



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería
Informática

Análisis de señales EEG
Documentación Técnica



Presentado por José Luis Pérez Gómez
en Universidad de Burgos — 9 de julio de 2024
Tutor: Bruno Baruque Zanón - Jesús Enrique
Sierra García

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	10
Apéndice B Especificación de Requisitos	15
B.1. Introducción	15
B.2. Objetivos generales	15
B.3. Catálogo de requisitos	16
B.4. Especificación de requisitos	18
Apéndice C Especificación de diseño	27
C.1. Introducción	27
C.2. Diseño de datos	27
C.3. Diseño procedimental	30
C.4. Diseño arquitectónico	33
Apéndice D Documentación técnica de programación	35
D.1. Introducción	35
D.2. Estructura de directorios	35
D.3. Manual del programador	36

D.4. Compilación, instalación y ejecución del proyecto	39
D.5. Pruebas del sistema	41
Apéndice E Documentación de usuario	45
E.1. Introducción	45
E.2. Requisitos de usuarios	45
E.3. Instalación	47
E.4. Manual del usuario	50
Apéndice F Anexo de sostenibilización curricular	63
F.1. Introducción	63
F.2. Aplicación práctica	63
F.3. Pensamiento crítico	64
F.4. Impacto Personal	64
F.5. Conclusión	65
Bibliografía	67

Índice de figuras

A.1. Commits realizados durante la realización del TFG	2
A.2. Additions realizados durante la realización del TFG	2
A.3. Deletions realizados durante la realización del TFG	3
A.4. Commits-Etapa 1	4
A.5. Commits-Etapa 2	6
A.6. Commits-Etapa 3	7
C.1. Cantidad de elementos para la característica Key	28
C.2. Análisis característica Timestamp del conjunto de datos	29
C.3. Diagrama Entidad Relación	30
C.4. Diagrama de interacción entre módulos	31
C.5. Organización del sistema	33
D.1. Descarga repositorio GitHub	38
D.2. Listado de Notebooks en Jupyter Notebook	40
D.3. Ejemplo resultados Tasa de acierto primer experimento compa- rativo para subconjunto Test	41
D.4. Ejemplo resultados Tasa de acierto experimento con aumento de datos a Test	42
D.5. Ejemplo resultados Tasa de acierto experimento ventanas tempo- rales a Test	43
D.6. Ejemplo resultados Tasa de acierto experimento ventanas tempo- rales con aumento de datos a Test	43
D.7. Ejemplo resultados Tasa de acierto experimento aumento de datos a datos reales Test	44
E.1. Inicio instalación máquina virtual	48
E.2. Importación máquina virtual	48

E.3. Arrancar máquina virtual	49
E.4. Descarga repositorio GitHub	49
E.5. Listado de Notebooks en Jupyter Notebook	51
E.6. Ejemplo Notebook Main	52
E.7. Botón Upload	53
E.8. Subida correcta de archivo csv.	54
E.9. Datos después de subirlos al entorno correctamente.	54
E.10. Ejemplo instalación automática	56
E.11. Ejemplo instalación manual	57
E.12. Ejemplo instalación bibliotecas	58
E.13. Ejemplo declaraciones a importar bibliotecas	58
E.14. Ejemplo variables declaradas	59
E.15. Ejemplo gráfica en análisis de datos	60
E.16. Ejemplo tabla comparativa de tasa de acierto entre modelos	61
E.17. Ejemplo matriz de confusión para un modelo	61
E.18. Ejemplo tabla comparativa de resultados	62

Índice de tablas

A.1. Tabla de tiempos para las primeras fases	5
A.2. Tabla de tiempos para las segundas fases	7
A.3. Tabla de tiempos para las ultimas fases	8
A.4. Costes del desarrollo	11
A.5. Licencias Software	12
B.1. CU-001 Carga archivo con datos EEG.	19
B.2. CU-002 Preprocesamiento de Datos EEG.	19
B.3. CU-003 Visualización de datos.	20
B.4. CU-004 Entrenamiento y validación de modelos de aprendizaje automático básico.	21
B.5. CU-005 Entrenamiento y validación de modelos de aprendizaje automático neuronal	22
B.6. CU-006 Control de BCI	23
B.7. CU-007 Almacenamiento y Recuperación de Resultados de Ex- perimentos	24
B.8. CU-008 Integración y Extensibilidad del sistema	25

Apéndice A

Plan de Proyecto Software

A.1. Introducción

La ejecución de este proyecto se llevó a cabo siguiendo el método CRISP-DM (Cross-Industry Standard Process for Data Mining), un marco estructurado que guió todas las etapas del proceso.

