



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**título del TFG
Documentación Técnica**



Presentado por José Luis Pérez Gómez
en Universidad de Burgos — 4 de julio de 2024
Tutor: Bruno Baruque Zanón - Jesús Enrique
Sierra García

Índice general

| | |
|---|------------|
| Índice general | i |
| Índice de figuras | iii |
| Índice de tablas | iv |
| Apéndice A Plan de Proyecto Software | 1 |
| A.1. Introducción | 1 |
| A.2. Planificación temporal | 2 |
| A.3. Estudio de viabilidad | 7 |
| Apéndice B Especificación de Requisitos | 9 |
| B.1. Introducción | 9 |
| B.2. Objetivos generales | 9 |
| B.3. Catálogo de requisitos | 10 |
| B.4. Especificación de requisitos | 12 |
| Apéndice C Especificación de diseño | 21 |
| C.1. Introducción | 21 |
| C.2. Diseño de datos | 23 |
| C.3. Diseño procedimental | 23 |
| C.4. Diseño arquitectónico | 25 |
| Apéndice D Documentación técnica de programación | 27 |
| D.1. Introducción | 27 |
| D.2. Estructura de directorios | 27 |
| D.3. Manual del programador | 28 |

| | |
|--|-----------|
| D.4. Compilación, instalación y ejecución del proyecto | 31 |
| D.5. Pruebas del sistema | 33 |
| Apéndice E Documentación de usuario | 35 |
| E.1. Introducción | 35 |
| E.2. Requisitos de usuarios | 35 |
| E.3. Instalación | 37 |
| E.4. Manual del usuario | 38 |
| Apéndice F Anexo de sostenibilización curricular | 51 |
| F.1. Introducción | 51 |

Índice de figuras

| | |
|--|----|
| A.1. Commits realizados durante la realización del TFG | 2 |
| A.2. Additions realizados durante la realización del TFG | 3 |
| A.3. Deletions realizados durante la realización del TFG | 3 |
| A.4. Commits-Etapa 1 | 4 |
| A.5. Commits-Etapa 2 | 5 |
| A.6. Commits-Etapa 3 | 6 |
| C.1. Cantidad de elementos para la característica Key | 22 |
| C.2. Análisis característica Timestamp del conjunto de datos | 23 |
| D.1. Descarga repositorio GitHub | 30 |
| D.2. Listado de Notebooks en Jupyter Notebook | 32 |
| E.1. Descarga repositorio GitHub | 37 |
| E.2. Listado de Notebooks en Jupyter Notebook | 38 |
| E.3. Ejemplo Notebook Main | 40 |
| E.4. Botón Upload | 41 |
| E.5. Subida correcta de archivo csv. | 41 |
| E.6. Datos después de subirlos al entorno correctamente. | 42 |
| E.7. Ejemplo instalación automática | 44 |
| E.8. Ejemplo instalación manual | 44 |
| E.9. Ejemplo instalación bibliotecas | 45 |
| E.10. Ejemplo declaraciones a importar bibliotecas | 46 |
| E.11. Ejemplo variables declaradas | 47 |
| E.12. Ejemplo gráfica en análisis de datos | 48 |
| E.13. Ejemplo tabla comparativa de tasa de acierto entre modelos | 49 |
| E.14. Ejemplo matriz de confusión para un modelo | 49 |

Índice de tablas

| | |
|---|----|
| B.1. CU-001 Carga archivo con datos EEG. | 13 |
| B.2. CU-002 Preprocesamiento de Datos EEG. | 13 |
| B.3. CU-003 Visualización de datos. | 14 |
| B.4. CU-004 Entrenamiento y validación de modelos de aprendizaje automático básico. | 15 |
| B.5. CU-005 Entrenamiento y validación de modelos de aprendizaje automático neuronal | 16 |
| B.6. CU-006 Control de BCI | 17 |
| B.7. CU-007 Almacenamiento y Recuperación de Resultados de Ex- perimentos | 18 |
| B.8. CU-008 Integración y Extensibilidad del sistema | 19 |

Apéndice A

Plan de Proyecto Software

A.1. Introducción

La Universidad de Burgos, dentro del área de conocimiento de Ingeniería de Sistemas y Automática, dispone de un interfaz BCI (Brain Computer Interface) para la captación de señales cerebrales. Empleando ese interfaz se han realizado diferentes experimentos que han permitido recoger información de la actividad cerebral mientras los usuarios ejecutaban diferentes tareas cotidianas.

Este Trabajo de Fin de Grado (TFG) tiene como objetivo el análisis de la información obtenida en esos experimentos. Se entrenarán diferentes algoritmos para clasificar la acción realizada por el usuario a partir de las señales generadas por el BCI. Con este propósito, se evaluarán diferentes algoritmos de procesamiento de señales y de machine/deep learning para la clasificación automática de señales.

Los datos aportados son de tipo EEG (Electroencefalografía) para la realización del TFG son datos referentes a experimentos basados en pulsaciones sobre teclas de un teclado: arriba, abajo, izquierda, derecha.

El análisis de estos datos y su evaluación en diferentes algoritmos esta basada en predecir qué teclas del teclado se han pulsado según las señales captadas con la interfaz BCI.

Para esto no ha habido una planificación como tal registrada en Github, pero sí una progresión definida en los commits de código generados durante la composición del TFG.

A.2. Planificación temporal

En la reunión inicial con los tutores definimos utilizar Python para realizar el código y una serie de aprendizajes básicos para poder acometer el TFG sin problemas.

En las siguientes reuniones se definieron algoritmos y experimentos a realizar con los datos EGG aportados.

Desde el principio del proyecto debido a las circunstancias personales del alumno no se ha podido realizar metodología Scrum, pero sí se produjeron estas líneas temporales:

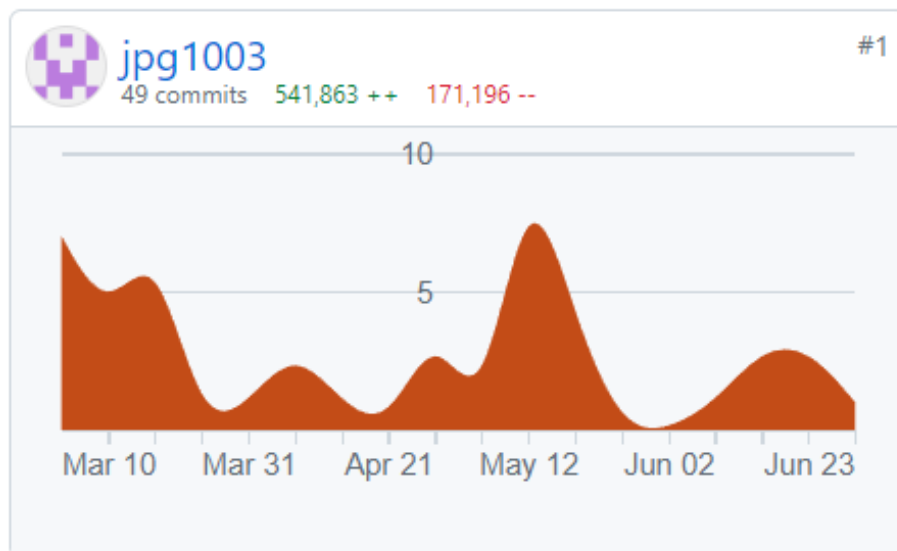


Figura A.1: Commits realizados durante la realización del TFG

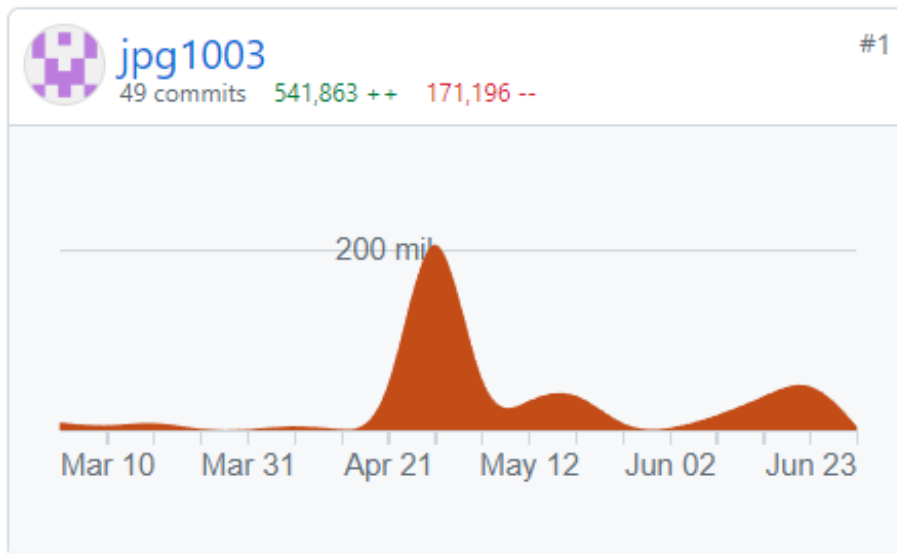


Figura A.2: Additions realizados durante la realización del TFG



Figura A.3: Deletions realizados durante la realización del TFG

Se pueden dividir en 3 grandes etapas:

- Etapa de estudio: (Febrero a Marzo)
- Etapa de desarrollo: (Abril a Mayo)
- Etapa de desarrollo y optimización: (Junio a Julio)

Etapa de estudio: (Febrero a Marzo)

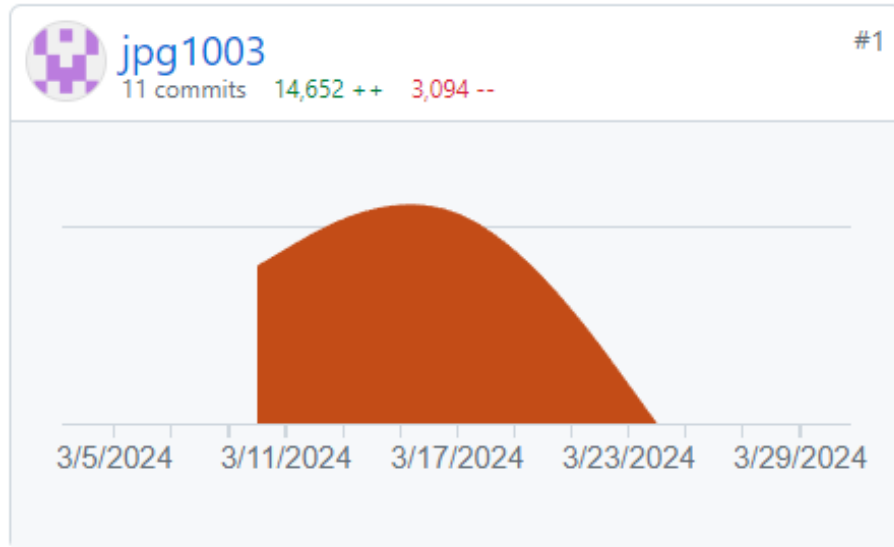


Figura A.4: Commits-Etapa 1

- **Realización de cursos online** propuestos por los tutores Bruno Baruque y Jesús Enrique Sierra. [Tutorial de Inicio de Pandas](#), [Visualización de Datos](#), [Trabajo con series temporales](#)

- **Análisis del conjunto de datos**, en archivo csv, proporcionado por los tutores.

