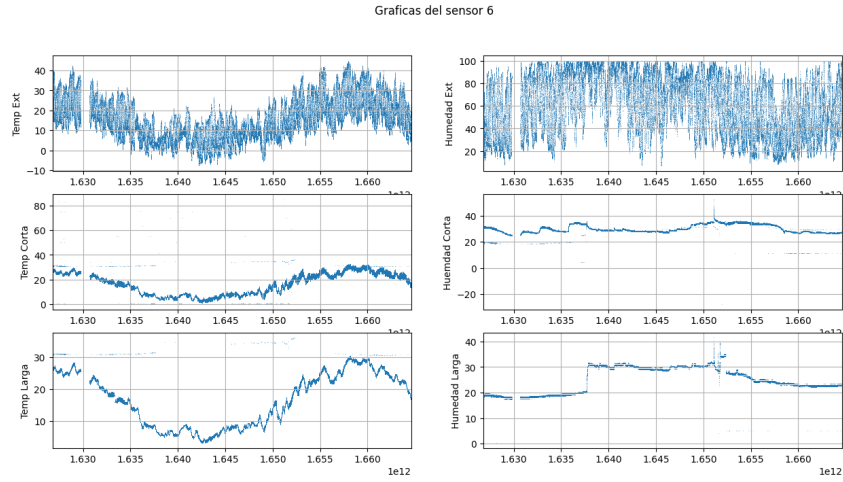


Figura D.6: Datos no procesados: sensor 6

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

Una de ellas es la presencia generalizada de ruido 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

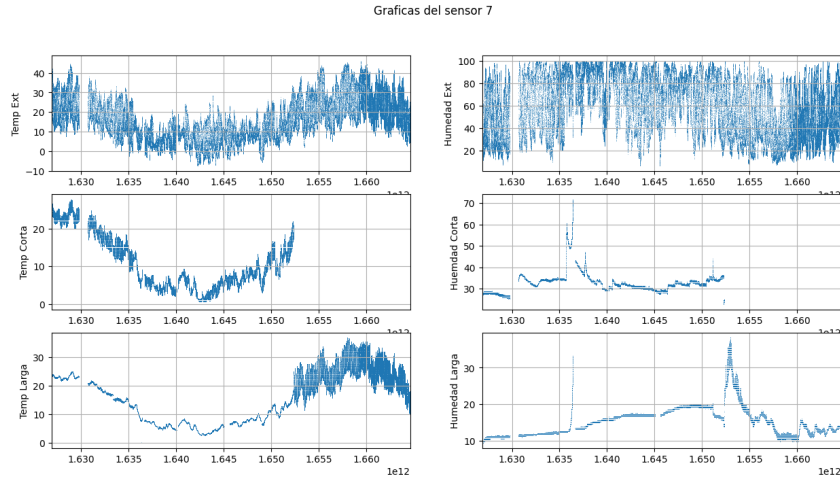


Figura D.7: Datos no procesados: sensor 7

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

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, al igual que en el sensor 4, se observa un incremento significativo de la variabilidad de la temperatura del suelo, produciéndose, además, un aumento de las lecturas en la sonda de humedad más profunda.

De igual manera que en el sensor mencionado, las labores en los cultivos habían provocado un cambio en el emplazamiento original del sensor, lo que producía las 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

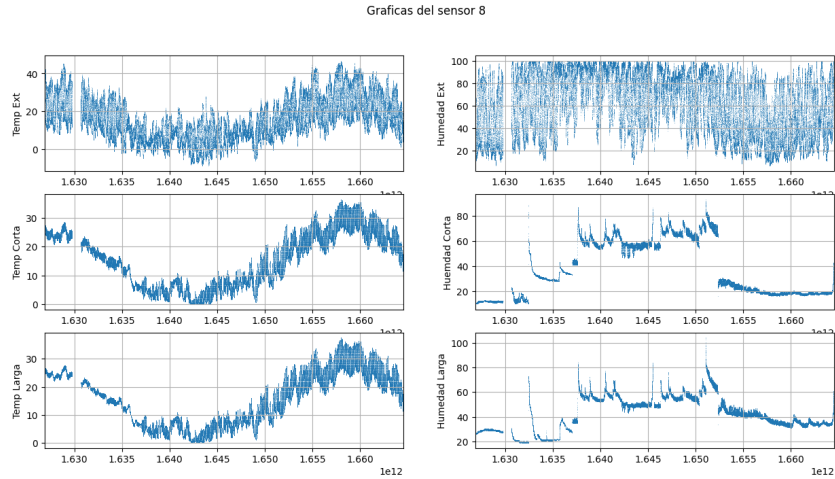


Figura D.8: Datos no procesados: sensor 8

Como puede observarse en la figura D.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 término, los aumentos de temperatura debidos al agotamiento de la batería y finalmente la alta variabilidad en las temperaturas, que, por otro lado, se asemejan en gran medida a la situación del sensor 4, lo que indica que de nuevo las sondas habían sido desplazadas de su emplazamiento original.

Este será uno de los sensores con el conjunto de datos con más problemas de todos los expuestos anteriormente, encontrándose de igual forma con incidencias continuas por los niveles de batería.

Pluviómetro

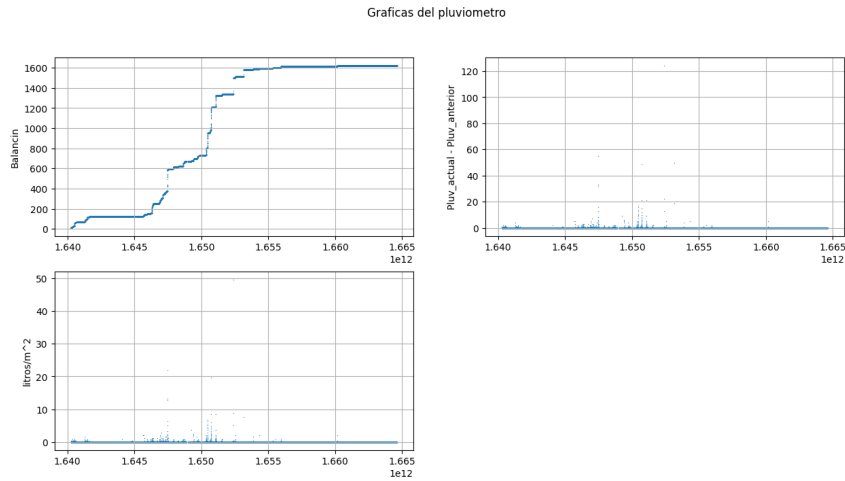


Figura D.9: Datos no procesados: pluviómetro

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

Apéndice E

Preparación de los datos

E.1. Introducción

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

Este proceso se compone de 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 es posible encontrar valores erróneos, que pueden ser estadísticamente correctos, pero no tener sentido contextual. En este caso concreto se tratarán de incrementos o bajadas inverosímiles de temperaturas y/o humedades del suelo, que se encuentran dentro de los valores naturales, pero que es difícil que tengan lugar a la velocidad a la que se producen. 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-Prescott [7]), puesto que no existe un interés específico en los valores concretos, sino en las posibles tendencias de las diferentes variables, que nos permitan obtener una predicción adecuada en cada circunstancia. De esta manera, se calcula la tendencia diaria de cada periodo temporal.

E.2. Valores faltantes

En cuanto a los valores faltantes, la decisión fue eliminar las muestras que tuvieran nulos 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.

E.3. Valores extremos

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

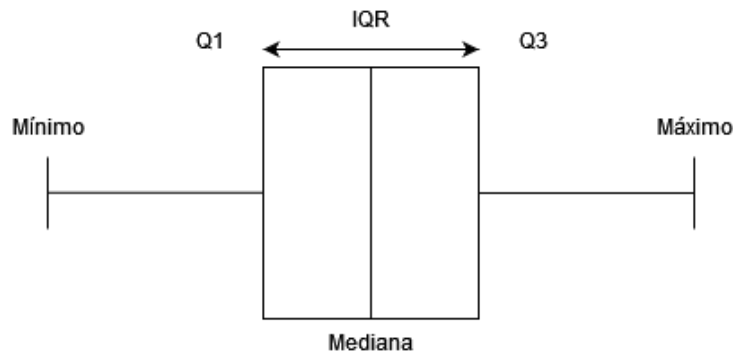


Figura E.1: Representación gráfica del método del rango intercuartílico

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.

E.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 esta detección se desarrolló de forma visual directamente 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.

E.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

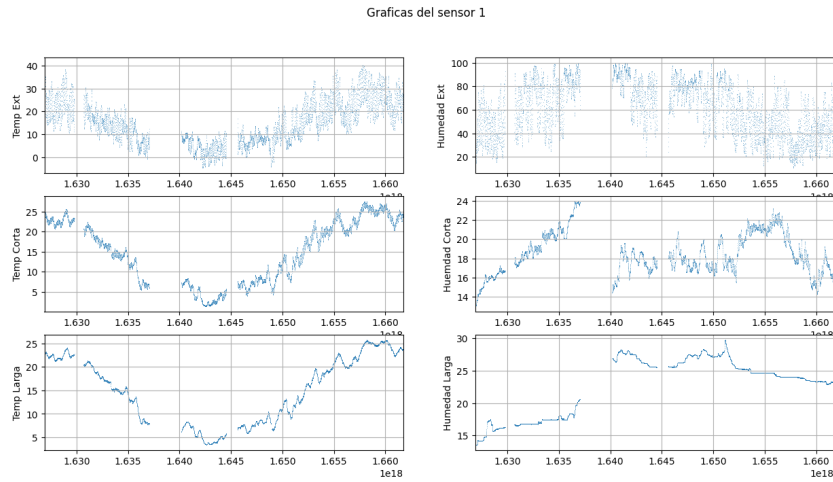


Figura E.2: Datos procesados: sensor 1

Como puede observarse en la Figura E.2, se ha tratado el ruido presente en el conjunto 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 los valores siguientes.