Desde la comprensión inicial del problema hasta la implementación final de la solución, cada fase fue cuidadosamente planificada y ejecutada.

Para la realización del proyecto 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 como lenguaje de programación para escribir el código y una serie de cursos de aprendizaje básicos para poder acometer el TFG sin problemas.

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

Desde el principio del proyecto debido a circunstancias personales no se ha podido realizar metodología Scrum, pero siguiendo el método CRISP-DM (Cross-Industry Standard Process for Data Mining) se produjeron estas líneas temporales:

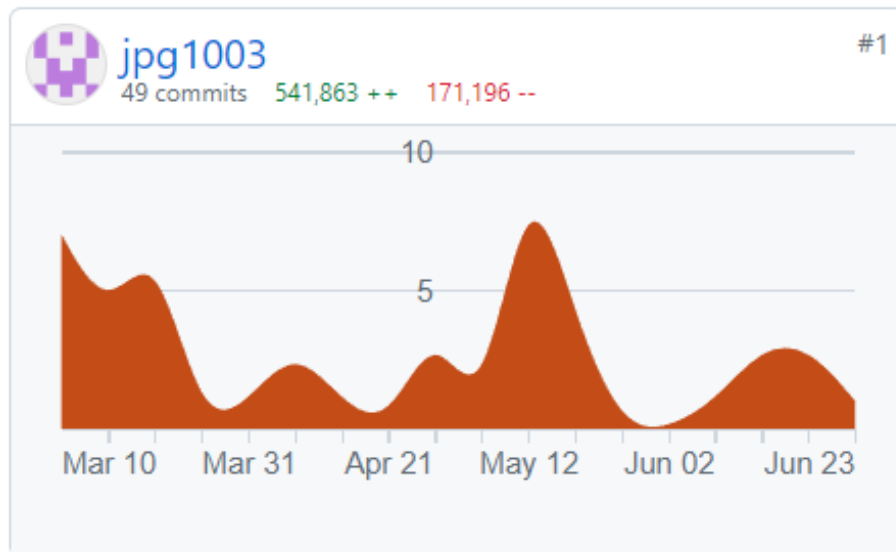


Figura A.1: Commits realizados durante la realización del TFG



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

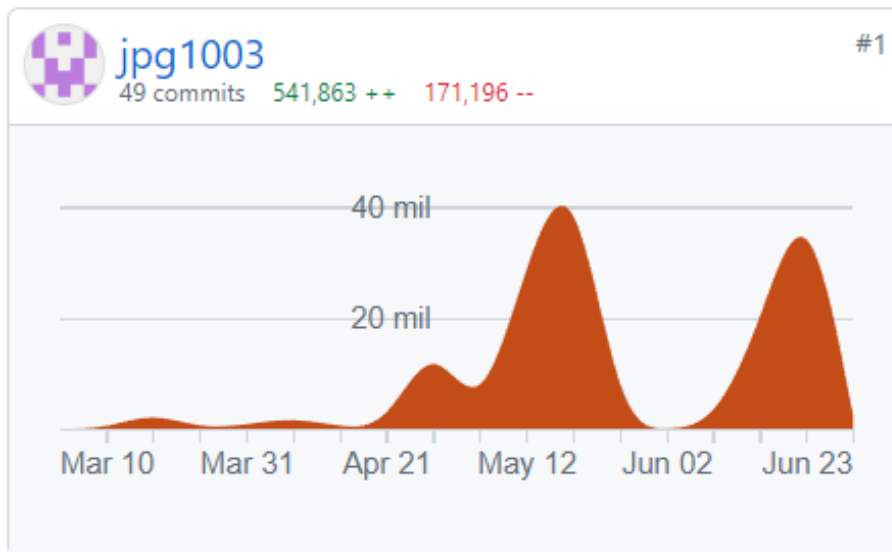


Figura A.3: Deletions realizados durante la realización del TFG

Se pueden dividir en 3 grandes etapas:

- Fase de comprensión del Negocio y compresión de los datos: (Febrero a Marzo)
- Fase de modelado y Fase de evaluación: (Abril a Mayo)
- Fase de Evaluación y despliegue: (Junio a Julio)

Fase de comprensión del Negocio y compresión de los datos: (Febrero a Marzo)

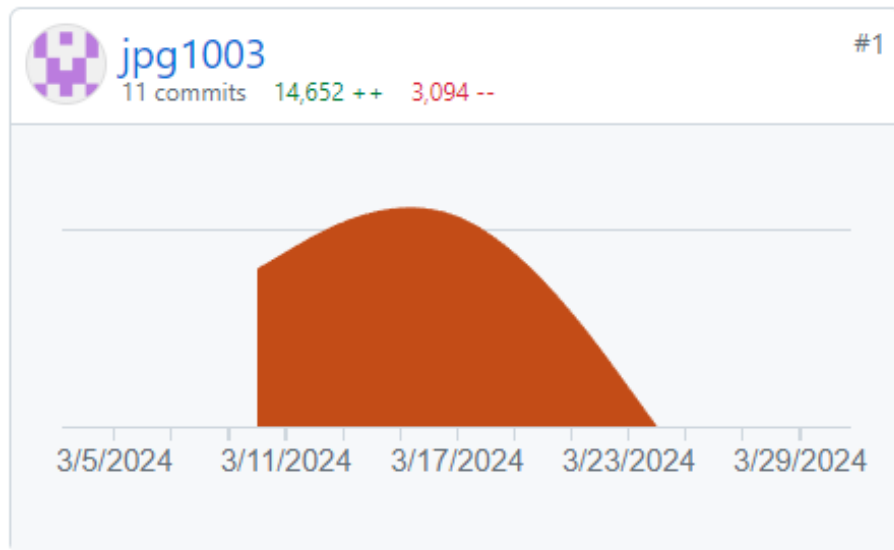


Figura A.4: Commits-Etapa 1

- Realización de cursos online:

Se han realizado cursos online básicos propuestos en las reuniones de seguimiento: Tutorial de Inicio de Pandas [4], Visualización de Datos [5], Trabajo con series temporales. [3]

- 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 modelo o experimento a realizar.

- Impresiones de gráficas y definiciones básicas como normalizar o escalar el conjunto de datos.

Tabla de tiempos

Actividad	Estimación temporal
Curso Tutorial de Inicio de Pandas	6 horas
Curso Visualización de Datos	5 horas
Curso Trabajo con series temporales	5 horas
Análisis datos	25 horas
Definir estructura notebooks	5 horas
Pruebas de impresión	7 horas
Inicio Preprocesado	30 horas
Inicio Machine Learning	20 horas
Total	103 horas

Tabla A.1: Tabla de tiempos para las primeras fases

Fase de modelado y Fase de evaluación: (Abril a Mayo)

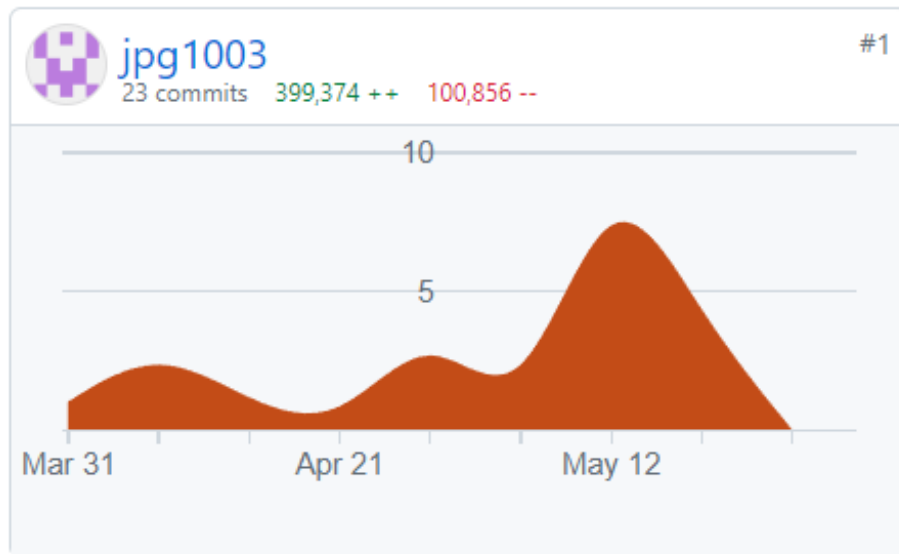


Figura A.5: Commits-Etapa 2

- 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.