- **Creación esqueleto para la estructura el TFG en Github.**

- **Definición y creación de los primeros notebooks**, principalmente análisis del conjunto de datos.

- **Definición y creación de los primeros notebooks de machine learning**, al final de la etapa.

- **Comentar código e imprimir comentarios en los notebooks.**

Los principales problemas o obstáculos que me encontré fueron principalmente los siguientes:

- **Tiempo invertido en la realización de los cursos.** - **Estructuración del código.** Me tomo mucho tiempo poder llegar a definir como quería mostrar el código, me decidí por un notebook principal que realizara llamadas al resto de notebooks con códigos mas específicos para cada mo-

delo o experimento a realizar. - **Plotteos y definiciones básicas como normalizar o escalar el conjunto de datos.**

Etapas de desarrollo: (Abril a Mayo)

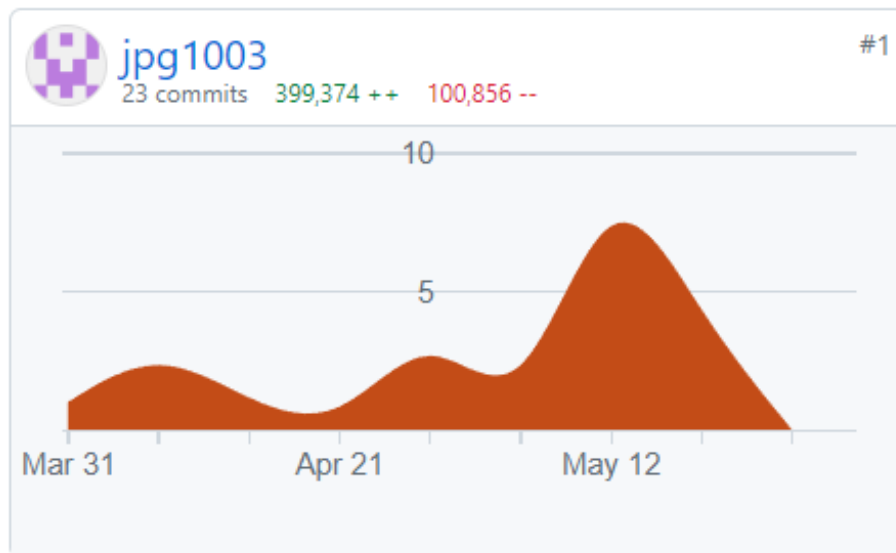


Figura A.5: Commits-Etapa 2

- Cambio de análisis del conjunto de datos, preprocessing.
- Definición y creación notebooks de machine learning.
- Definición y creación notebooks de deep learning.

Los problemas que me encontré en esta etapa fueron:

- **Compilación modelos deep learning.** Al ejecutar los primeros modelos de deep learning los datos normalizados me proporcionaban errores de compilación con modelos SRNN o LSTM, cambiando a datos escalados y shapeando los modelos pudieron ejecutarse.
- **Utilización de ventanas temporales en los modelos deep learning.** En esta etapa no supe identificar este requerimiento por parte de los tutores y estuve implementando varias formas de poder utilizar ventanas temporales en el código.
- Utilización de modelos deep learning y callbacks.
- Gráficas y definiciones básicas como normalizar o escalar el conjunto de datos.

Etapa de desarrollo y optimización: (Junio a Julio)

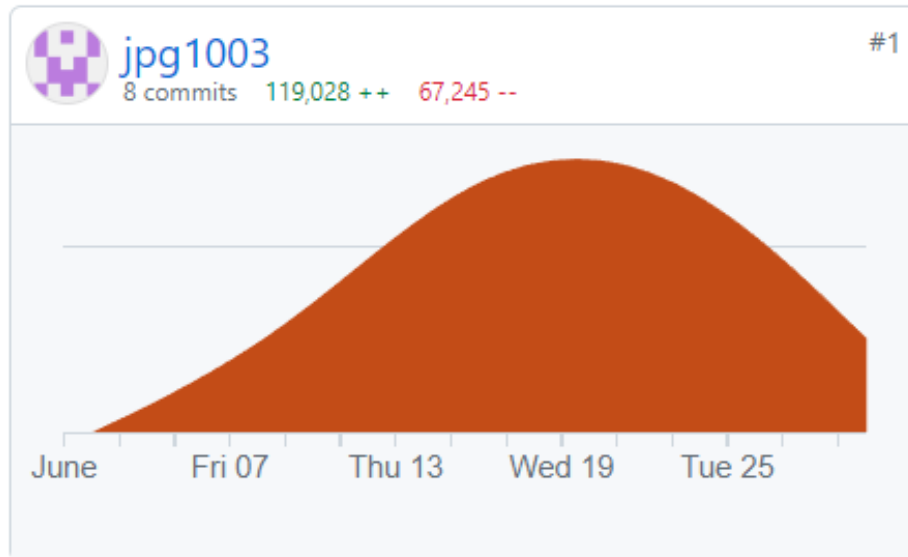


Figura A.6: Commits-Etapa 3

- **Añado nuevos gráficos** en análisis del conjunto de datos, preprocesing.
- **Definición y creación ventanas temporales acordadas con Bruno Baruque.**
- **Definición y creación nuevos notebooks de deep learning**
- **Definición y creación nuevos datos sintéticos a través de el aplicativo smote.**
- **Continuar con el comentado del código e imprimir comentarios en los notebooks.**

En la última etapa los problemas que se han acontecido son:

- **Utilización de ventanas temporales en los modelos deep learning.** Después de varias algunas reuniones con Bruno Baruque se llego a la defnicon correcta para las ventanas temporal en el conjunto de datos.
- **Utilización de modelos deep learning y callbacks.**
- **Utilización datos sintéticos.** Definir correctamente esta generacion de datos y poder utilizarlos en los modelos deep learning correctamente.

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

La Especificación de Requisitos tiene como objetivo definir de manera clara y detallada las necesidades y expectativas del proyecto que consta de un notebook Jupyter definido en Python, el cual invoca a otros notebooks Jupyter secundarios para realizar diversas tareas.

Este documento sirve para asegurar que todas las partes interesadas comprendan y acuerden los objetivos y funcionalidades del sistema a desarrollar.

En un entorno de trabajo basado en notebooks Jupyter, es esencial contar con una especificación precisa que guíe el desarrollo e implementación. Esto facilita la colaboración y la comunicación entre los desarrolladores y usuarios finales.

B.2. Objetivos generales

Los objetivos principales de este TFG son los siguientes:

- **Facilitar el procesamiento de datos EEG.** Generados en la universidad mediante el interfaz BCI.
- **Automatizar tareas repetitivas.** Con la creación de notebooks secundarios para poder ejecutar las tareas comunes y repetitivas relacionadas con los datos EEG.
- **Mejorar la tasa de acierto en los análisis de modelos** implementando modelos precisos y fiables.

- **Asegurar la accesibilidad y usabilidad en los notebooks.** Los notebooks son herramientas intuitivas y accesibles que aportan esa facilidad a la hora de interactuar.
- **Integración y extensibilidad.** Al utilizar llamadas a otros notebooks se asegura que se permita la integración de nuevos notebooks y se asegura posibles nuevas mejoras en el código.

B.3. Catálogo de requisitos

Hay dos tipos de requisitos, los funcionales (qué debe hacer el código) y los no funcionales (cómo debe funcionar el código):

Requisitos funcionales

- **RF-001 Cargar archivos con datos EEG:**
 - **Descripción:** El sistema debe permitir cargar archivos de datos EEG para su posterior análisis.
 - **Prioridad:** Alta
 - **Criterios de aceptación:** El sistema debe permitir al usuario poder subir archivos con datos EEG manualmente.
- **RF-002 Preprocesar datos:**
 - **Descripción:** El sistema debe preprocesar los datos EEG cargados de manera manual, estandarizar, escalar, eliminar outliers,... para su posterior análisis. .
 - **Prioridad:** Alta
 - **Criterios de aceptación:** Se ha de poder comprobar que el sistema ha realizado el preprocesado de los datos EEG.
- **RF-003 Visualizar y analizar datos:**
 - **Descripción:** El sistema debe permitir la visualización gráfica de las señales EEG antes y después de ser preprocesadas.
 - **Prioridad:** Alta
 - **Criterios de aceptación:** Se ha de poder comparar los datos EEG antes y después del preprocesado.
- **RF-004 Modelado y validación aprendizaje automático básico:**
 - **Descripción:** El sistema debe permitir entrenar modelos de aprendizaje básicos (KNN, Árboles de decisión, random forest)

con diferentes técnicas de validación (Hold-Out, K-Fold, Cross-Validation, Leave-One-Out Cross-Validation) utilizando los datos EEG preprocesados.

- **Prioridad:** Alta
- **Criterios de aceptación:** .

■ **RF-005 Modelado y validación aprendizaje automático neuronal:**

- **Descripción:** El sistema debe permitir entrenar modelos de aprendizaje automático (MLP, SRNN, LSTM) utilizando los datos EEG preprocesados.
- **Prioridad:** Alta
- **Criterios de aceptación:** .

■ **RF-006 Control de BCI:**

- **Descripción:** El sistema debe ser capaz de identificar o clasificar con precisión las señales EEG que correspondan a una de las direcciones especificadas (arriba, abajo, izquierda, derecha).
- **Prioridad:** Alta
- **Criterios de aceptación:** .