Sensor 2

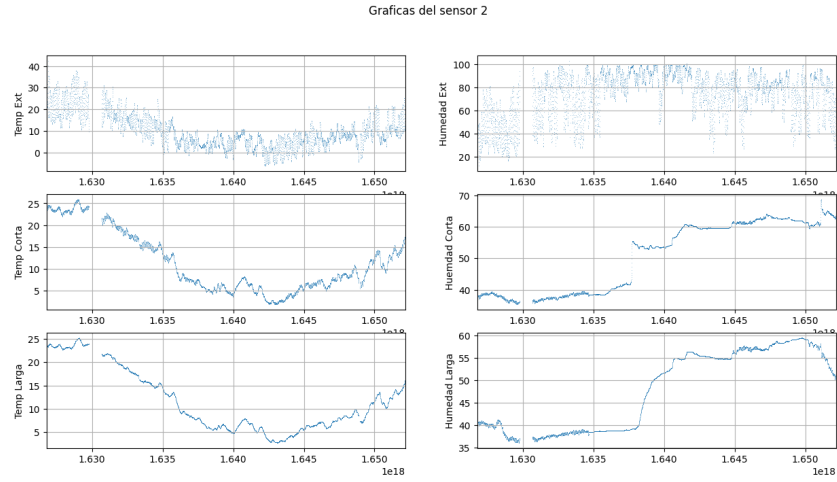


Figura E.3: Datos procesados: sensor 2

En este caso, como puede observarse en la Figura E.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

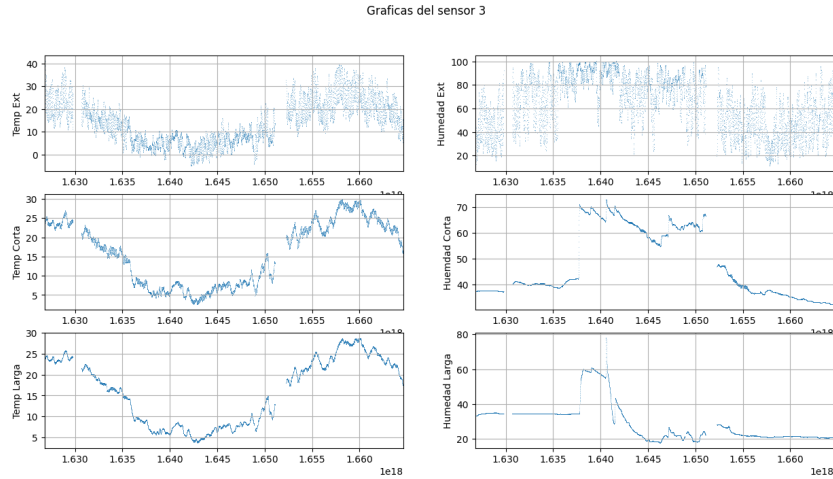


Figura E.4: Datos procesados: sensor 3

Como puede observarse en la Figura E.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 una porción moderada de las muestras totales que habían resultado del agotamiento de la batería del sensor.

Sensor 4

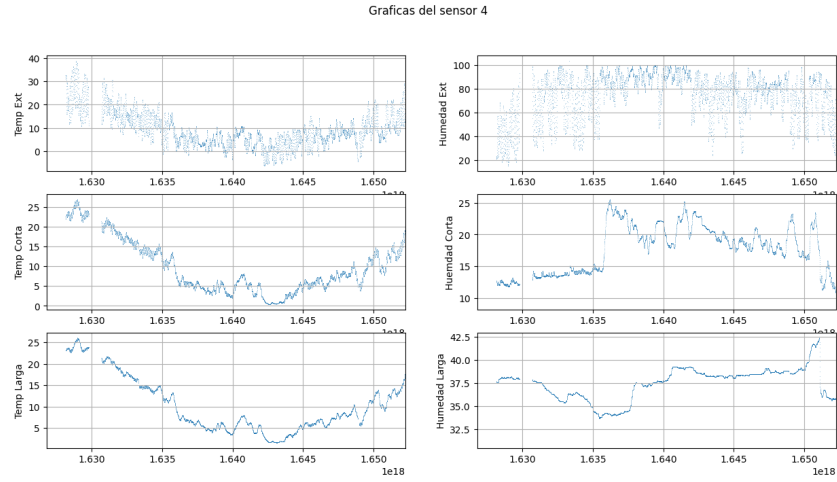


Figura E.5: Datos procesados: sensor 4

Como refleja la Figura E.5, se optó 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 preparado para ser empleado en los modelos.

Sensor 5

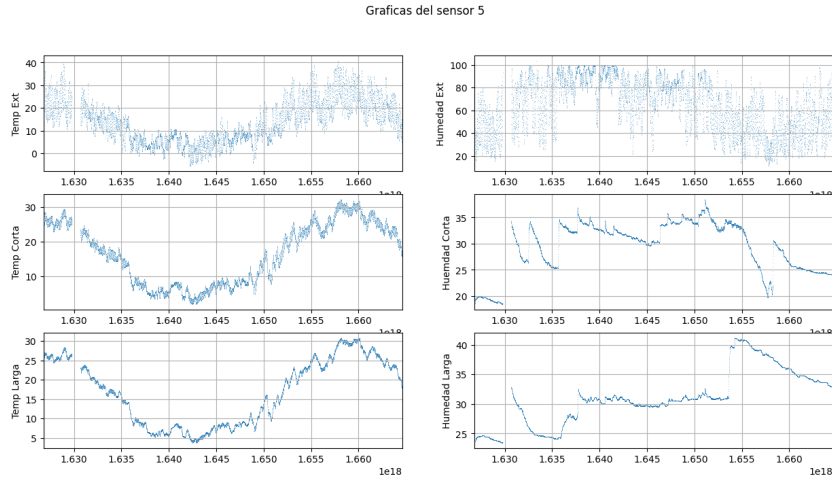


Figura E.6: Datos procesados: sensor 5

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 realizados fueron la detección de *outliers* o valores extremos y la eliminación de los ejemplos con valores faltantes.

Sensor 6

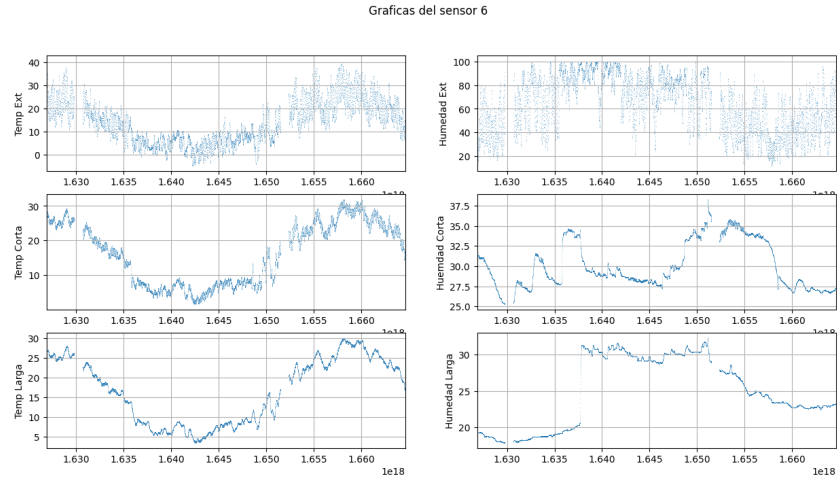


Figura E.7: Datos procesados: sensor 6

En el caso del sensor 6, como puede apreciarse en la Figura E.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

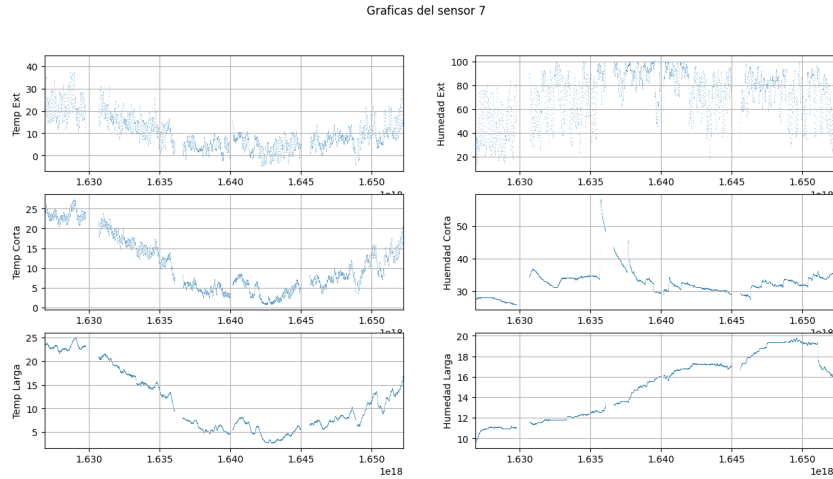


Figura E.8: Datos procesados: sensor 7

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 había atributos que no contaban con valores, además, al igual que en otros casos, el agotamiento de la batería producía sondeos incorrectos en las humedades.