Actividad	Estimación temporal
Cambio en el preprocesado	20 horas
Modelos Machine Learning	25 horas
Modelos Deep Learning	52 horas
Ventanas Temporales	12 horas
Callbacks	6 horas
Pruebas de impresión	8 horas
Total	123 horas

Tabla A.2: Tabla de tiempos para las segundas fases

Fase de Evaluación y despliegue: (Junio a Julio)

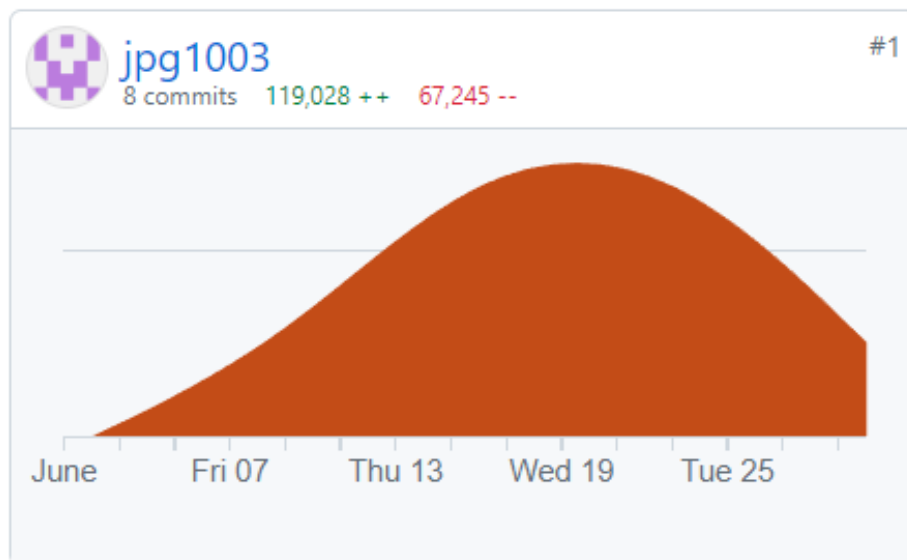


Figura A.6: Commits-Etapa 3

- **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 llegó a la definición correcta para las ventanas temporal en el conjunto de datos.

- **Utilización de modelos deep learning y callbacks.**

- **Utilización datos sintéticos.** Definir correctamente esta generación de datos y poder utilizarlos en los modelos deep learning correctamente.

Actividad	Estimación temporal
Modelos Deep Learning	40 horas
Ventanas Temporales	8 horas
Datos sintéticos	12 horas
Pruebas de impresión	3 horas
Documentación	72 horas
Total	135 horas

Tabla A.3: Tabla de tiempos para las ultimas fases

En este proyecto el tiempo dedicado ha sido de 361 horas.

Releases

Release 1.0

En esta primera release el código era funcional pero no se llegó a un acuerdo para la utilización de ventanas temporales.

El código consta de varias fases: Análisis de datos, Preprocessing de datos, Machine Learning y Deep Learning. En Deep Learning es donde no acabamos de aprobar un código para terminar de cerrarlo y poder entregarlo.

Hasta ahora los datos aportados en Machine Learning son muy positivos dando tasas de acierto mayores de 90 por ciento (Esto luego se demostró que no era real), pero en cambio para Deep Learning lo máximo que aportan los modelos es de una tasa de acierto del 50 por ciento.

Estamos conversando en como aplicar ventanas deslizantes en el análisis de los datos, pero puede que estos datos sean de una extensión muy pequeña para que los modelos LSTM o RNN puedan procesarlos correctamente y llegar a aprender y no memorizar datos, hasta ahora en los entrenamientos el modelo empieza a realizar sobreajuste a los pocos epochs de empezar el entrenamiento.

Los siguientes pasos serian: poder cerrar el código, revisión y limpieza de código, conclusiones, crear la memoria con todos los cambios.

Release 2.0

En esta release se llegó a un acuerdo para el código entre los tutores y yo.

El código consta de varias fases: Análisis de datos, Preprocessing de datos, Machine Learning y Deep Learning. En Deep Learning ya se llego al acuerdo de implementar Ventanas Temporal asociadas a solo un target por ventana y utilizando stratify.