■ **RF-007 Almacenamiento y recuperación de resultados:**

- **Descripción:** El sistema debe permitir almacenar los resultados de los experimentos y sus resultados.
- **Prioridad:** Alta
- **Criterios de aceptación:** .

■ **RF-008 Integración y Extensibilidad:**

- **Descripción:** El sistema debe permitir la integración de nuevos modelos de aprendizaje automático sin necesidad de reestructurar significativamente el código existente.
- **Prioridad:** Alta
- **Criterios de aceptación:** .

Requisitos no funcionales

■ **RNF-001 Rendimiento:**

- **Descripción:** El sistema donde se ejecuten los notebooks ha de poder manejar la ejecución sin causar demoras significativas.

- **Criterios de aceptación:** El tiempo de ejecución para la ejecución de cada notebook secundario no ha de ser superior a 15 minutos.
- **RNF-002 Usabilidad:**
 - **Descripción:** Los notebooks han de ser fáciles de entender y usar por los usuarios.
 - **Criterios de aceptación:** La interfaz de los notebooks ha de ser intuitiva y clara.
- **RNF-003 Escalabilidad:**
 - **Descripción:** En la configuración del notebooks principal se debe permitir la adhesión de nuevos notebooks y sus llamadas desde el notebook principal.
 - **Criterios de aceptación:** Poder seguir integrando nuevos notebooks para ser llamados desde el notebook principal sin afectar al funcionamiento del código.

Restricciones

- **R-001:** El sistema debe operar en un entorno Jupyter Notebook.
- **R-002:** Todos los notebooks secundarios deben estar disponibles en el mismo entorno de ejecución que el notebook principal.
- **R-003:** El procesamiento y análisis de datos debe realizarse utilizando bibliotecas compatibles con Python 3.x.

B.4. Especificación de requisitos

Voy a describir cada caso de uso identificado:

| CU-001 | Carga archivo con datos EEG |
|-----------------------------|---|
| Actor | Investigador |
| Versión | 1.0 |
| Autor | José Luis Pérez Gómez |
| Requisitos asociados | RF-001 |
| Descripción | El investigador carga el archivo de datos EEG para su posterior análisis. |
| Precondición | R-001, R-002, R-003 |
| Acciones | <ol style="list-style-type: none"> 1. El usuario siguiendo los pasos indicados desde el sistema, carga el archivo con datos EEG que quiere analizar. |
| Postcondición | El usuario podrá visualizar los datos cargados en el sistema |
| Excepciones | |
| Importancia | Alta |

Tabla B.1: CU-001 Carga archivo con datos EEG.

| CU-002 | Preprocesamiento de Datos EEG |
|-----------------------------|---|
| Actor | Investigador |
| Versión | 1.0 |
| Autor | José Luis Pérez Gómez |
| Requisitos asociados | RF-002 |
| Descripción | El investigador ejecuta el preprocesado de los datos EEG. |
| Precondición | R-001, R-002, R-003 |
| Acciones | El sistema aplica técnicas de preprocesamiento para limpiar los datos, estandarizar, etc. |
| Postcondición | La salida de la ejecución no tenga ningún error |
| Excepciones | |
| Importancia | Alta |

Tabla B.2: CU-002 Preprocesamiento de Datos EEG.

| | |
|-----------------------------|---|
| Actor | Investigador |
| CU-003 | Visualización de datos. |
| Versión | 1.0 |
| Autor | José Luis Pérez Gómez |
| Requisitos asociados | RF-003 |
| Descripción | El sistema genera visualizaciones gráficas de las señales EEG antes y después del preprocesado. |
| Precondición | R-001, R-002, R-003 |
| Acciones | <ol style="list-style-type: none"> 1. El sistema aplica técnicas de impresión por pantalla de los datos a analizar." |
| Postcondición | Visualizaciones gráficas de los datos EEG. |
| Excepciones | |
| Importancia | Alta |

Tabla B.3: CU-003 Visualización de datos.

| | |
|-----------------------------|---|
| CU-004 | Entrenamiento y validación de modelos de aprendizaje automático básico |
| Actor | Investigador |
| Versión | 1.0 |
| Autor | José Luis Pérez Gómez |
| Requisitos asociados | RF-004 |
| Descripción | El investigador entrena y valida modelos de aprendizaje automático básico utilizando diferentes técnicas de validación con los datos preprocesados EEG. |
| Precondición | R-001, R-002, R-003 |
| Acciones | <ol style="list-style-type: none"> 1. El sistema entrena y valida los modelos aprendizaje automático básico con los datos preprocesados EEG. |
| Postcondición | Modelos entrenados y validados con sus respectivas métricas de tasa de acierto y matrices de confusión. |
| Excepciones | |
| Importancia | Alta |

Tabla B.4: CU-004 Entrenamiento y validación de modelos de aprendizaje automático básico.

| CU-005 | Entrenamiento y validación de modelos de aprendizaje automático neuronal |
|-----------------------------|---|
| Actor | Investigador |
| Versión | 1.0 |
| Autor | José Luis Pérez Gómez |
| Requisitos asociados | RF-005 |
| Descripción | El investigador entrena y valida modelos de deep learning utilizando los datos EEG preprocesados. |
| Precondición | R-001, R-002, R-003, haber iniciado correctamente el notebook principal (Main) y haber ejecutado todas las celdas anteriores a esta celda |
| Acciones | <ol style="list-style-type: none"> 1. El sistema entrena y valida los modelos aprendizaje automático neuronal con los datos preprocesados EEG. |
| Postcondición | Modelos entrenados y validados con sus respectivas métricas de tasa de acierto y matrices de confusión. |
| Excepciones | |
| Importancia | Alta |

Tabla B.5: CU-005 Entrenamiento y validación de modelos de aprendizaje automático neuronal

| | |
|-------------------------------|---|
| Actor CU-006 | Investigador Control de BCI |
| Versión | 1.0 |
| Autor | José Luis Pérez Gómez |
| Requisitos asociados | RF-006 |
| Descripción | El sistema debe clasificar las señales EEG para determinar la dirección de movimiento (arriba, abajo, izquierda, derecha). |
| Precondición | R-001, R-002, R-003 |
| Acciones | <ol style="list-style-type: none"> 1. El sistema tras el entrenamiento y validacion de los diferentes modelos debe ser capaz de identificar o clasificar con precisión las señales EEG que correspondan a una de las direcciones especificadas (arriba, abajo, izquierda, derecha) |
| Postcondición | |
| Excepciones | |
| Importancia | Alta |

Tabla B.6: CU-006 Control de BCI

| | |
|-------------------------------|---|
| Actor CU-007 | Investigador Almacenamiento y Recuperación de Resultados de Experimentos |
| Versión | 1.0 |
| Autor | José Luis Pérez Gómez |
| Requisitos asociados | RF-007 |
| Descripción | : El sistema almacena y recupera los resultados de los experimentos de análisis de datos EEG. |
| Precondición | R-001, R-002, R-003 |
| Acciones | <ol style="list-style-type: none"> 1. El investigador solicita la recuperación de resultados de los experimentos realizados y el sistema muestra los resultados de los experimentos. |
| Postcondición | Resultados de los experimentos almacenados y recuperados para análisis comparativo posterior. |
| Excepciones | |
| Importancia | Alta |

Tabla B.7: CU-007 Almacenamiento y Recuperación de Resultados de Experimentos

| | |
|-----------------------------|---|
| Actor | Desarrollador |
| CU-008 | Integración y Extensibilidad del sistema |
| Versión | 1.0 |
| Autor | José Luis Pérez Gómez |
| Requisitos asociados | RF-008 |
| Descripción | : El desarrollador integra, modifica y extiende todos los modelos de aprendizaje automático en el sistema. |
| Precondición | R-001, R-002, R-003 |
| Acciones | <ol style="list-style-type: none"> 1. El desarrollador accede al código fuente del sistema, implementa el modelos, actualiza la interfaz de usuario para permitir la selección de los modelos. |
| Postcondición | Modelos integrados y disponibles para experimentos de análisis de datos EEG. |
| Excepciones | |
| Importancia | Alta |

Tabla B.8: CU-008 Integración y Extensibilidad del sistema

Apéndice C

Especificación de diseño

C.1. Introducción

La especificación de datos es un componente crítico en el desarrollo de sistemas de información, especialmente cuando se trabaja con conjuntos de datos complejos como los datos EEG.

Mediante BCI, sistema que permite mediante adquisición de señales EEG, se han podido interpretar las señales adquiridas a través de las señales EEG y poder transformarlas en un conjunto de datos para su posterior análisis.

Descripción de los Datos:

El archivo datosEEGTotal.csv contiene los datos facilitados para poder realizar los experimentos para la ejecución del TFG.

Su formato es de tipo CSV con un separador (;) entre los datos que lo componen.

Las características recogidas en el archivo de los datos EEG son las siguientes:

Timestamp, Attention, Meditation, Delta, Theta, LowAlpha, HighAlpha, LowBeta, HighBeta, LowGamma, HighGamma, Signal y Key.

Timestamp: Registro de tiempo para los experimentos, medido en mili-segundos.

Attention: Registra el grado de atención del participante que realiza el experimento.

Meditation: Grado de calma que tendría el individuo.

Delta: Son ondas de baja frecuencia (1 y 4 Hz), están presentes en etapas de sueño profundo, durante una meditación profunda y en pacientes con lesiones cerebrales o con TDAH severo.

Theta: Ondas entre 4 y 8 Hz, se encuentran en estados de calma profunda y sueño R.E.M., están ligadas al aprendizaje, memoria y intuición.

Alpha: Ondas entre 8 y 12 Hz, representan un estado de poca actividad cerebral y se asocian a un estado de calma mental. Divididas en dos señales LowAlpha y HighAlpha

Beta: Se diferencian en LowBeta y HighBeta, su frecuencia esta entre 12 y 35Hz, asociadas a una alta actividad mental.

Gamma: En los datos se diferencia LowGamma y HighGamma, son ondas por encima de 30Hz y suelen aparecer cuando hay una alta concentración o atención

Signal: Podría ser la señal de que aporta la interfaz BCI.

Key: Valores target de lo que el individuo estaba pensando o visualizando durante el experimento.

La transformación de datos o el preprocessing que se ha realizado para poder afrontar los experimentos del TFG han sido los siguientes:

Unificación de características Key: El conjunto de datos tiene varios valores en Key que indican los mismo. LButton y Left.

