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 — 24 de mayo de 2023
Tutor: nombre tutor

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

A.2. Planificación temporal

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 [1]. 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 [2], sin embargo para su ejecución podemos emplear otros entornos como Anaconda Navigator [3]. 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).

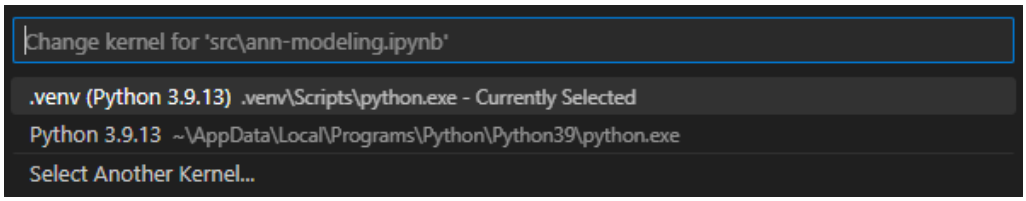

Figura D.1: Búsqueda del entorno virtual

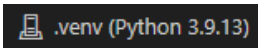

Figura D.2: Selección del entorno virtual

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** [1], junto con la versión 2.11.0 de **TensorFlow** [4], 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 [1], mientras que para instalar la dependencia concreta 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 [5]:

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

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

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

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

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