En la reunión también salió un nuevo experimento con datos sintéticos que ya están implementados en esta release. Los porcentajes aportados para tasa de acierto en Machine Learning y Deep Learning son inferiores al 50 por ciento pero con aumento de datos con smote podemos llegar a tasas de acierto mas esperanzadoras.

Los siguientes pasos serian: revisar y seguir comentando lo que quede del nuevo código, limpieza de código, crear memoria y anexo con todos los cambios.

Release 3.0

En esta release se modifica el modelo MLP y se empieza a documentar la memoria y los anexos.

Esta release contiene todos los apartados comentados en la releases anteriores.

Esta igual que las otras releases es una version totalmente funcional.

Release 4.0

En esta ultima release se limpian los comentarios en los Notebooks y se completa la documentación en latex de los archivos memoria y anexos. También se realiza limpieza de código.

Esta release contiene todos los apartados comentados en la releases anteriores.

Esta release es la version definitiva.

A.3. Estudio de viabilidad

Viabilidad económica

La viabilidad económica evalúa la posibilidad de generar ingresos y recuperar la inversión realizada a través del proyecto desarrollado. Se han de considerar las siguientes áreas:

- **Potencial del proyecto para Otros BCIs:**

El código desarrollado para el análisis de señales EEG podría adaptarse y aplicarse a una variedad de sistemas BCI disponibles en el mercado. La flexibilidad del lenguaje de programación Python permite su integración con sistemas existentes, ampliando su potencial de mercado.

- **Mercado Objetivo:**

Los fabricantes de sistemas BCI constituyen el mercado principal para la comercialización del software desarrollado. Ejemplos de fabricantes incluyen NeuroSky, Emotiv, Brain Products y otros proveedores de dispositivos de interfaz cerebro-ordenador utilizados en investigación, medicina y aplicaciones comerciales. En España destacan Starlab Barcelona, Bitbrain, Neuroelectronics y Starlab.

- **Modelo de Negocio:**

Se propone un modelo de negocio basado en licencias de software. Los fabricantes de BCI podrían adquirir licencias comerciales del software para integrarlo en sus productos o servicios. Las licencias podrían estructurarse en diferentes niveles de funcionalidad y soporte, permitiendo adaptarse a las necesidades específicas de cada fabricante.

- **Estrategia de Comercialización:**

La estrategia de comercialización incluiría demostraciones técnicas, pruebas piloto y colaboraciones estratégicas con fabricantes de BCI. Se realizarían participaciones y colaboraciones en conferencias y eventos de la industria, así como la utilización de plataformas online para promover el software y captar el interés de potenciales clientes.

Costes del desarrollo del proyecto:

Concepto	Tarifa en euros
Personal	
2500 euros Equipo informático	2500 euros
Infraestructura	500 euros
Total	5500 euros

Tabla A.4: Costes del desarrollo

Viabilidad legal

Tipos de licencias para el software y bibliotecas instalado:

- **BSD (Berkeley Software Distribution):**

BSD es una licencia permisiva de código abierto que permite el uso, modificación y distribución del software con pocas restricciones. Requiere que se mantengan los avisos de copyright y las cláusulas de la licencia..

- **ASF (Apache Software License):**

ASF 2.0 permite el uso, modificación y distribución del software, otorgando derechos de patente explícitos. Requiere mantener los avisos de copyright y proporcionar una NOTIFICACIÓN de cambios significativos.

- **Código abierto (Open source):**

Las licencias de código abierto permiten el acceso, uso, modificación y distribución libre del software. Estas licencias fomentan la colaboración y la transparencia en el desarrollo de software, garantizando la libertad de uso y mejora.

El estudio de viabilidad ha demostrado que el desarrollo del software de análisis de datos EEG presenta una sólida oportunidad económica y cumple con los requisitos legales necesarios para su comercialización.