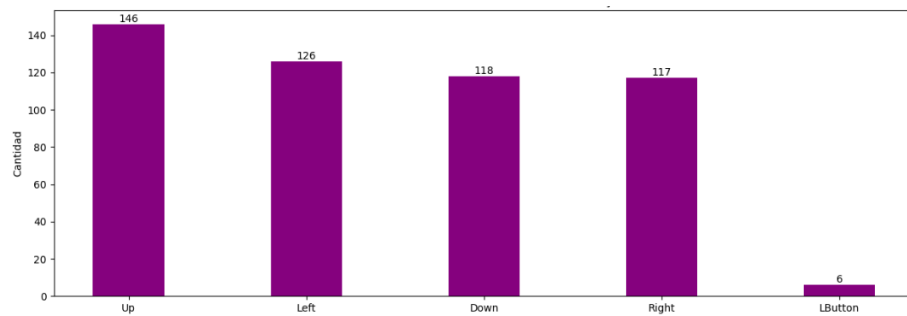


Figura C.1: Cantidad de elementos para la característica Key

División del conjunto de datos: El conjunto de datos tiene cuatro segmentos divididos por su Timestamp, superponiéndose entre ellos. Divido en estos cuatro segmentos para poder realizar experimentos y también de

el conjunto de datos sin dividir para realizar experimentos conjuntos a los cuatro segmentos.

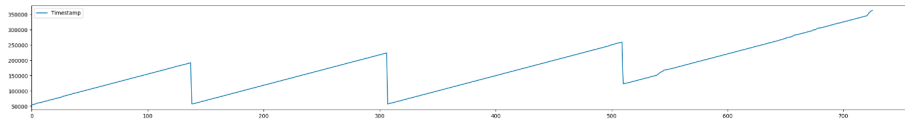


Figura C.2: Análisis característica Timestamp del conjunto de datos

Eliminación de características: Elimino las características Signal por no aportar nada significativo en el conjunto de datos y Timestamp porque no quiero que los datos aportados puedan tener una patrón temporal que haga que los experimentos no sean reales.

Eliminación de outliers: Elimino los outliers (datos atípicos) del conjunto de datos. Utilizo zcore con un umbral de 3. **Regla del 68-95-99.7**

Escalado de datos: He utilizado la opción de escalar los datos ya que no son datos normales puesto que no tienen una distribución gaussiana en sus datos.

C.2. Diseño de datos

En este trabajo con datos EEG, el diseño de datos consta de:

Diagramas entidad-relación (ERD)

Relaciones y Dependencias:

C.3. Diseño procedimental

En este trabajo he utilizado notebooks Jupyter para analizar el conjunto de datos EEG, por este motivo el diseño arquitectónico resultante ha sido el siguiente:

Arquitectura del Sistema:

La idea principal era que la ejecución de varios modelos de machine y deep learning se pudieran ejecutar de una manera eficiente y que fueran independientes sus ejecuciones. Para esto, el diseño se compone de un notebook jupyter que llama celda a celda a diferentes notebooks en los cuales se desarrolla el código a ejecutar.

Los componentes principales son:

- **Notebook principal**, desde el cual se orquesta todas las ejecuciones de los demás notebooks.
- **Notebooks secundarios preparatorios**, que como su nombre indica, preparan el entorno para que puedan ser ejecutados los experimentos.
- **Notebook subida datos en archivo CSV**, desde esta llamada se sube el archivo con el conjunto de datos a analizar.
- **Notebook secundarios para experimentos** donde se ejecutan los experimentos asociados al TFG.
- **Notebook de resultados** que imprime los resultados de tasa de acierto en los experimentos contra los datos de tipos test.

Las tecnologías y Herramientas utilizadas han sido las siguientes:

Manejo de datos y preprocessing:

- **Pandas**: Para la manipulación y análisis de datos estructurados en DataFrames.
- **NumPy**: Para operaciones numéricas eficientes en matrices, arrays y listas.
- **Scikit-learn y SciPy**: Para escalado de datos, transformación y técnicas de preprocesamiento como StandardScaler y z-score.

Técnicas de validación:

- **Holdout**: División de los datos en conjuntos de entrenamiento, validación y prueba para evaluación inicial.
- **K-Fold Cross-Validation**: Validación cruzada con KFold para evaluar la variabilidad del modelo.
- **Leave-One-Out**: Validación de uno en uno los datos dejando los demás fuera, así se evalúa la robustez del modelo.

Modelos de aprendizaje automático:

- **K-Nearest Neighbors (KNN)**: Clasificación basada en la proximidad de los puntos de los datos.
- **Árboles de decisión (Decision Tree)**: Utiliza una estructura de árbol para realizar predicciones basadas en decisiones binarias.
- **Random Forest**: Mejora la precisión y reduce el sobreajuste utilizando múltiples árboles de decisión.

Redes Neuronales y Modelos Avanzados:

- **Multi-Layer Perceptron (MLP):** Implementan redes neuronales feedforward. Esta arquitectura de red neuronal no tiene ciclos en sus conexiones neuronales sino que los datos van en una dirección desde la capa de entrada a la capa de salida. Es el más básico.
- **Simple Recurrent Neural Network (SRNN):** Es un tipo básico de red neuronal que tiene conexiones recurrentes que le permite mantener y utilizar información previa en la secuencia de datos. Es muy parecida a MLP pero en este caso las conexiones entre neuronas si pueden ir hacia atrás.
- **Long Short-Term Memory (LSTM):** Es una red neuronal avanzada en comparación con la recurrente SRNN. La diferencia principal es que puede retener la información relevante durante mas tiempo que SRNN.

Técnicas para la creación de datos sintéticos:

- **SMOTE (Synthetic Minority Over-sampling Technique):** Uso de la biblioteca imblearn para crear ejemplos sintéticos.

Infraestructura utilizada?

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.1. Introducción

Este TFG de análisis de un conjunto de datos EEG se centra en la aplicación de técnicas avanzadas de aprendizaje automático utilizando notebooks Python. El objetivo principal es procesar y analizar datos EEG para identificar patrones y correlaciones significativas que puedan revelar comportamientos específicos en la actividad cerebral a la hora de realizar un experimento el individuo.

El propósito fundamental es aplicar modelos de aprendizaje automático para mejorar la comprensión de las señales EEG, los datos aportados por la universidad de burgos constan de unos experimentos realizados con BCI y varios voluntarios es los que estos usuarios deben visualizar una serie de imagenes compuestas por direcciones, arriba, abajo, derecha e izquierda.

Este trabajo podría tener aplicaciones importantes en tecnologías de interfaz cerebro-computadora (BCI) aplicadas a por ejemplo, control de sillas de ruedas, control de dispositivos electrónicos, realidad virtual, videojuegos, control de robots, etc...

D.2. Estructura de directorios

La estructura de los directorios del trabajo es la siguiente:

- **/documentacion/**: Esta carpeta contiene toda la documentación relacionada con el TFG.
- **/documentacion/imagenes/**: Archivo de imágenes para todo el proyecto, documentación y código.
- **/documentacion/latex**: Documentación en formato latex y PDF.
- **/codigo**: Carpeta para archivar notebooks y datos para la ejecución del código.
- **/codigo/datos/**: Archivo datosEEGTotal.csv con el origen de datos.
- **/codigo/datos/csv/**: Archivos de datos csv auto generados tras las ejecuciones de algunos notebooks.
- **/codigo/notebooks**: Contiene los notebooks Jupyter y por lo tanto, el código.

D.3. Manual del programador

Este manual sera utilizado principalmente por las personas involucradas en futuros cambios, mejoras o nuevos desarrollos en el código.

Para ello, detallo las funciones de cada uno de los componentes:

- **Notebook Principal (1.Main.ipynb)**:
Crea y ejecuta un entorno virtual. Coordina la ejecución de los notebooks secundarios. Es el punto de entrada para la ejecución del sistema de análisis de datos EEG.
- **Notebooks secundarios**:
 - **2.Bibliotecas.ipynb**:
En este notebooks están todas las instalaciones de bibliotecas necesarias para el sistema.
 - **3.Importaciones.ipynb**:
Importa las bibliotecas y configuraciones necesarias para el análisis futuro de los datos.
 - **SubirCSV.ipynb**:
Permite la subida de un archivo de datos al Notebook principal.
 - **5.CargaDatos.ipynb**:
Realiza un análisis de los datos cargados.
 - **6.Preprocessing.ipynb**:
Realiza la limpieza y preparación inicial de los datos EEG..
 - **7.MachineLearning.ipynb**:
Implementa experimentos utilizando modelos KNN, Árboles de Decisión, Random Forest con validaciones como hold-ouy, k-

fold cross validation ... para evaluar los modelos de aprendizaje automático básico.

- **8.1.DeepLearning-MLP.ipynb**
- **8.2.DeepLearning-SRNN.ipynb**
- **8.3.DeepLearning-LSTM.ipynb:**
Realizan experimentos específicos con modelos de Perceptrón Multicapa, Red Neuronal Recurrente Simple y Memoria a Corto y Largo Plazo, respectivamente.
- **8.4.DeepLearning-SRNN(SlidingWindows).ipynb**
- **8.5.DeepLearning-LSTM(SlidingWindows).ipynb:**
Realizan experimentos específicos con modelos de Perceptrón Multicapa, Red Neuronal Recurrente Simple y Memoria a Corto y Largo Plazo, respectivamente pero añadiendo ventanas deslizantes.
- **9.1.ProcesadoAumentoDatosSmote.ipynb:**
Con smote aumenta y reparte con datos sintéticos la cantidad de datos para el análisis EEG.
- **9.2.DeepLearning-SRNN(AumentoDatos).ipynb**
- **9.3.DeepLearning-LSTM(AumentoDatos).ipynb:**
Implementa experimentos utilizando modelos KNN, Árboles de Decisión, Random Forest con validaciones como hold-out, k-fold cross validation ... pero con el aumento de la cantidad de datos con smote.
- **9.4.DeepLearning-SRNN(AumentoDatosSlidingWindows).ipynb**
- **9.5.DeepLearning-LSTM(AumentoDatosSlidingWindows).ipynb:**
Realizan experimentos específicos con modelos de Perceptrón Multicapa, Red Neuronal Recurrente Simple y Memoria a Corto y Largo Plazo, respectivamente pero añadiendo ventanas deslizantes y aumento de la cantidad de datos con smote.
- **10.Resultadosconjuntodatostest.ipynb:**
Recopila los resultados de los experimentos y los traslada a varias tablas.