Ante estos problemas la solución tomada fue eliminar los ejemplos afectados. En el primer caso puesto 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

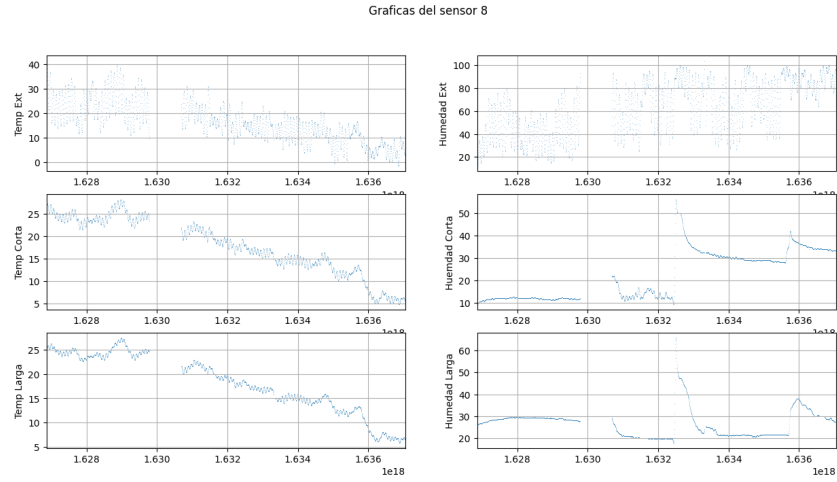


Figura E.9: Datos procesados: sensor 8

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

Hablando en términos de sesgos, es posible que, al realizar la recuperación de los valores en los sensores mencionados, puedan haberse sesgado en cierta medida los conjuntos de datos, sin embargo, estos 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 *F*

Modelado

F.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 considerar con la etapa anterior, pues existen herramientas que aúnan estos procedimientos.

En este caso, se toman por separado, teniendo en cuenta que los resultados en una fase afectan 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 y las respectivas predicciones, así como las diferentes características de cada uno de los modelos.

F.2. Transformación del conjunto de datos

El primer problema que es necesario afrontar es que estos deberán aceptar varias entradas de ejemplares, es decir, se necesitarán n muestras sucesivas para realizar la regresión.

Por otro lado, también deben ser capaces de realizar m predicciones, de forma que es preciso crear modelos con n muestras de entrada y m salidas 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, se necesita recorrer los datos en busca de los saltos temporales, para obtener los subconjuntos completos, y de esta manera, asegurar que dentro de éstos estas situaciones no se producen.

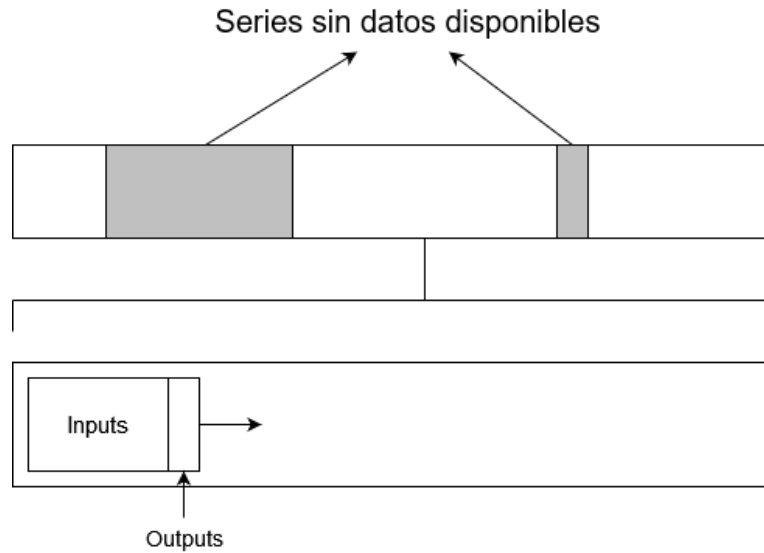


Figura F.1: Selección de la serie de tiempo

En la Figura F.1 puede observarse el proceso esquematizado. De esta forma, cada uno de los subconjuntos que no tienen saltos, se recorrerá mediante 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 contener muestras suficientes, es decir, $n + m$ ejemplos.

F.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 observarse en la Figura F.2.

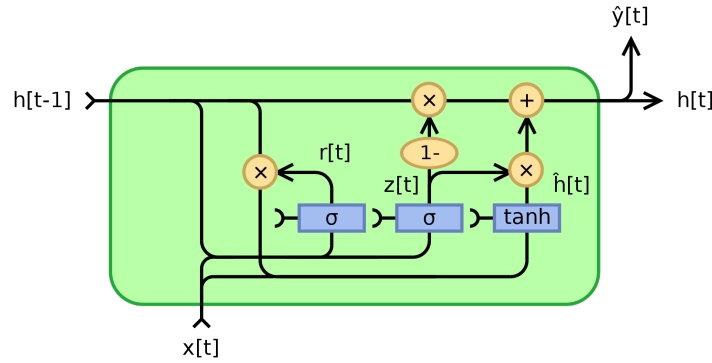


Figura F.2: Representación celda GRU. Extraído de [1]

F.4. LSTM

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

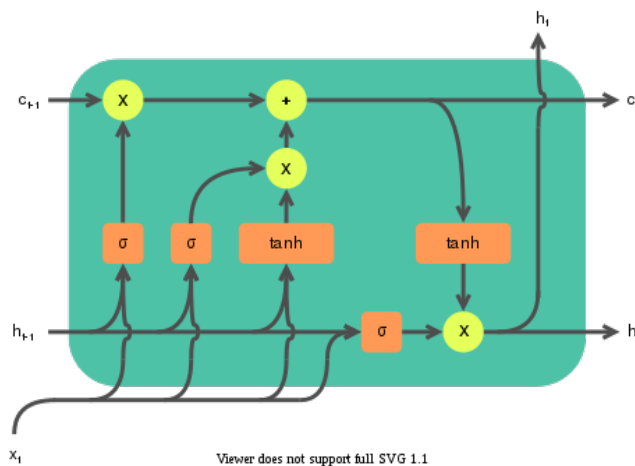


Figura F.3: Representación celda LSTM. Extraído de [2]

A diferencia de *GRU*, *LSTM* cuenta con 3 puertas diferentes en cada celda, la compuerta de olvido (*Forget Gate*), de entrada (*Input Gate*), de salida (*Output Gate*) y un estado oculto (*Cell State*), mientras que el primer modelo únicamente dispone de dos (*Reset Gate* y *Update Gate*).

De esta forma, en los modelos *LSTM*, se necesitarán realizar más operaciones que en *GRU*, además de almacenar más información por celda, de manera que el tiempo de entrenamiento será generalmente mayor.

F.5. MLP

El perceptrón multicapa, a diferencia de los otros dos modelos, no acepta la entrada de secuencias variables, por lo que se debe establecer una neurona de entrada por cada uno de los atributos, es decir, la red contendrá $a * n$ neuronas de entrada, siendo a el número de características de los ejemplos 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 ejemplares de salida.

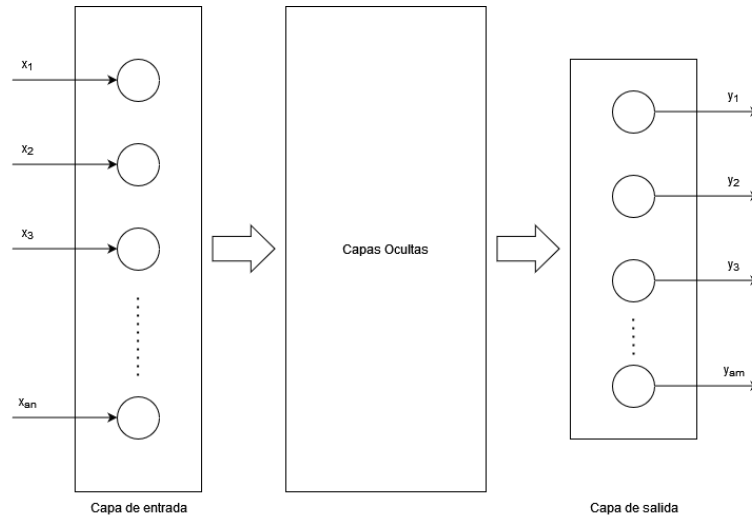


Figura F.4: Diagrama de entradas y salidas MLP

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

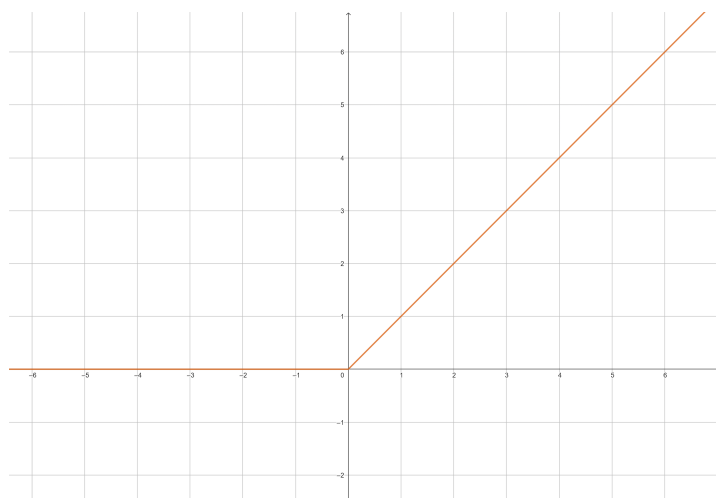


Figura F.5: Función ReLU

A pesar de que *MLP* no acepta entradas secuenciales y que por tanto requiere de 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 recurrentes, puesto que se realizan menos operaciones por término medio, además de necesitar de una cantidad de memoria menor para almacenar los pesos sinápticos.