La aplicación potencial del software a otros sistemas BCI y la flexibilidad del modelo de negocio basado en licencias ofrecen un camino claro hacia la monetización y la expansión del proyecto.

l	Software	Licencia	Restricciones
	Anaconda	Gratuita	Uso personal.
	Jupyter Notebook	Libre	Código abierto
	Python	Libre permisiva	BSD
	Numpy	Libre permisiva	BSD
	Scipy	Libre permisiva	BSD
	Pandas	Libre permisiva	BSD
	Matplotlib	Libre permisiva	BSD
	Seaborn	Libre permisiva	BSD
	Sklearn	Libre permisiva	BSD
	Tensorflow	Libre permisiva	ASF
	Ipywidgets	Libre	Código abierto

Tabla A.5: Licencias Software

A continuación se presenta una tabla con algunas de las principales bibliotecas y herramientas de software utilizadas en el proyecto, junto con el tipo de licencia y las restricciones asociadas:

Para recuperar la inversión estimada de 5500 euros y garantizar la viabilidad económica del proyecto, se puede optar por vender licencias de software con diferentes niveles de funcionalidad y soporte.

Esta sería una propuesta para estructurar las licencias y sus costes:

■ **Licencia Básica:**

Descripción: Incluye acceso al estudio con funcionalidades básicas de análisis de señales EEG.

Soporte: Limitado a soporte por correo electrónico.

Coste: 100 euros por licencia.

■ **Licencia Profesional:**

Descripción: Incluye todas las funcionalidades de la Licencia Básica, más características avanzadas de análisis y procesamiento de datos.

Soporte: Soporte técnico por correo electrónico y acceso a actualizaciones menores.

Coste: 500 euros por licencia.

■ **Licencia Empresarial:**

Descripción: Incluye todas las funcionalidades de la Licencia Profesional, integración con otros tipos de modelos que se quieran utilizar y soporte para proyectos a gran escala.

Soporte: Soporte técnico completo, incluyendo asistencia telefónica y acceso a todas las actualizaciones.

Coste: 2500 euros por licencia.

Si se llega a publicitar correctamente este software podría ser altamente rentable si se llega a profesionalizar hacia licencias empresariales.

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.

B.2. Objetivos generales

Los objetivos principales de este TFG son los siguientes:

- **Facilitar el procesamiento de datos EEG.** Generados mediante un 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 sistema informático desarrollado) 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 datos:**
 - **Descripción:** El sistema debe permitir almacenar cada proceso tanto de manera temporal como permanente de los resultados experimentos y de la creación de archivos intermedios.
 - **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 en el diagrama:

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 validación 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	Investigador
CU-007	Almacenamiento y Recuperación datos
Versión	1.0
Autor	José Luis Pérez Gómez
Requisitos asociados	RF-007
Descripción	: El sistema almacena y recupera los archivos intermedios o 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.

C.2. Diseño de datos

1. 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 preprocesado 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.

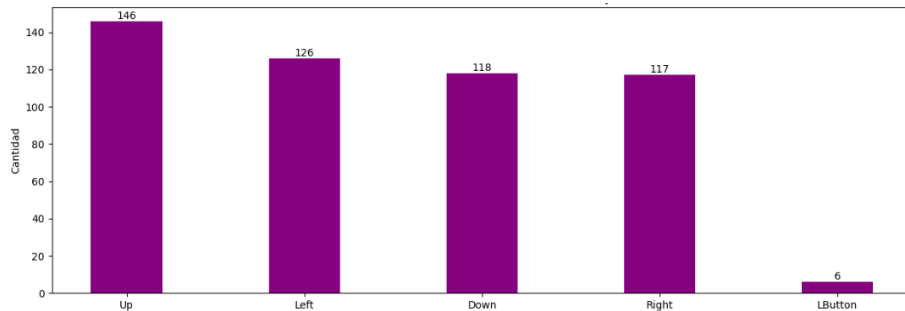


Figura C.1: Cantidad de elementos para la característica Key

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

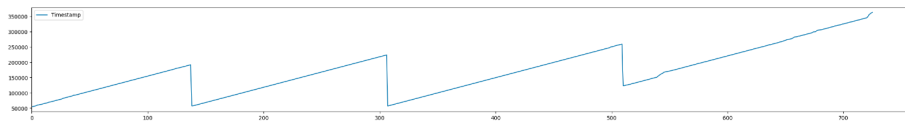


Figura C.2: Análisis característica Timestamp del conjunto de datos

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

2. Almacenamiento de datos:

- **Almacenamiento temporal:**

Durante el análisis, los DataFrames se almacenan en memoria.

- **Almacenamiento Permanente:**

Los resultados intermedios o finales se guardan en archivos CSV o en otros formatos como keras para un acceso más eficiente.