Requisitos de Software:

- **Anaconda:** Para el desarrollo del trabajo he utilizado la solución completa de Anaconda así he evitado posibles dependencias a la hora de instalar Python o Jupyter Notebook separados.

Anaconda incluye su propia distribución de Python, por lo que al instalar Anaconda en el sistema, automáticamente se tendrá una instalación de

Python junto con una serie de paquetes adicionales útiles para el análisis de datos y la programación científica.

En la ejecución del proyecto también se ejecuta un entorno virtual, al crear y activar un entorno virtual en el proyecto se obtienen los siguientes beneficios:

1. Aislo dependencias: Puedo instalar bibliotecas y paquetes específicos para el proyecto evitando conflictos entre versiones de paquetes.
2. Reproducibilidad: Se puede reproducir el entorno del proyecto en diferentes ordenadores sin pensar en las dependencias asociadas a los paquetes.
3. Facilidad de gestión: Se puede instalar, actualizar o eliminar paquetes sin afectar a otros proyectos o al sistema de Notebooks de Python.

Para desarrollar este proyecto se ha utilizado la plataforma GitHub. La descarga del repositorio se realizara de la siguiente manera:

1. Acceder al repositorio GitHub [Acceso repositorio Github](#)
2. Pulsar en Code. (Botón verde)
3. Y a continuación pulsar en Download zip.

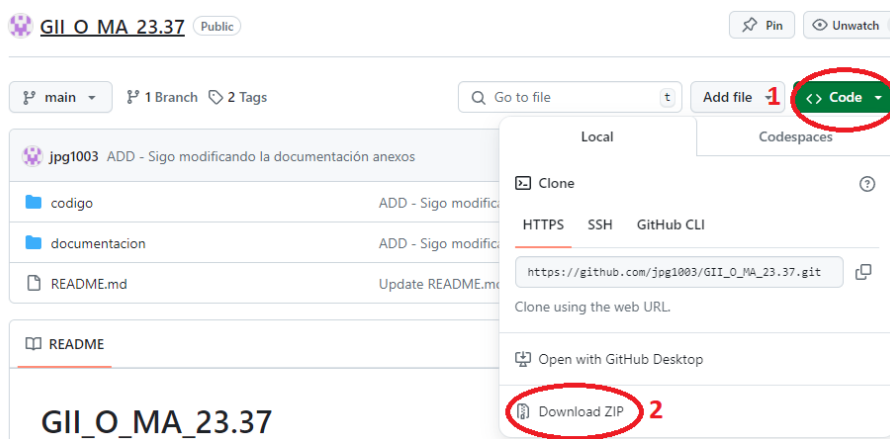


Figura D.1: Descarga repositorio GitHub

4. Llevar el archivo zip descargado a la carpeta que se quiera utilizar para acceder al repositorio.
5. Descomprimir el archivo zip con el nombre de carpeta que se desee.

D.4. Compilación, instalación y ejecución del proyecto

Una vez descargado el repositorio en la carpeta del sistema operativo utilizado se ha de proceder a la instalación del software requerido.

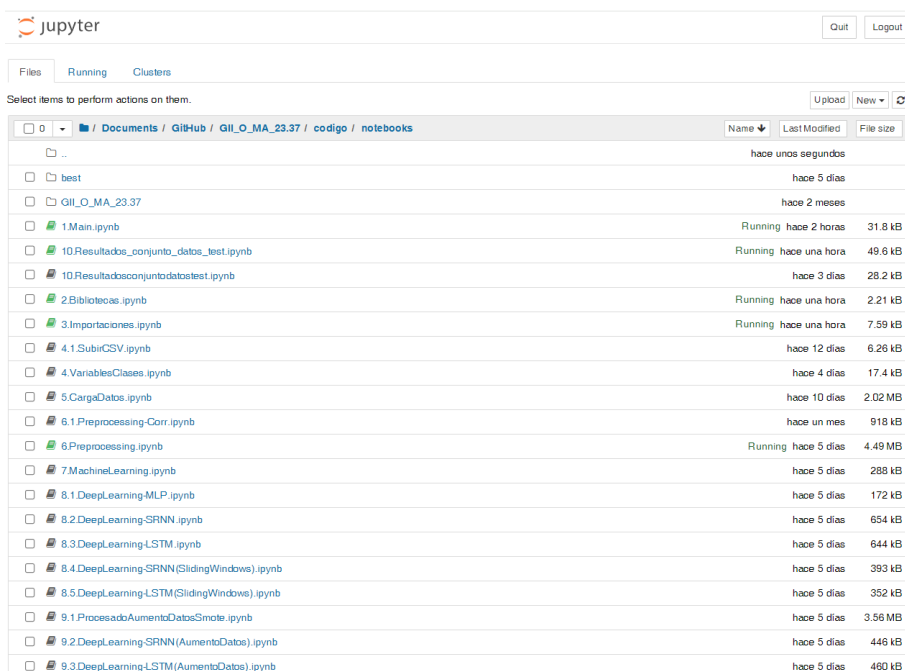
Instalación de software requerido:

■ Anaconda:

- Ir al sitio web oficial de Anaconda en [Enlace url Anaconda](#).
- Descargar la versión más reciente de Python 3.x. para el sistema operativo compatible.
- Instalar el archivo descargado.

Una vez instalado el paquete de aplicaciones Anaconda, en el sistema operativo aparecerá una aplicación llamada **Jupyter Notebook**:

Ejecutar la aplicación **Jupyter Notebook**, una vez iniciada, se abrirá un navegador de Internet (el que este predeterminado en el sistema operativo) y mostrara un explorador de archivos por el cual, tendrá que navegar hasta la carpeta descargada del repositorio Github, se debería ver algo parecido a esto:



| Name | Last Modified | File size |
|---|-----------------------|-----------|
| .. | hace unos segundos | |
| best | hace 5 días | |
| GII_O_MA_23.37 | hace 2 meses | |
| 1.Main.ipynb | Running hace 2 horas | 31.8 kB |
| 10.Resultados_conjunto_datos_test.ipynb | Running hace una hora | 49.6 kB |
| 10.Resultadosconjuntodatos_test.ipynb | hace 3 días | 28.2 kB |
| 2.Bibliotecas.ipynb | Running hace una hora | 2.21 kB |
| 3.Importaciones.ipynb | Running hace una hora | 7.59 kB |
| 4.1.SubirCSV.ipynb | hace 12 días | 6.26 kB |
| 4.VariablesClases.ipynb | hace 4 días | 17.4 kB |
| 5.CargaDatos.ipynb | hace 10 días | 2.02 MB |
| 6.1.Preprocessing-Corr.ipynb | hace un mes | 918 kB |
| 6.Preprocessing.ipynb | Running hace 5 días | 4.49 MB |
| 7.MachineLearning.ipynb | hace 5 días | 288 kB |
| 8.1.DeepLearning-MLP.ipynb | hace 5 días | 172 kB |
| 8.2.DeepLearning-SRNN.ipynb | hace 5 días | 654 kB |
| 8.3.DeepLearning-LSTM.ipynb | hace 5 días | 644 kB |
| 8.4.DeepLearning-SRNN(SlidingWindows).ipynb | hace 5 días | 393 kB |
| 8.5.DeepLearning-LSTM(SlidingWindows).ipynb | hace 5 días | 352 kB |
| 9.1.ProcesadoAumentoDatosSmote.ipynb | hace 5 días | 3.56 MB |
| 9.2.DeepLearning-SRNN(AumentoDatos).ipynb | hace 5 días | 446 kB |
| 9.3.DeepLearning-LSTM(AumentoDatos).ipynb | hace 5 días | 460 kB |

Figura D.2: Listado de Notebooks en Jupyter Notebook

Para ejecutar el proyecto de estudio de datos EEG se puede hacer de dos maneras:

■ Automática:

- Abrir el archivo 1.Main.ipynb a través de Jupyter Notebook y ejecutar una a una las celdas de este notebook. No se deben ejecutar todas a la vez porque se necesita la interacción con el usuario para subir los datos de tipo EEG a analizar.

Manual:

- Abrir y ejecutar cada uno de los notebooks a través de Jupyter Notebook.
- Se ha de seguir la numeración marcada por cada uno de los Notebooks, obviando los notebooks 1.Main.ipynb y los que no tengan numeración.

Con la ejecución Automática los notebooks se ejecutarán en orden y se quedará toda la información en el mismo notebook. Si fuera de manera Manual, se tendría más control de lo que se está ejecutando al estar dividido

el código de cada notebook en varias celdas, pero toda la información estaría dividida en cada uno de los notebooks y no en un solo lugar.

D.5. Pruebas del sistema

No se han realizado test con ninguna herramienta porque habría que realizar transformaciones de los notebooks

Apéndice *E*

Documentación de usuario

E.1. Introducción

Con este manual se pretende ayudar al usuario a poder ejecutar correctamente el proyecto de análisis de datos EEG. A continuación, se describirán los requisitos de usuarios, como instalar el software para su ejecución y un manual de usuario mas detallado.

E.2. Requisitos de usuarios

Los requisitos de los usuarios que podrían aprovechar y utilizar este proyecto serian:

Usuarios potenciales:

1. Doctores, Ingenieros y Estudiantes en Ciencias de Datos.
 - **Descripción:** Individuos avanzados en técnicas de análisis de datos y aprendizaje automático.
 - **Utilización:**
2. Investigadores en Neurociencia.
 - **Descripción:** Profesionales que estudian el cerebro y su actividad.
 - **Utilización:** Análisis de datos EEG para investigar patrones cerebrales relacionados con movimientos específicos y desarrollar aplicaciones BCI.
3. Desarrolladores de Interfaces Cerebro-Computadora (BCI).

- **Descripción:** Ingenieros y desarrolladores que crean sistemas que permiten la comunicación directa entre el cerebro y dispositivos externos.
 - **Utilización:** Análisis de datos EEG para mejorar la precisión y eficiencia de las interfaces cerebro-computadora enfocadas en el control direccional.
4. Desarrolladores de Videojuegos.
- **Descripción:** Profesionales que desarrollan videojuegos controlados por la mente.
 - **Utilización:** Implementación de sistemas de control mediante EEG para crear experiencias de juego inmersivas y accesibles para personas con discapacidad.
5. Diseñadores de Dispositivos de Asistencia
- **Descripción:** Ingenieros que diseñan sillas de ruedas y otros dispositivos controlados por EEG.
 - **Utilización:** Creación de sistemas de control direccional basados en EEG para mejorar la movilidad e independencia de personas con discapacidades motoras.