Apéndice G

Evaluación

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

Para validarlos se realizarán pruebas variando el número de capas ocultas, así como el de celdas/neuronas en cada una de estas y el ratio de aprendizaje. De esta manera, se realizarán combinaciones con 1, 2 y 3 capas 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 dividido el conjunto de datos original en un conjunto de entrenamiento del 75 % de los disponibles por sensor, 15 % para validación siendo el restante de test.

Con respecto al número de muestras de entrada se emplearán 6 horas previas para realizar una predicción, empleando un tamaño de bloque de 256 ejemplos (los modelos no se entrenan con todos los datos en todas las iteraciones, sino que se emplean un subconjunto). En la salida, por otro lado, se realizarán predicciones de 1, 2 y 3 horas (marcadas como MSE 1, MSE 2 y MSE 3 en las respectivas tablas).

Además, se emplea un planificador de la tasa de aprendizaje, de forma que en la época 50 disminuye multiplicando $lr * e^{-0,1}$, 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, tomando como medida comparativa el error cuadrático medio de validación, lo que supone 405 ejecuciones por modelo, es decir, 1215 ejecuciones en total.

G.2. Resultados de GRU

En la Tabla G.1 se presentan los errores cuadráticos medios que se obtendrán para cada una de las pruebas. En esta se puede observar los resultados obtenidos con 1, 2 y 3 predicciones como salida de los modelos.

Capas	Tamaño capa	LR	MSE 1	MSE 2	MSE 3
1	8	0.001	1.72183E-05	3.98320E-05	5.84809E-05
1	8	0.01	3.37952E-06	7.99268E-06	1.41913E-05
1	8	0.1	0.00551	0.00127	0.00188
1	16	0.001	9.1907E-06	1.73932E-05	3.24371E-05
1	16	0.01	9.55771E-07	2.96765E-06	7.32603E-06
1	16	0.1	0.00474	0.00718	0.00138
1	32	0.001	4.50204E-06	1.22795E-05	2.51439E-05
1	32	0.01	5.96471E-07	1.69264E-06	4.50310E-06
1	32	0.1	0.00297	0.00158	0.00175
2	8	0.001	1.56769E-05	2.90745E-05	4.93188E-05
2	8	0.01	1.65277E-06	6.30449E-06	1.14688E-05
2	8	0.1	0.00965	0.00972	0.00026
2	16	0.001	5.07987E-06	1.33184E-05	2.59817E-05
2	16	0.01	5.66822E-07	1.45485E-06	3.49998E-06
2	16	0.1	0.01148	0.01579	0.00363
2	32	0.001	2.18717E-06	7.23831E-06	1.42593E-05
2	32	0.01	2.69205E-07	9.00049E-07	4.04636E-06
2	32	0.1	0.01727	0.00611	0.00797
3	8	0.001	1.61179E-05	2.57331E-05	4.20175E-05
3	8	0.01	1.42658E-06	5.39812E-06	9.51720E-06
3	8	0.1	0.00706	0.00029	0.00242
3	16	0.001	5.18860E-06	1.23398E-05	2.02378E-05
3	16	0.01	4.66034E-07	1.29760E-06	3.49093E-06
3	16	0.1	0.02418	0.01457	0.01497
3	32	0.001	2.36664E-06	5.75822E-06	1.13415E-05
3	32	0.01	8.90246E-07	9.57532E-07	2.41350E-06
3	32	0.1	0.00976	0.01064	0.01097

Tabla G.1: Evaluación GRU

Como puede observarse, las mayores eficiencias en términos generales se obtienen con un *learning rate* de 0.01, destacando los modelos de 2 capas con 32 y 16 celdas de memoria respectivamente.

Cabe destacar, además, que, como se podía intuir inicialmente, los errores aumentan generalmente a medida que realizamos más predicciones como salida.

A continuación se presentan las comparativas de los errores cuadráticos medios promedio para un número diferente de predicciones (1, 2 y 3 predicciones) del modelo estudiado y un parámetro diferente en cada ocasión.

En el caso del número de capas, como puede observarse en la Figura G.1, en promedio se obtendrán modelos con menor error generalmente en sistemas con 3 predicciones a la salida, siendo los modelos de 1 capa los que mayor precisión han presentado en las pruebas.

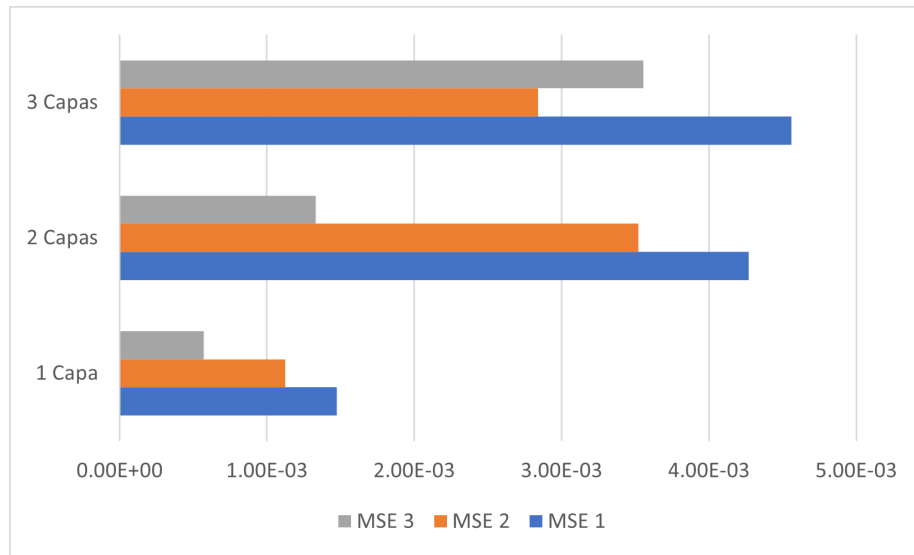


Figura G.1: Comparativa de número de capas para GRU

En lo que se refiere a las celdas de memoria, por término medio generalmente se obtendrán modelos más precisos con estructuras que realizan 3 predicciones por entrada, siendo la configuración de 8 unidades la que proporciona modelos con errores menores en promedio.

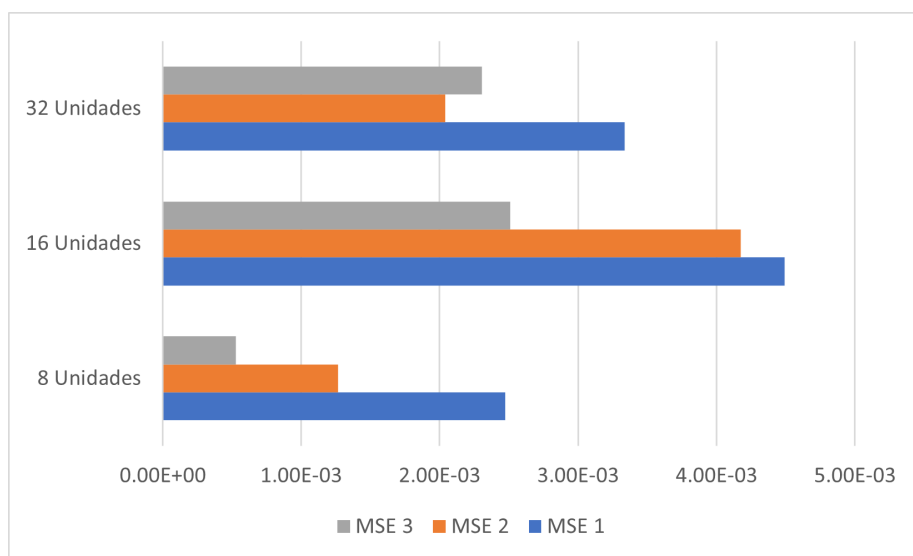


Figura G.2: Comparativa de número de unidades para GRU

Como puede apreciarse en la Figura G.3, con 0.1 como ratio de aprendizaje se obtienen los resultados con mayor error en promedio.

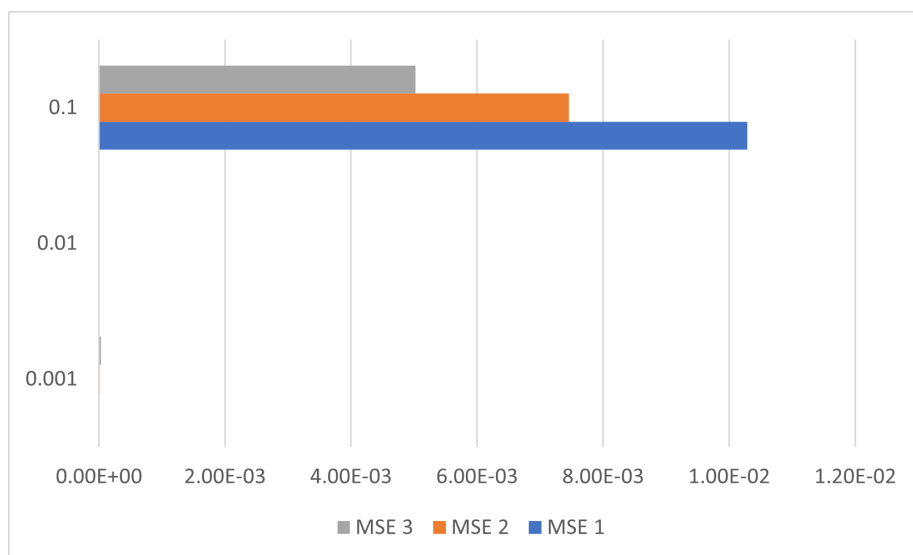


Figura G.3: Comparativa de ratio de aprendizaje para GRU

En la Figura G.4 puede observarse la diferencia del resto de los valores del parámetro. En este caso, se obtienen mejores resultados por término

medio con un ratio de aprendizaje de 0.01, siendo los modelos de 1 predicción por entrada los que generalmente obtienen errores menores.