3. Modelo Entidad-Relación (ERD):

- **Entidad: Archivo CSV.**

Representa el archivo CSV cargado al sistema.

- **Entidad: Registro EEG.**

Cada fila del CSV correspondiente a un registro de señales EEG que crearan el dataframe a analizar.

- **Entidad: Preprocesado.**

Representa la transformación necesaria para la ejecución de los modelos.

- **Entidad Modelo.**

Representa cada modelo entrenado por cada conjunto de datos. Los datos que se guardan son de tipo csv con los resultados de la métrica Tasa de acierto.

- **Entidad Resultado:.**

Almacena los resultados obtenidos de los modelos.

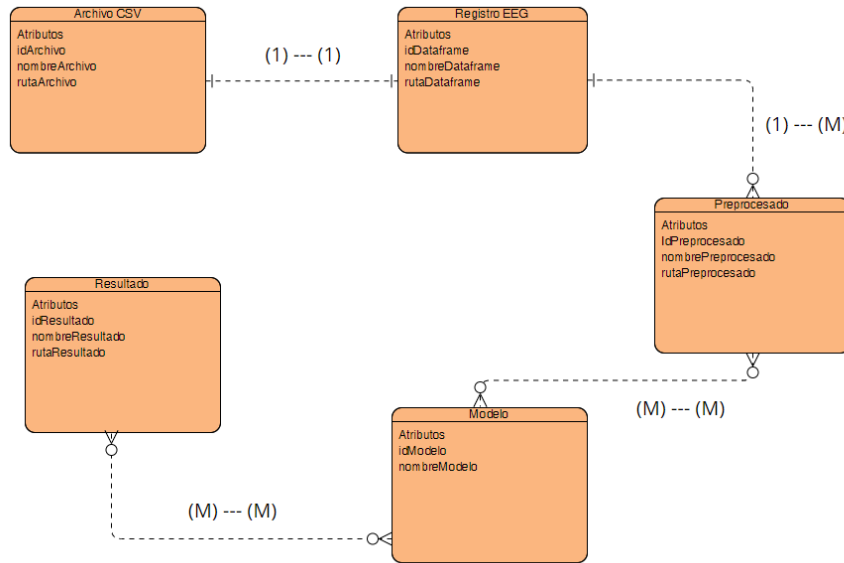


Figura C.3: Diagrama Entidad Relación

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.

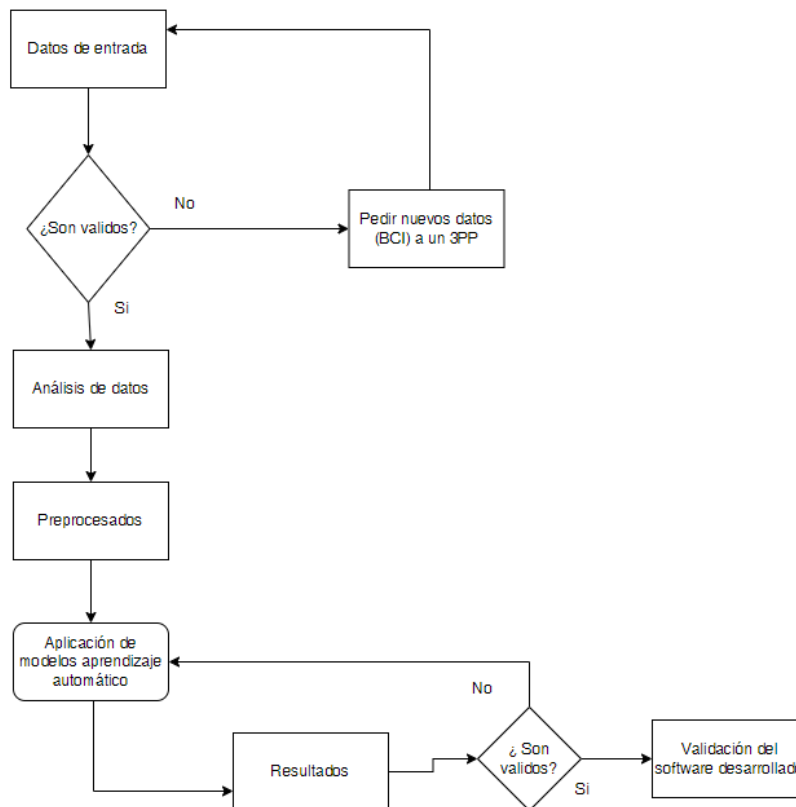


Figura C.4: Diagrama de interacción entre módulos

Manejo de datos y proprocesado:

- **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.

