UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

título del TFG
Documentación Técnica

Presentado por nombre alumno
en Universidad de Burgos — 25 de mayo de 2023
Tutor: nombre tutor

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 [1], 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 [2]) 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)

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

Una muestra de cómo podría ser una tabla de casos de uso:

B.2. Objetivos generales

B.3. Catalogo de requisitos

B.4. Especificación de requisitos

CU-1	**Ejemplo de caso de uso**
Versión	1.0
Autor	Alumno
Requisitos asociados	RF-xx, RF-xx
Descripción	La descripción del CU
Precondición	Precondiciones (podría haber más de una)
Acciones	1. Pasos del CU 2. Pasos del CU (añadir tantos como sean necesarios)
Postcondición	Postcondiciones (podría haber más de una)
Excepciones	Excepciones
Importancia	Alta o Media o Baja...

Tabla B.1: CU-1 Nombre del caso de uso.

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.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.

D.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.

D.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 [3]. 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 [4], sin embargo para su ejecución podemos emplear otros entornos como Anaconda Navigator [5]. 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).

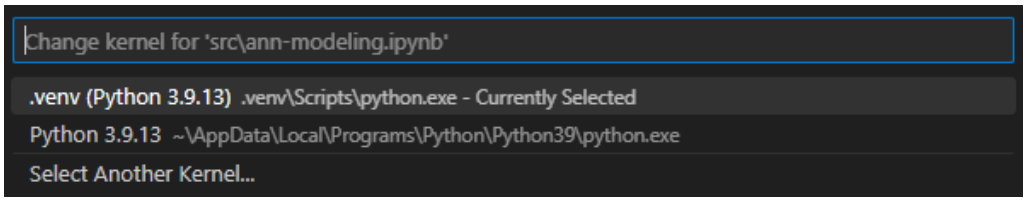

Figura D.1: Búsqueda del entorno virtual

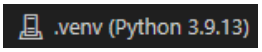

Figura D.2: Selección del entorno virtual

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.

D.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 E

Documentación de usuario

E.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.

E.2. Requisitos de usuarios

En cuanto a los requisitos del usuario, se deberá tener instalado **Python 3.9.13** [3], junto con la versión 2.11.0 de **TensorFlow** [6], de forma que se puedan cargar los modelos almacenados empleando la función de la *API* **Keras** correspondiente.

E.3. Instalación

En cuanto a la instalación de la versión concreta del intérprete de Python puede realizarse en [3], 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 [7]:

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

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

[2] [Online]. Available: https://www.zenhub.com/

[3] [Online]. Available: https://www.python.org/downloads/release/python-3913/

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

[5] [Online]. Available: https://anaconda.org/anaconda/anaconda-navigator

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

[7] [Online]. Available: https://www.tensorflow.org/guide/keras/save_and_serialize?hl=es-419