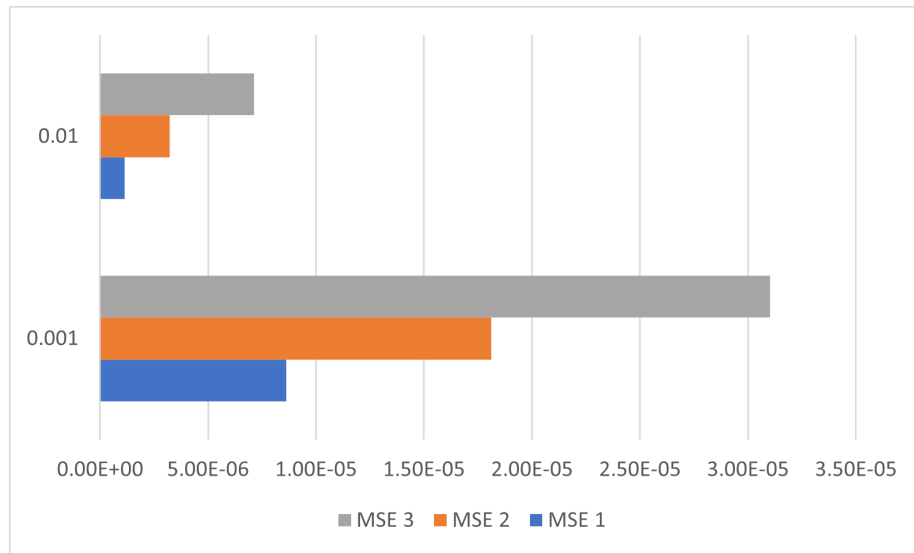


Figura G.4: Comparativa de ratio de aprendizaje para GRU (0.01 y 0.001)

G.3. Resultados de LSTM

Al igual que en la situación anterior, se realizarán pruebas con el mismo número de predicciones por cada entrada, obteniéndose los resultados reflejados en la Tabla G.2.

Capas	Tamaño capa	LR	MSE 1	MSE 2	MSE 3
1	8	0.001	2.12520E-05	3.03812E-05	5.60944E-05
1	8	0.01	3.4556E-06	1.05315E-05	1.68195E-05
1	8	0.1	3.9049E-06	2.55932E-05	2.56167E-05
1	16	0.001	1.18867E-05	1.98377E-05	4.04516E-05
1	16	0.01	1.31837E-06	3.46421E-06	7.71354E-06
1	16	0.1	3.14115E-06	7.24853E-06	7.55877E-05
1	32	0.001	5.84463E-06	1.46677E-05	2.83427E-05
1	32	0.01	6.62421E-07	1.96414E-06	5.22169E-06
1	32	0.1	9.51891E-07	0.00011	1.05361E-05
2	8	0.001	3.29916E-05	4.19003E-05	6.14671E-05
2	8	0.01	4.94574E-06	8.51327E-06	1.33202E-05
2	8	0.1	6.85972E-06	8.04903E-06	1.58783E-05
2	16	0.001	1.33461E-05	2.75447E-05	4.1725E-05
2	16	0.01	1.23340E-06	2.38821E-06	4.63577E-06
2	16	0.1	6.65626E-06	2.42052E-05	1.43046E-05
2	32	0.001	5.63944E-06	1.15909E-05	2.34114E-05
2	32	0.01	4.56945E-07	1.43223E-06	2.78594E-06
2	32	0.1	4.20038E-06	1.00814E-05	7.57666E-05
3	8	0.001	4.58400E-06	5.81849E-05	0.00012
3	8	0.01	5.90159E-06	9.07351E-06	1.48364E-05
3	8	0.1	0.00031	0.00486	1.73157E-05
3	16	0.001	2.22344E-05	3.47894E-05	5.07083E-05
3	16	0.01	1.17654E-06	2.41703E-06	4.06992E-06
3	16	0.1	0.00487	1.02880E-05	0.00490
3	32	0.001	8.16043E-06	1.64882E-05	2.57956E-05
3	32	0.01	6.54021E-07	1.19201E-06	2.58606E-06
3	32	0.1	0.00491	0.01459	0.00986

Tabla G.2: Evaluación LSTM

De forma similar que en *GRU*, en *LSMT* se generan mejores resultados con un *learning rate* de 0.01, destacando, de igual forma, los modelos con

2 capas, además de los de 3 capas y 32 celdas de memoria. Al igual que antes, los errores se incrementan, generalmente, con el aumento del número de predicciones.

A continuación, del mismo modo que en la sección anterior, se presentan las comparativas de los errores cuadráticos medios promedio para un número diferente de predicciones (1, 2 y 3 predicciones) del modelo estudiado y un parámetro diferente en cada ocasión.

En la Figura G.5 se observa que las configuraciones de 2 capas registran los errores más bajos en promedio que el resto de valores del parámetro, siendo los modelos de 1 predicción futura los que generalmente obtendrán mejores resultados.

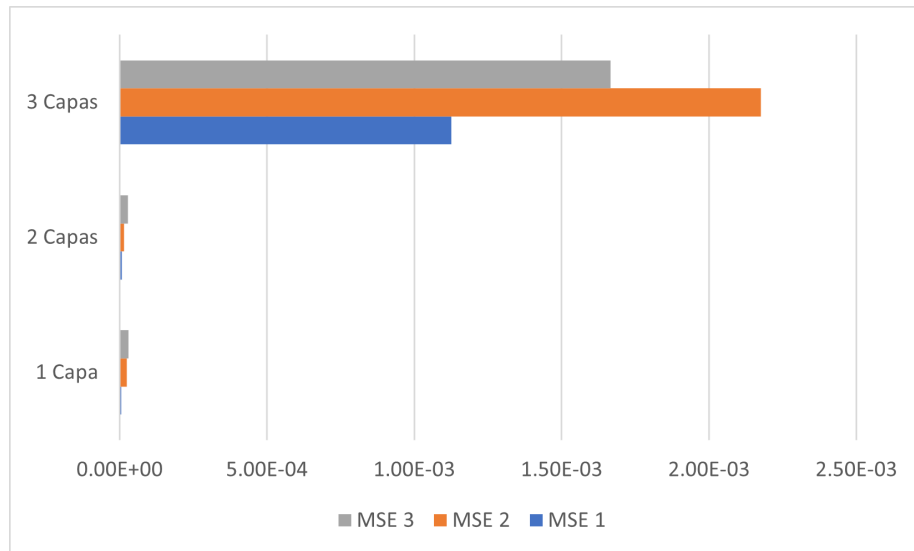


Figura G.5: Comparativa de número de capas para LSTM

En el caso del número de celdas de memoria, se obtendrán modelos más precisos por término medio con 8 unidades por capa (a excepción de los sistemas con 2 predicciones por entrada que contarán con mejor desempeño en redes con 16 unidades).

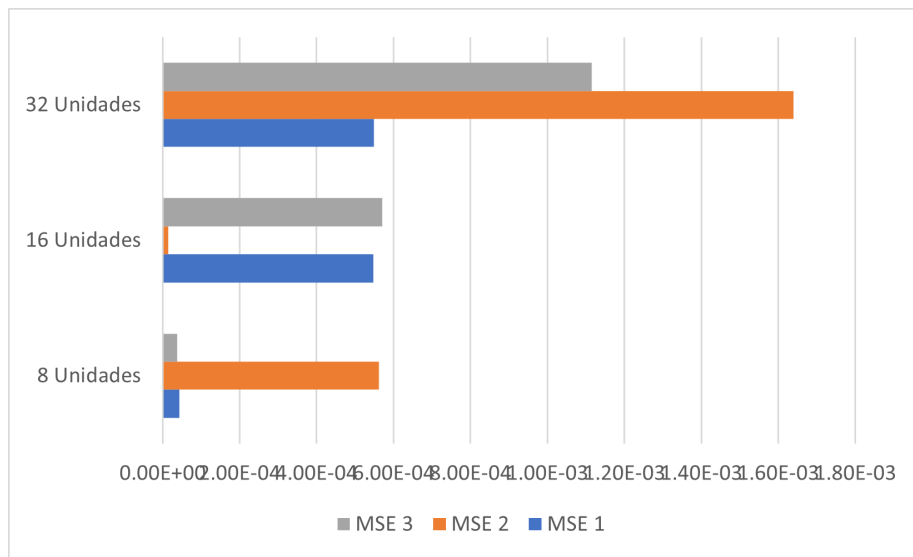


Figura G.6: Comparativa de número de unidades para LSTM

En lo referente al ratio de aprendizaje, los valores del parámetro para los que se obtienen modelos con menor precisión por término medio será 0.1, seguido de 0.001.