Requisitos Educativos:

1. Conocimientos Básicos.
- **Programación:** Conocimiento de Python y experiencia con Jupyter Notebooks.
 - **Matemáticas y Estadística:** Conceptos básicos de álgebra lineal, cálculo, y estadística.
 - **Neurociencia Básica:** Conocimientos generales sobre la actividad cerebral y los principios de la electroencefalografía (EEG).
2. Conocimientos Avanzados.
- **Ciencia de Datos:** Experiencia en manejo y análisis de conjunto de datos.
 - **Aprendizaje Automático:** Conocimiento práctico de técnicas de machine learning y su implementación.

Con estos conocimientos y habilidades, los usuarios podrán utilizar efectivamente el entorno desarrollado para el análisis y clasificación de datos EEG.

E.3. Instalación

Instalación de software requerido:

1. Acceder al repositorio GitHub [Acceso repositorio Github](#)
2. Pulsar en Code. (Botón verde)
3. Y a continuación pulsar en Download zip.

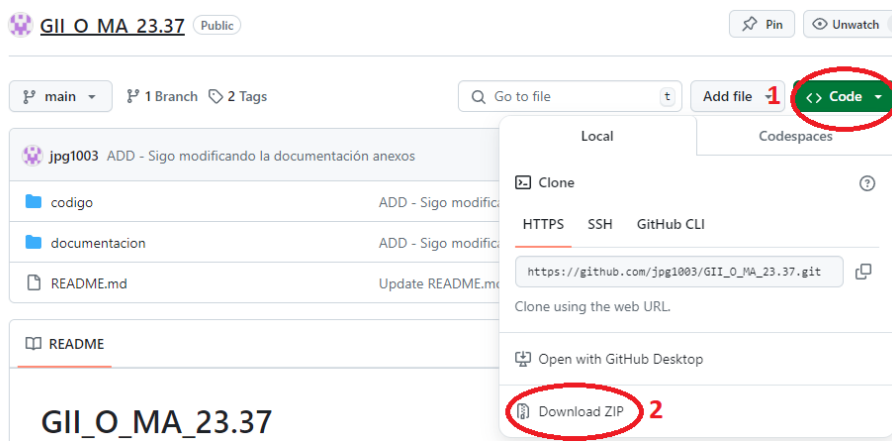


Figura E.1: Descarga repositorio GitHub

4. Llevar el archivo zip descargado a la carpeta que se quiera utilizar para acceder al repositorio.
5. Descomprimir el archivo zip con el nombre de carpeta que se desee.

Una vez descargado el repositorio en la carpeta del sistema operativo utilizado se ha de proceder a la instalación del software requerido.

Instalación de software requerido:

■ Anaconda:

- Ir al sitio web oficial de Anaconda en [Enlace url Anaconda](#).
- Descargar la versión más reciente de Python 3.x. para el sistema operativo compatible.
- Instalar el archivo descargado.

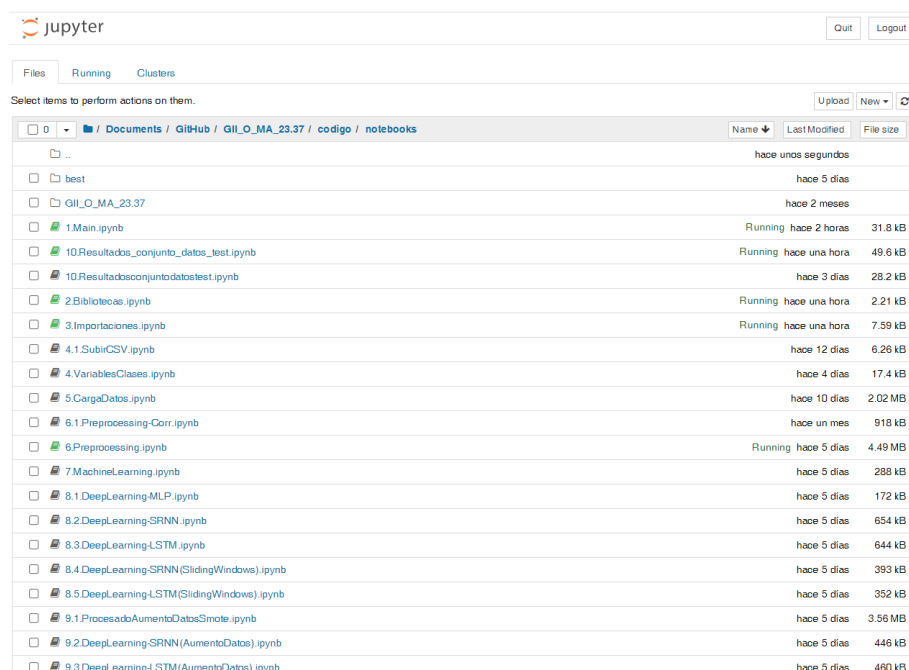
Una vez instalado el paquete de aplicaciones Anaconda, en el sistema operativo aparecerá una aplicación llamada **Jupyter Notebook**:

E.4. Manual del usuario

Después de la instalación del software necesario para la ejecución del proyecto se detallarán las diferentes partes del código y como llegar a ellas:

Iniciar proyecto análisis de datos EEG:

Ejecutar la aplicación **Jupyter Notebook**, una vez iniciada, se abrirá un navegador de Internet (el que este predeterminado en el sistema operativo) y mostrara un explorador de archivos por el cual, tendrá que navegar hasta la carpeta descargada del repositorio Github, se debería ver algo parecido a esto:



The screenshot shows the Jupyter Notebook interface with the 'Files' tab selected. The breadcrumb path is 'Documents / GitHub / GII_O_MA_23.37 / codigo / notebooks'. A table lists the following items:

| Name | Last Modified | File size |
|---|-----------------------|-----------|
| .. | hace unos segundos | |
| best | hace 5 días | |
| GIIO_MA_23.37 | hace 2 meses | |
| 1.Main.ipynb | Running hace 2 horas | 31.8 kB |
| 10.Resultados_conjunto_datos_test.ipynb | Running hace una hora | 49.6 kB |
| 10.Resultadosconjuntodatos_test.ipynb | hace 3 días | 28.2 kB |
| 2.Bibliotecas.ipynb | Running hace una hora | 2.21 kB |
| 3.Importaciones.ipynb | Running hace una hora | 7.59 kB |
| 4.1.SubirCSV.ipynb | hace 12 días | 6.26 kB |
| 4.VariablesClases.ipynb | hace 4 días | 17.4 kB |
| 5.CargaDatos.ipynb | hace 10 días | 2.02 MB |
| 6.1.Preprocessing-Corr.ipynb | hace un mes | 918 kB |
| 6.Preprocessing.ipynb | Running hace 5 días | 4.49 MB |
| 7.MachineLearning.ipynb | hace 5 días | 288 kB |
| 8.1.DeepLearning-MLP.ipynb | hace 5 días | 172 kB |
| 8.2.DeepLearning-SRNN.ipynb | hace 5 días | 654 kB |
| 8.3.DeepLearning-LSTM.ipynb | hace 5 días | 644 kB |
| 8.4.DeepLearning-SRNN(SlidingWindows).ipynb | hace 5 días | 393 kB |
| 8.5.DeepLearning-LSTM(SlidingWindows).ipynb | hace 5 días | 352 kB |
| 9.1.ProcesadoAumentoDatosSmote.ipynb | hace 5 días | 3.56 MB |
| 9.2.DeepLearning-SRNN(AumentoDatos).ipynb | hace 5 días | 446 kB |
| 9.3.DeepLearning-LSTM(AumentoDatos).ipynb | hace 5 días | 460 kB |

Figura E.2: Listado de Notebooks en Jupyter Notebook

Para ejecutar el proyecto de estudio de datos EEG se puede hacer de dos maneras:

■ Automática:

- Abrir el archivo 1.Main.ipynb a través de Jupyter Notebook y ejecutar una a una las celdas de este Notebook. No se deben ejecutar todas a la vez porque se necesita la interacción con el usuario para subir los datos de tipo EEG a analizar.

Manual:

- Abrir y ejecutar cada uno de los notebooks a través de Jupyter Notebook.
- Se ha de seguir la numeración marcada por cada uno de los Notebooks, obviando los notebooks 1.Main.ipynb y los que no tengan numeración.

Con la ejecución semiautomática los notebooks se ejecutaran en orden y se quedara toda la información en el mismo notebook. Si fuera la ejecución fuera de manera Manual, se tendría más control de lo que se esta ejecutando al estar dividido el código de cada notebook en varias celdas, pero toda la información estaría dividida en cada uno de los notebooks y no como en la ejecución automática en un solo.