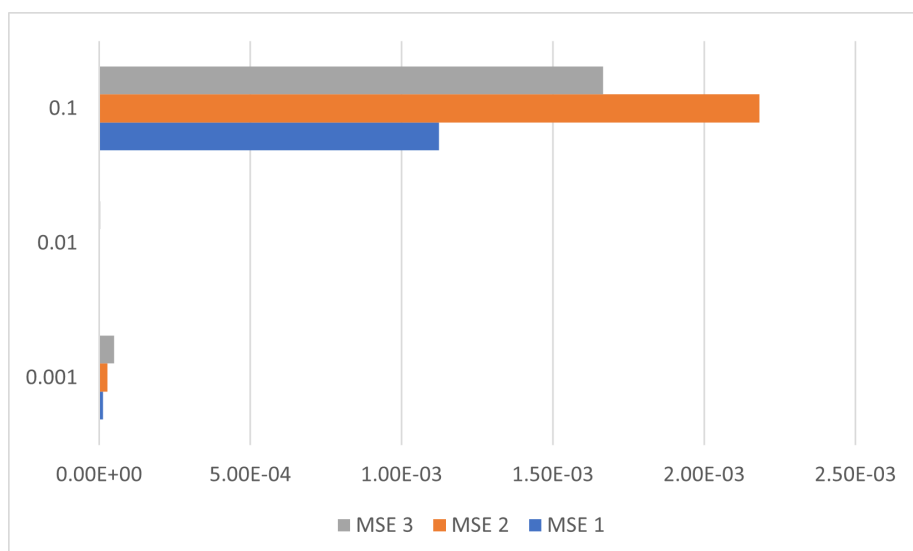


Figura G.7: Comparativa de ratio de aprendizaje para LSTM

En la Figura G.8 puede observarse que para 0.01 de ratio de aprendizaje se obtienen los resultados con mayor precisión por término con 1 y 2 predicciones por entrada. Siendo, de esta forma, la configuración de 1 predicción la que menores errores arroja en promedio en comparación con el resto de parámetros.

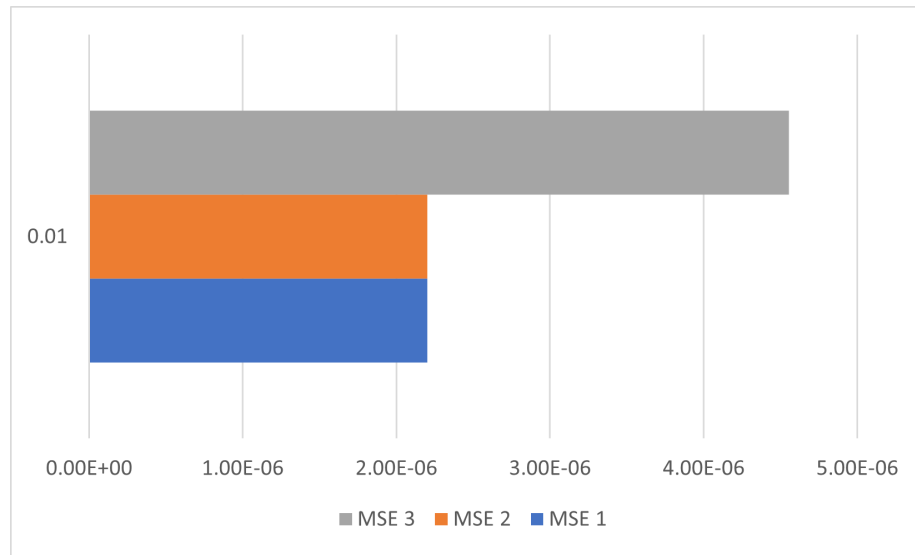


Figura G.8: Comparativa de ratio de aprendizaje para LSTM (0.01)

G.4. Resultados de MLP

En la Tabla G.3 se proporcionan los diferentes resultados obtenidos con *MLP*.

Capas	Tamaño capa	LR	MSE 1	MSE 2	MSE 3
1	8	0.001	0.00076	0.00017	0.00089
1	8	0.01	0.01218	0.00313	0.00229
1	8	0.1	0.02064	0.02415	0.02420
1	16	0.001	6.21736E-05	7.85435E-05	6.44446E-05
1	16	0.01	0.00059	0.00119	8.35920E-05
1	16	0.1	0.02056	0.02415	0.02420
1	32	0.001	1.33441E-05	2.31054E-05	4.27180E-05
1	32	0.01	5.57840E-06	1.32064E-05	2.49582E-05
1	32	0.1	0.02415E-05	0.02062	0.02420
2	8	0.001	0.00094	0.00043	0.00085
2	8	0.01	0.00673	0.00332	0.00327
2	8	0.1	0.02428	0.02430	0.02063
2	16	0.001	1.96797E-05	2.97114E-05	0.00013
2	16	0.01	0.00042	0.00040	0.00089
2	16	0.1	0.01624	0.02422	0.01343
2	32	0.001	8.32571E-06	2.15010E-05	3.63325E-05
2	32	0.01	5.56436E-06	5.32363E-05	2.65348E-05
2	32	0.1	0.02032	0.02060	0.02043
3	8	0.001	0.00152	0.00283	0.00195
3	8	0.01	0.00168	0.00284	0.00188
3	8	0.1	0.02417	0.02428	0.02041
3	16	0.001	5.74476E-05	5.53494E-05	0.00016
3	16	0.01	0.00031	0.00015	0.00095
3	16	0.1	0.02035	0.02036	0.01269
3	32	0.001	6.49630E-06	2.14812E-05	3.33377E-05
3	32	0.01	6.91262E-06	3.07683E-05	2.54072E-05
3	32	0.1	0.02028	0.01674	0.01653

Tabla G.3: Evaluación MLP

Al igual que en las anteriores secciones, se presentan las comparativas de los errores cuadráticos medios promedio para un número diferente de

predicciones (1, 2 y 3 predicciones) del modelo estudiado y un parámetro diferente en cada ocasión.

En el caso del número de capas, se puede observar (Figura G.9) que a diferencia de los otros modelos, la influencia del número de capas en la precisión es aparentemente menor.

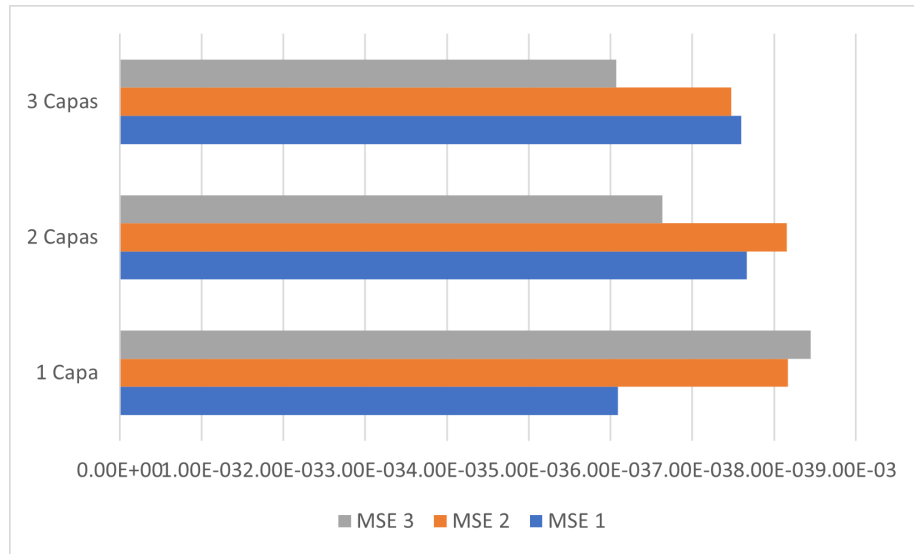


Figura G.9: Comparativa de número de capas para MLP

En cuanto al número de neuronas por capa, puede apreciarse que con 32 unidades se obtienen los modelos con mayor precisión por término medio, siendo la configuración de 2 predicciones por entrada la que generalmente arrojará resultados con menor error.

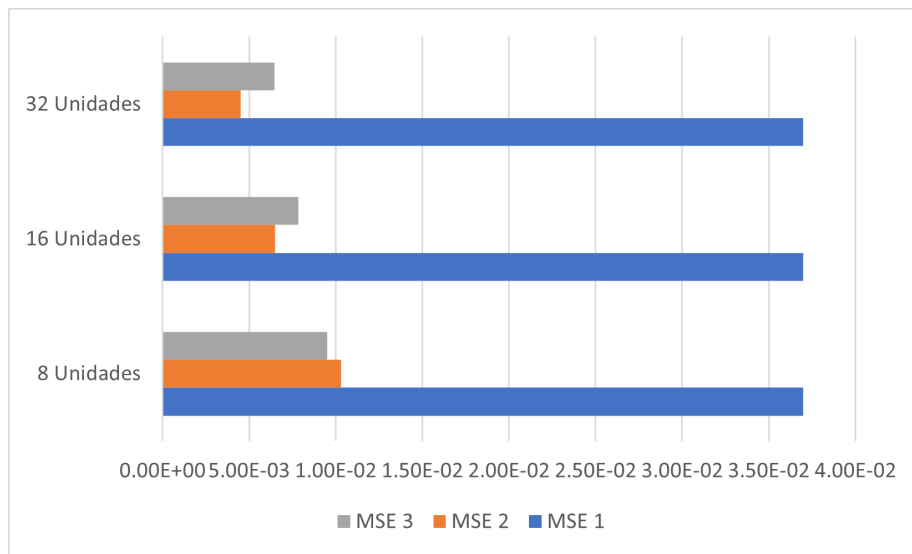


Figura G.10: Comparativa de número de unidades para MLP

Como puede observarse en la Figura G.11, el ratio de aprendizaje tiene una aparente mayor influencia sobre los resultados, siendo la configuración de 0.001 la que menores errores por término medio proporcionan.

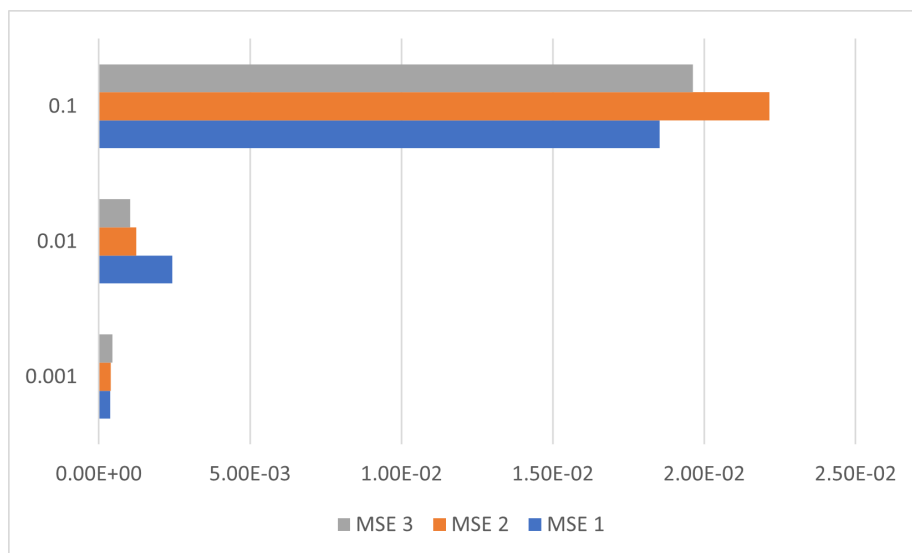


Figura G.11: Comparativa de ratio de aprendizaje para MLP

G.5. Comparativa entre modelos

Tras la comparativa de los resultados obtenidos con los diferentes valores de los hiperparámetros, se puede proceder de forma similar para los diferentes modelos implementados. De forma similar a la anterior, se realizan los promedios de los resultados obtenidos para cada valor individual de un parámetro concreto.

Como puede observarse en la Figura G.12, por término medio es el modelo *LSTM* quien obtiene los resultados más precisos generalmente para el número de capas del ensayo.

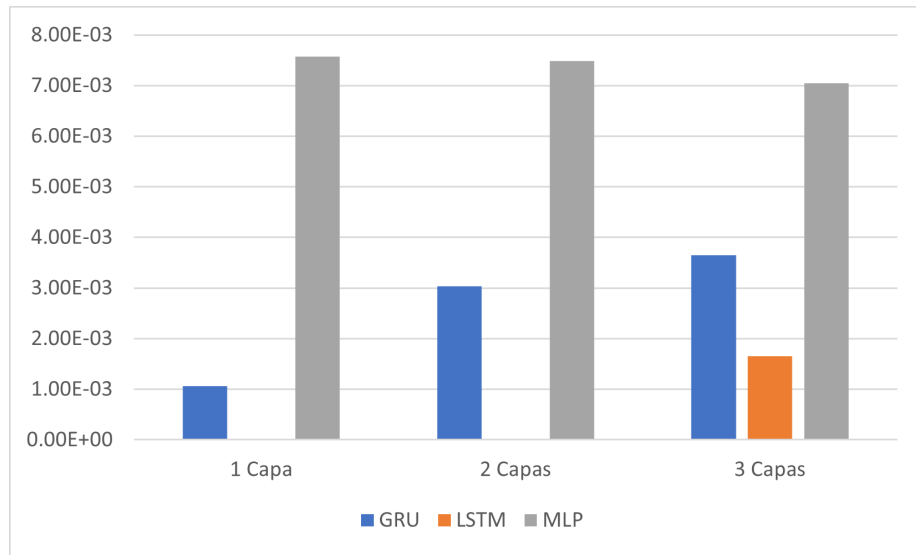


Figura G.12: Comparativa de número de capas

De forma similar a la anterior, se obtienen errores menores para el modelo mencionado, siendo las arquitecturas *LSTM* con las que se obtienen predicciones más precisas en promedio, destacando los modelos con 16 celdas de memoria en cada capa.

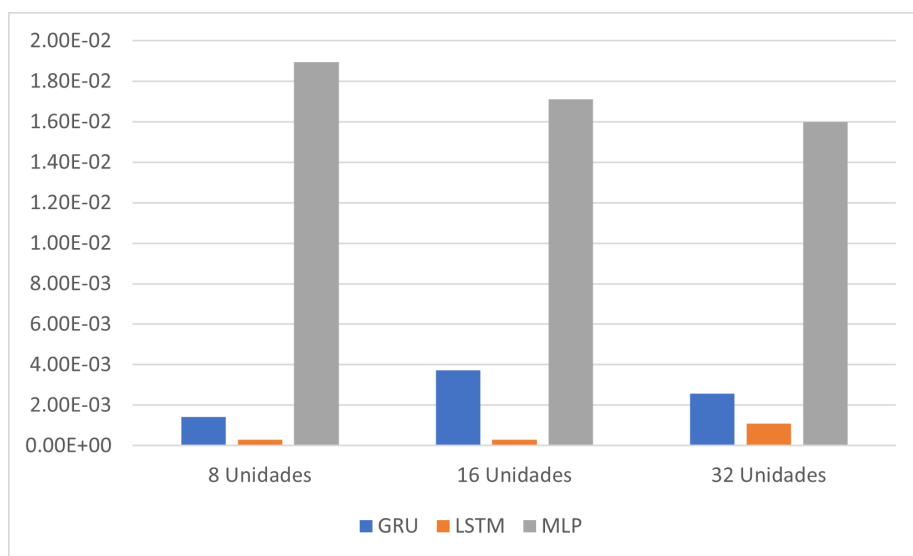


Figura G.13: Comparativa de número de unidades

En cuanto al ratio de aprendizaje, como se aprecia en la Figura G.14, el valor de 0.1 arroja los resultados con mayor error, siendo *MLP* las arquitecturas que por término medio obtendrán predicciones con menor precisión.

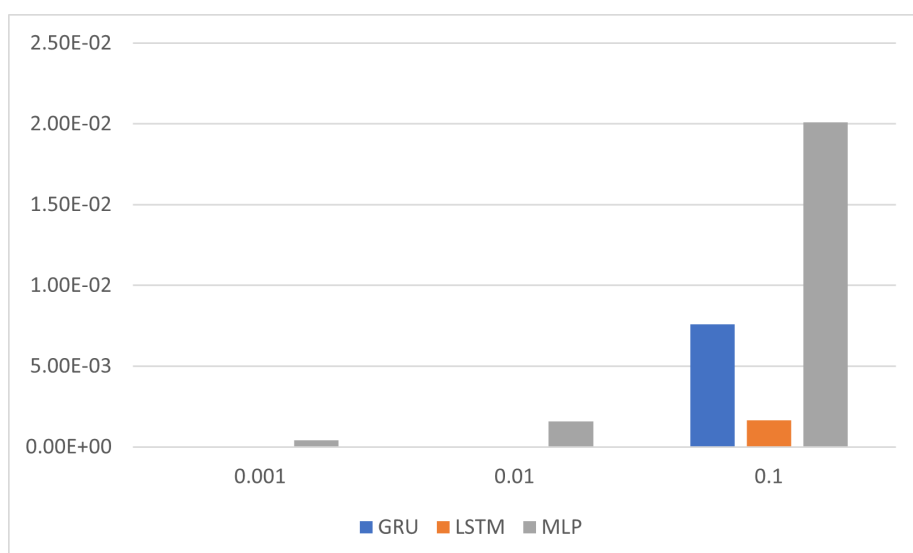


Figura G.14: Comparativa de ratio de aprendizaje

De este modo, como se puede observar en la Figura G.15, los resultados más precisos en promedio generalmente se obtienen con 0.01 de ratio de aprendizaje.

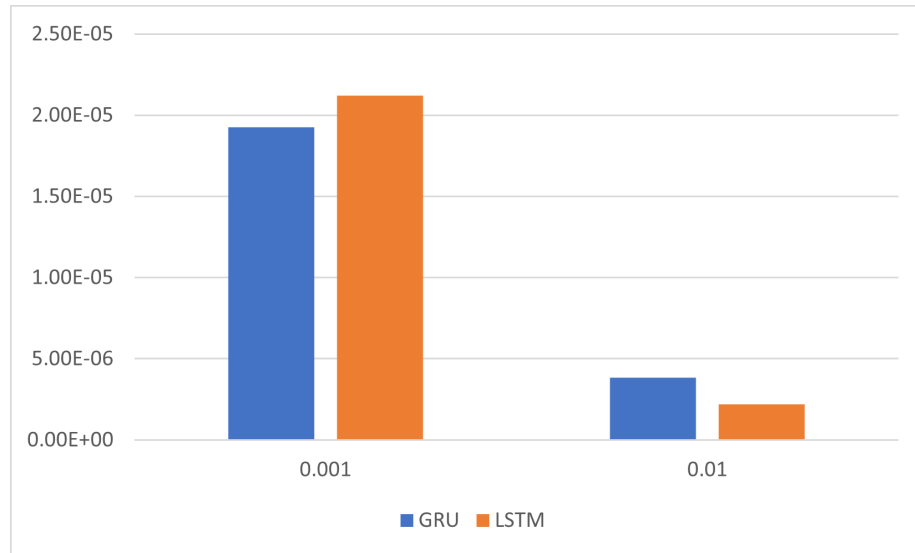


Figura G.15: Comparativa de ratio de aprendizaje ampliado

Apéndice H

Documentación técnica de programación

H.1. Introducción

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

H.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 (únicamente 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.

H.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** en el caso de realizar las ejecuciones con CPU.