1. Ejecución automática:

- **Abrir Notebook principal:** Desde Jupyter Notebook, seleccionar el archivo 1.Main.ipynb y hacer doble click sobre el archivo. Este seria el notebook ya abierto:

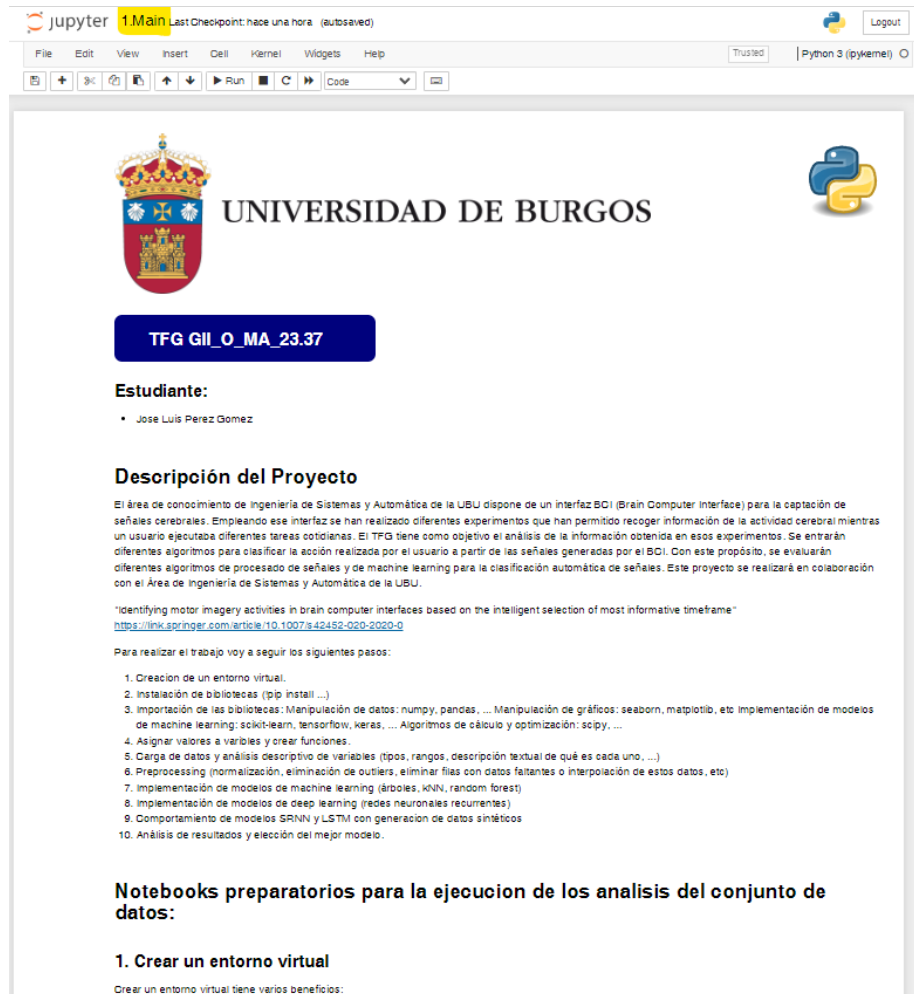


Figura E.3: Ejemplo Notebook Main

■ Ejecución celdas preparatorias del entorno:

a) Celda ejecutable 1. Crear un entorno virtual:

En esta celda se crea un entorno virtual para las ejecuciones posteriores.

b) Celda ejecutable 2. Instalación de bibliotecas:

Desde esta celda la llamada al Notebook secundario 2.Bibliotecas.ipynb instalaría las bibliotecas necesarias para el proyecto.

c) Celda ejecutable 3.Importación de las bibliotecas instaladas:

Esta celda se ejecuta llamando al Notebook secundario 3.Importaciones.ipynb que se ocupa de la declaracion de las importaciones de las bibliotecas instaladas en el paso anterior.

d) Celda ejecutable 4. Asignación de valores a variables y creación de funciones:

Celda con llamada al Notebook secundario 4.VariablesClases.ipynb. Este Notebook asigna valores a variables que se utilizaran en cualquier momento durante el analisis de datos EEG.

e) Celda ejecutable para la carga de archivo CSV con el conjunto de datos a analizar:

Celda que necesita la interacción con el usuario. Solo podra subirse un archivo en formato csv. Cuando se ejecuta la celda aparecerán dos botones, Upload(0) (activo) y Procesar datos (sin activar):

Carga de archivo CSV con el conjunto de datos a analizar:

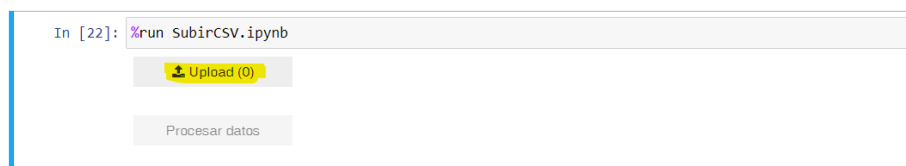


Figura E.4: Botón Upload

Se selecciona el archivo a subir con un explorador de archivos y el botón Upload reflejaría entre paréntesis que tiene un archivo subido (1) y si el archivo ha sido subido con éxito mediante texto. El botón Procesar Datos ahora aparecerá activo. Si el archivo no se ha podido subir aparecerá un error en la subida del archivo.

Carga de archivo CSV con el conjunto de datos a analizar:

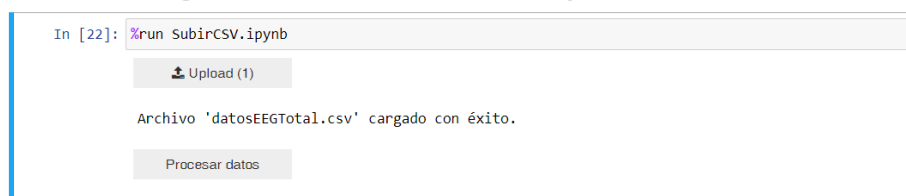


Figura E.5: Subida correcta de archivo csv.

A continuación se pulsa sobre el botón Procesar Datos y se imprimirá por pantalla el archivo que se ha subido al entorno

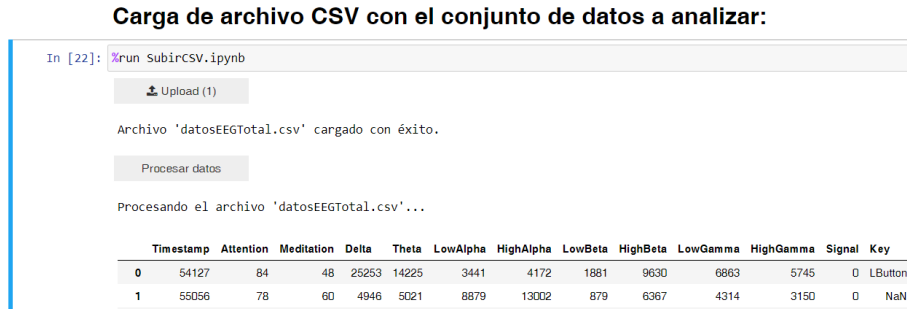


Figura E.6: Datos después de subirlos al entorno correctamente.

- **Ejecución celdas para análisis y preprocesado del conjunto de datos:**

- a) **Celda ejecutable 5. Análisis de datos inicial:**

Celda con llamada al Notebook secundario 5.CargaDatos.ipynb para realizar un análisis del tipo y forma del conjunto de datos.

- b) **Celda ejecutable 6. Preprocesado del conjunto de datos:**

Celda con llamada al Notebook secundario 6.Preprocessing.ipynb para realizar la estandarización, eliminación de outliers, rellenar datos perdidos en filas del conjunto de datos, ...

- **Ejecución celdas para experimentos de varios modelos de aprendizaje automático:**

- a) **Celda ejecutable 7. Implementación modelos Machine Learning:**

La llamada al Notebook secundario 7.MachineLearning.ipynb activa el primer experimento para realizar un análisis de los datos con modelos KNN(K-Nearest Neighbors), Árboles de Decisión, Random Forest con validaciones como holdout, k-folds cross validation y leave-one-out.

- b) **8.1 MLP:**

Llamada a Notebook secundario 8.1.DeepLearning-MLP.ipynb que inicia un experimento para realizar un análisis de los datos con modelo MLP (Multi-Layer Perceptron).

- c) **8.2 SRNN:**

Llamada a Notebook secundario 8.2.DeepLearning-SRNN.ipynb que inicia un experimento para realizar un análisis de los datos con modelo SRNN (Simple Recurrent Neural Network).

d) **8.3 LSTM:**

Llamada a Notebook secundario 8.3.DeepLearning-LSTM.ipynb que inicia un experimento para realizar un análisis de los datos con modelo LSTM (Long Short-Term Memory).

e) **8.4 SRNN Sliding Windows:**

Llamada a Notebook secundario 8.4.DeepLearning-SRNN(SlidingWindows).ipynb que inicia un experimento para realizar un análisis de los datos con modelo SRNN y implementación de ventanas deslizantes.

f) **8.5 LSTM Sliding Windows:**

Llamada a Notebook secundario 8.5.DeepLearning-LSTM(SlidingWindows).ipynb que inicia un experimento para realizar un análisis de los datos con modelo LSTM e implementación de ventanas deslizantes.

g) **9.1 Aumento de datos con SMOTE:**

Llamada a Notebook secundario 9.1.ProcesadoAumentoDatosSmote.ipynb que utiliza smote para poder crear datos sintéticos y distribuir los datos equitativamente según los targets que se le indiquen.

h) **9.2 SRNN Aumento de datos:**

Llamada a Notebook secundario 9.2.DeepLearning-SRNN(AumentoDatos).ipynb que inicia un experimento para realizar un análisis de los datos con modelo SRNN y con datos sintéticos.

i) **9.3 LSTM Aumento de datos:**

Llamada a Notebook secundario 9.3.DeepLearning-LSTM(AumentoDatos).ipynb que inicia un experimento para realizar un análisis de los datos con modelo LSTM y con datos sintéticos.

j) **9.4 SRNN Aumento de datos Sliding Windows:**

Llamada a Notebook secundario 9.4.DeepLearning-SRNN(AumentoDatosSlidingWindows).ipynb que inicia un experimento para realizar un análisis de los datos con modelo SRNN ventanas deslizantes y con datos sintéticos.

k) **9.5 LSTM Aumento de datos Sliding Windows:**

Llamada a Notebook secundario 9.5.DeepLearning-LSTM(AumentoDatosSlidingWindows).ipynb que inicia un experimento para realizar un análisis de los datos con modelo LSTM ventanas deslizantes y con datos sintéticos.

- **Ejecución celda 10. Recopilado de resultados:** Realiza la ultima llamada a Notebooks secundarios. El Notebook 10.Resultadosconjuntodatosetest.ipynb recopila e imprime por pantalla todos los resultados de los datos de validación y test.

2. Ejecución manual:

Para la ejecución manual se ha de seguir el orden preestablecido con la numeración de los archivos ipynb, empezando por el numero 2 y terminando por el numero 10 de uno en uno.