Ejecución de los scripts mediante CPU

Para instalar el intérprete del lenguaje empleado en el proyecto se debe acudir a la página web oficial de los desarrolladores y descargar 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 utilizado 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 los pasos a seguir con el editor mencionado para habilitar los entornos, 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 lo permita. Para ello se debe abrir la terminal mencionada como administrador del sistema e introducir:

```
Set-ExecutionPolicy Unrestricted
```

Tras esto, se puede introducir el siguiente comando para iniciar el proceso de instalación:

```
./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 podrán ejecutar los diferentes ficheros fuente de Python Notebook abriendo el proyecto en Visual Studio Code y estableciendo el Kernel de ejecución al entorno configurado (Figura H.1). Está definido que el entorno virtual se denomine **.venv**, por lo que será necesario buscar entre los diferentes instalados haciendo clic en la parte superior derecha del notebook (en el botón para la selección del intérprete de ejecución).

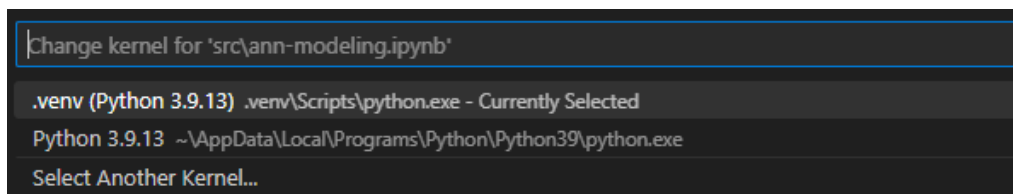


Figura H.1: Búsqueda del entorno virtual

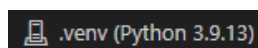


Figura H.2: Selección del entorno virtual

Posteriormente podrá ejecutarse cualquier fichero Python Notebook con el botón de “Execute All”, o proceder por celda.

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 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.

Para ejecutar los ficheros de Python adecuadamente es necesario estar situados con la terminal en el directorio correspondiente. En este caso se debe navegar hasta `./src/`

Ejecución acelerada mediante GPU: CUDA

Para disminuir el tiempo de entrenamiento de los modelos puede realizarse una instalación de un contenedor de Docker que emplea la tecnología CUDA (plataforma de computación paralela desarrollada por NVIDIA [11]) e incorpora las librerías necesarias de TensorFlow.

Para ello, es necesario disponer de una tarjeta gráfica de la marca mencionada compatible con una versión de CUDA mayor a 11.7, al mismo tiempo que actualizar sus *drivers* a la última versión estable disponible.

Para realizar la instalación en un sistema Windows es necesario disponer a su vez del WSL (Windows Subsystem for Linux) activo y funcionando con una distribución de Ubuntu. Se puede instalar la tecnología mencionada ejecutando la PowerShell en modo administrador e introduciendo:

```
wsl —install
```

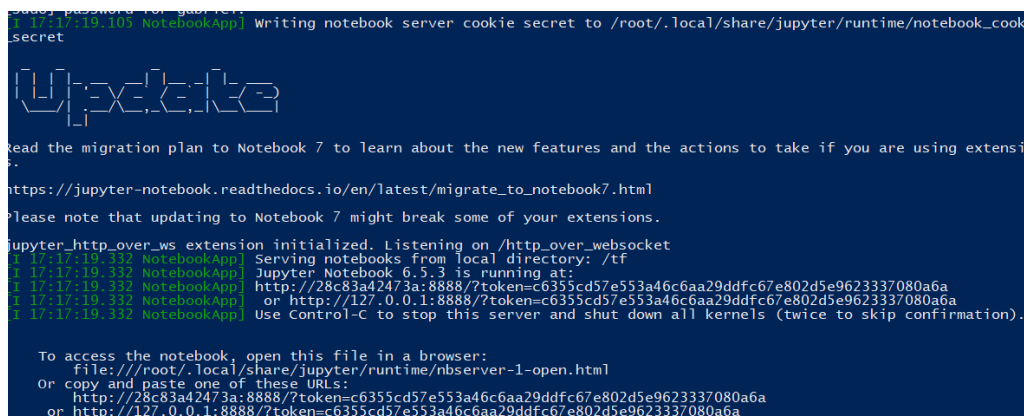
Al finalizar se deberá crear el entorno de Linux siguiendo los pasos indicados en la terminal para posteriormente establecer el usuario de WSL. Los siguientes pasos serán comunes.

Es necesario situarse en la carpeta principal del proyecto para poder realizar una correcta inicialización del contenedor. Si Docker ya se encuentra instalado en el sistema, únicamente se debe ejecutar el script **docker-create.sh** (se recomienda una instalación limpia para evitar posibles incompatibilidades), en caso contrario, previamente ejecutar **docker-install.sh**. Ambos con permisos de superusuario (empleando *sudo*).

Tras iniciar el contenedor se creará el servicio de Jupyter Notebook al que podremos acceder mediante el navegador con:

`https://localhost:8888/?token=<token>`

De esta forma, como se muestra en la Figura H.3, `<token>` podrá obtenerse del terminal en el que se encuentra corriendo el contenedor.



```

root@passmore:/opt/conda# jupyter notebook --ip 0.0.0.0 --port 8888
[17:17:19.103 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret

Jupyter Notebook

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions:
https://jupyter-notebook.readthedocs.io/en/latest/migrate_to_notebook7.html

Please note that updating to Notebook 7 might break some of your extensions.

Jupyter HTTP over WS extension initialized. Listening on /http_over_websocket
[17:17:19.332 NotebookApp] Serving notebooks from local directory: /tf
[17:17:19.332 NotebookApp] Jupyter Notebook 6.5.3 is running at:
[17:17:19.332 NotebookApp] http://28c83a42473a:8888/?token=c6355cd57e553a46c6aa29ddfc67e802d5e9623337080a6a
[17:17:19.332 NotebookApp] or http://127.0.0.1:8888/?token=c6355cd57e553a46c6aa29ddfc67e802d5e9623337080a6a
[17:17:19.332 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

To access the notebook, open this file in a browser:
file:///root/.local/share/jupyter/runtime/nbserver-1-open.html
Or copy and paste one of these URLs:
http://28c83a42473a:8888/?token=c6355cd57e553a46c6aa29ddfc67e802d5e9623337080a6a
or http://127.0.0.1:8888/?token=c6355cd57e553a46c6aa29ddfc67e802d5e9623337080a6a

```

Figura H.3: Ejecución del servicio de Jupyter Notebook con GPU

Tras esto, se puede acceder desde Jupyter a los ficheros `.ipynb` para ejecutarlos según la necesidad.

Ejecución acelerada mediante GPU: Google Colab

Si no se dispone de una tarjeta gráfica compatible con CUDA puede emplearse Google Colab para realizar la ejecución acelerada del entrenamiento. Para ello, se debe abrir el proyecto desde la plataforma mencionada, de forma que se dispongan de los datos en `./data/integrated_data` (pueden crearse para ello las estructuras de directorios necesarios) y el Python Notebook a ejecutar.

Debe cambiarse, por otro lado, el tipo de entorno de ejecución, seleccionando en **Entorno de ejecución > Seleccionar entorno de ejecución** la opción “GPU” en “Acelerador por hardware” como se muestra en la Figura H.4.

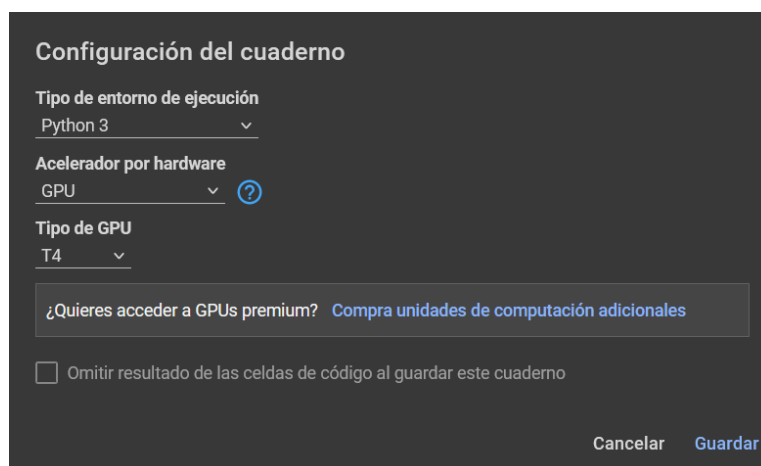


Figura H.4: Selección de GPU en Google Colab

H.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 con 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 correspondiente, además de las gráficas comparativas de los valores predichos y los reales por sensor y atributo, así como el error máximo total en cada característica para todos los sensores y las gráficas de los errores en el conjunto de test, de forma que se pueda dar una idea de los valores en los que ronda el error en cada una de las variables.

Apéndice I

Documentación de usuario

I.1. Introducción

En esta sección se presentará la manera 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 los datos de salida.

I.2. Requisitos de usuarios

En cuanto a los requisitos del usuario, será necesario tener instalado **Python 3.9.13** [8], junto con la versión 2.11.0 de **TensorFlow** [12], de manera que se puedan cargar los modelos almacenados empleando la función de la *API Keras* para tal propósito.

I.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 [13]:

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

I.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] Wikipedia, “Cuda — wikipedia, la enciclopedia libre,” 2023, [Internet; descargado 12-abril-2023]. [Online]. Available: <https://es.wikipedia.org/w/index.php?title=CUDA&oldid=150503522>
- [12] “Tensorflow.” [Online]. Available: <https://www.tensorflow.org/?hl=es-419>
- [13] “Guardando y serializando modelos con tensorflow keras | tensorflow core.” [Online]. Available: https://www.tensorflow.org/guide/keras/save_and_serialize?hl=es-419