Algo en común con todos los Notebooks que no sean el principal (1.Main) a partir del numero 4, es que se ha de modificar su ejecución de AUTOMATICA A MANUAL, cada vez que se abra un Notebook secundario se tendrá que realizar esta acción antes de su ejecución:

```
# escribir solo entre las comillas AUTOMATICA o MANUAL  
EJECUCION = 'AUTOMATICA'
```

```
# en esta celda no ha de ser modificada  
if EJECUCION == 'MANUAL':  
  
    %run 3.Importaciones.ipynb
```

Figura E.7: Ejemplo instalación automática

Y se debe cambiar a MANUAL:

```
# escribir solo entre las comillas AUTOMATICA o MANUAL  
EJECUCION = 'MANUAL'
```

```
# en esta celda no ha de ser modificada  
if EJECUCION == 'MANUAL':  
  
    %run 3.Importaciones.ipynb
```

Figura E.8: Ejemplo instalación manual

IMPORTANTE: Si se quisiera ejecutar el proyecto en manera AUTOMÁTICA habría que cambiar la ejecución de nuevo a MANUAL en los Notebooks anteriormente cambiados.

a) **Abrir y ejecutar Notebook 2.Bibliotecas.ipynb:**

Desde Jupyter Notebook, seleccionar el archivo 2.Bibliotecas.ipynb y hacer doble clic sobre el archivo y después ejecutarlo.

En este Notebook se pueden añadir mas bibliotecas a instalar si fuera necesario su uso por futuras integraciones. Este seria un ejemplo de las bibliotecas que se instalan:

```
#Cálculo numérico
!pip install numpy
!pip install scipy

#Manipulación y análisis de datos:
!pip install pandas

#Visualización de datos:
!pip install matplotlib
!pip install seaborn
```

Figura E.9: Ejemplo instalación bibliotecas

b) **Abrir y ejecutar Notebook 3.Importaciones.ipynb:**

Desde Jupyter Notebook, seleccionar el archivo 3.Importaciones.ipynb y hacer doble clic sobre el archivo y después ejecutarlo. En este Notebook se pueden añadir mas declaraciones a importar de las bibliotecas instaladas en el Notebook anterior, se deberían definir las declaraciones necesarias si hubiera alguna futura integración. Este seria un ejemplo de las declaraciones que se declaran:

```
# Manipulación y análisis de datos
import pandas as pd

# Visualización de datos
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import ConfusionMatrixDisplay
```

Figura E.10: Ejemplo declaraciones a importar bibliotecas

- c) **Abrir y ejecutar Notebook 4.VariablesClases.ipynb:**
Desde Jupyter Notebook, abrir 4.VariablesClases.ipynb y ejecutar. En este Notebook se pueden añadir mas variables o funciones si hubiera alguna futura integración. Si el usuario dispone de conocimiento para modificar el valor de las variables definidas, cambiaría el alcance de los experimentos ejecutados puesto que estas variables se usan en cualquier ejecución posterior en los experimentos:

```
#Variables que se podrian modificar

SEED = 42
TRAIN = 0.9
TRAIN80 = 0.8
CV = 10
MAX_ITERACIONES = 10
ESTIMATOR = 10
MAX_DEPTH = 10
UNITS = 30
DROPOUT = 0.3
TIMESTEPS = 1
FEATURES = 10
NUM_CLASES = 5
LEARNING_RATE = 0.001
OPTIMIZER = Adam(learning_rate=LEARNING_RATE)
ACTIVATION = 'softmax'
METRICS = 'accuracy'
LOSS = 'sparse_categorical_crossentropy'
EPOCHS = 50
BACH_SIZE = 30
```

Figura E.11: Ejemplo variables declaradas

d) **Abrir y ejecutar Notebook 5.CargaDatos.ipynb:**

Desde Jupyter Notebook, abrir 5.CargaDatos.ipynb y ejecutar. Este Notebook realiza un análisis mediante gráficas o impresiones por texto del conjunto de datos aportado:

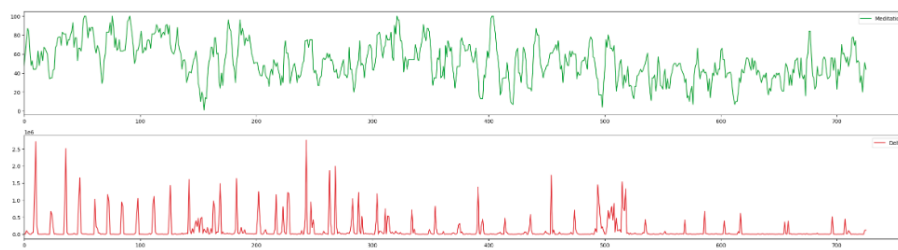


Figura E.12: Ejemplo gráfica en análisis de datos

e) **Abrir y ejecutar Notebook 6.Preprocessing.ipynb:**

Desde Jupyter Notebook, abrir 6.Preprocessing.ipynb y ejecutar. En este Notebook se realiza el preprocesado del conjunto de datos, para ello realiza varias acciones como rellenado de datos missing, eliminacion de outliers, estandarizacion de datos. Si el usuario dispone de conocimiento para modificar este codigo pero afectaría al resto de ejecuciones posteriores, concretamente los experimentos

f) **Abrir y ejecutar Notebook del 7 al 8.5 :**

Desde Jupyter Notebook, abrir y ejecutar los notebooks indicados. Estos notebooks contienen los experimentos para Machine y Deep Learning, no deberían ser modificados si no se tiene suficiente conocimiento de estos modelos. Lo ideal seria consultar con el desarrollador.

Los resultados que se imprimirán son resultados de tipo tasa de acierto, comparandola con diferentes ejecuciones, y matrices de confusión.

Ejemplo de Matrices de confusión y de comparativa en tasa de acierto por modelo:

| | HoldOut | | |
|------------|----------|----------|----------|
| | KNN | TREE | RANDOM |
| Segmento 1 | 0.416667 | 0.416667 | 0.416667 |
| Segmento 2 | 0.400000 | 0.333333 | 0.200000 |
| Segmento 3 | 0.277778 | 0.277778 | 0.500000 |
| Segmento 4 | 0.222222 | 0.444444 | 0.166667 |

Figura E.13: Ejemplo tabla comparativa de tasa de acierto entre modelos

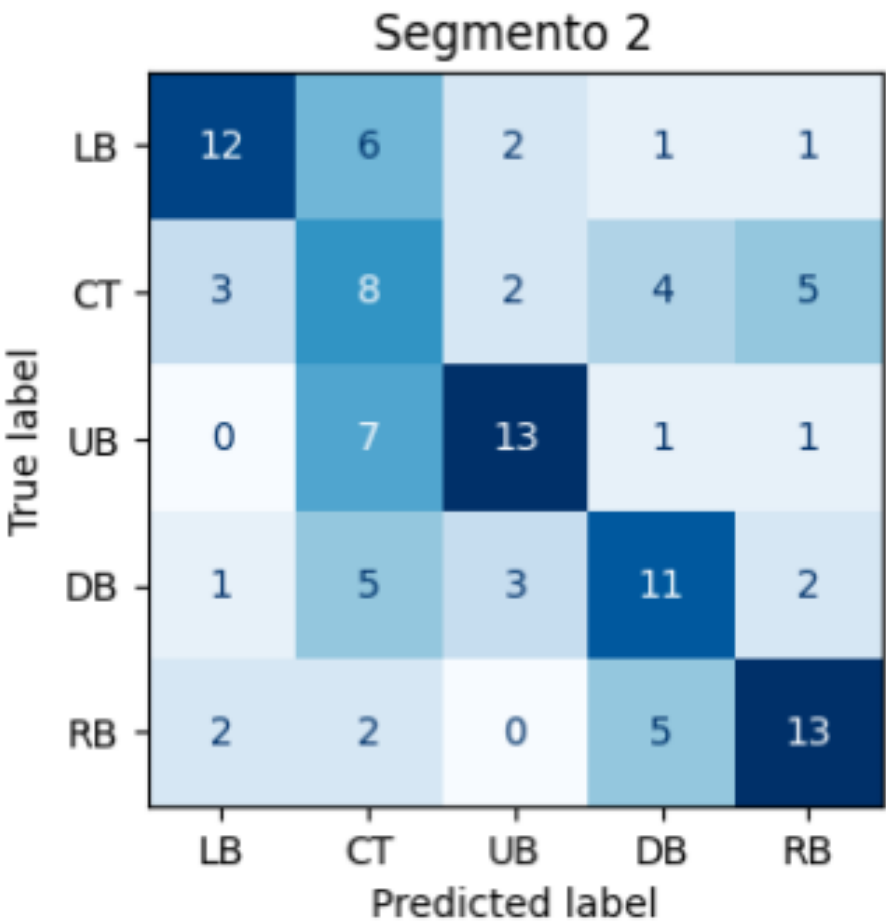


Figura E.14: Ejemplo matriz de confusión para un modelo

g) **Abrir y ejecutar Notebook 10.Resultadosconjuntodatos-test.ipynb:**

Desde Jupyter Notebook, abrir y ejecutar el Notebook 10.Resultadosconjuntodatos-test.ipynb. Este notebook recopila todos los datos referentes a tasa de acierto en los modelos ejecutados tanto para los datos de validación y test.

Ejemplo de tabla comparativa en tasa de acierto en este notebook:

Resultados de los siguientes experimento para particion de datos VALIDACION

| | KNN_VAL | TREE_VAL | RANDOM_VAL | MLP_VAL | SRNN_VAL | LSTM_VAL |
|----------------------|----------|----------|------------|----------|----------|----------|
| Segmento 1 | 0.416667 | 0.416667 | 0.500000 | 0.416667 | 0.333333 | 0.333333 |
| Segmento 2 | 0.400000 | 0.333333 | 0.333333 | 0.200000 | 0.400000 | 0.266667 |
| Segmento 3 | 0.277778 | 0.222222 | 0.333333 | 0.388889 | 0.333333 | 0.333333 |
| Segmento 4 | 0.222222 | 0.277778 | 0.222222 | 0.111111 | 0.111111 | 0.111111 |
| All Segmentos after | 0.344262 | 0.213115 | 0.311475 | 0.278689 | 0.360656 | 0.360656 |
| All Segmentos before | 0.393443 | 0.295082 | 0.262295 | 0.377049 | 0.311475 | 0.344262 |

Figura E.15: Ejemplo tabla comparativa de resultados

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

Este anexo incluirá una reflexión personal del alumnado sobre los aspectos de la sostenibilidad que se abordan en el trabajo. Se pueden incluir tantas subsecciones como sean necesarias con la intención de explicar las competencias de sostenibilidad adquiridas durante el alumnado y aplicadas al Trabajo de Fin de Grado.

Más información en el documento de la CRUE https://www.crue.org/wp-content/uploads/2020/02/Directrices_Sostenibilidad_Crue2012.pdf.

Este anexo tendrá una extensión comprendida entre 600 y 800 palabras.