C.4. Diseño arquitectónico

El sistema está diseñado de manera modular, con notebooks secundarios encargados de tareas específicas. El notebook principal orquesta la ejecución de los demás notebooks, asegurando que cada etapa del proceso se complete antes de pasar a la siguiente. Los datos y resultados se almacenan de manera eficiente para facilitar su análisis y reutilización.

Este diseño integral garantiza una gestión y análisis de datos estructurados y eficientes, facilitando el desarrollo y la evaluación de modelos predictivos para datos EEG. Además, cada notebook secundario puede ejecutarse de manera independiente siguiendo un orden predefinido, lo que ofrece flexibilidad en la realización de análisis específicos o la repetición de etapas del proceso según sea necesario.

Esta adición enfatiza la versatilidad del sistema al permitir ejecuciones independientes de los notebooks secundarios, mientras se mantiene la integridad y la eficiencia del flujo de trabajo general.

La organización del sistema sería de esta manera:

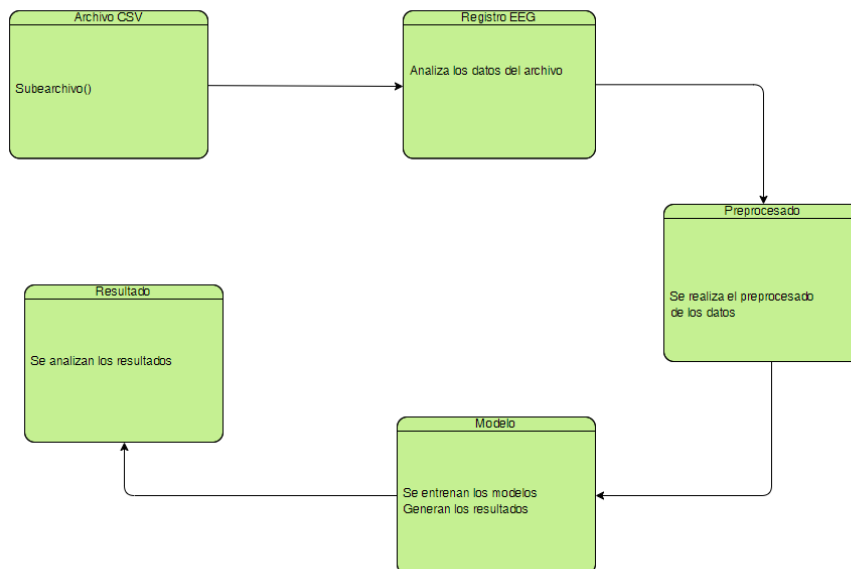


Figura C.5: Organización del sistema

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 imágenes 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:

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

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. Aíslo 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.

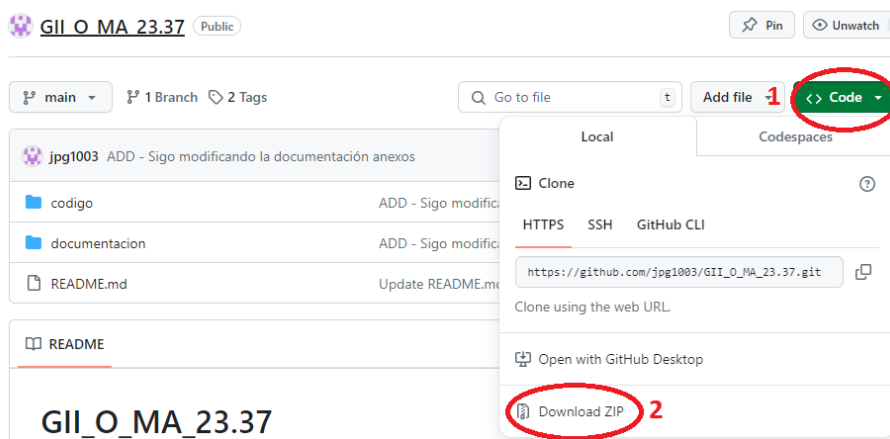


Figura D.1: Descarga repositorio GitHub

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.

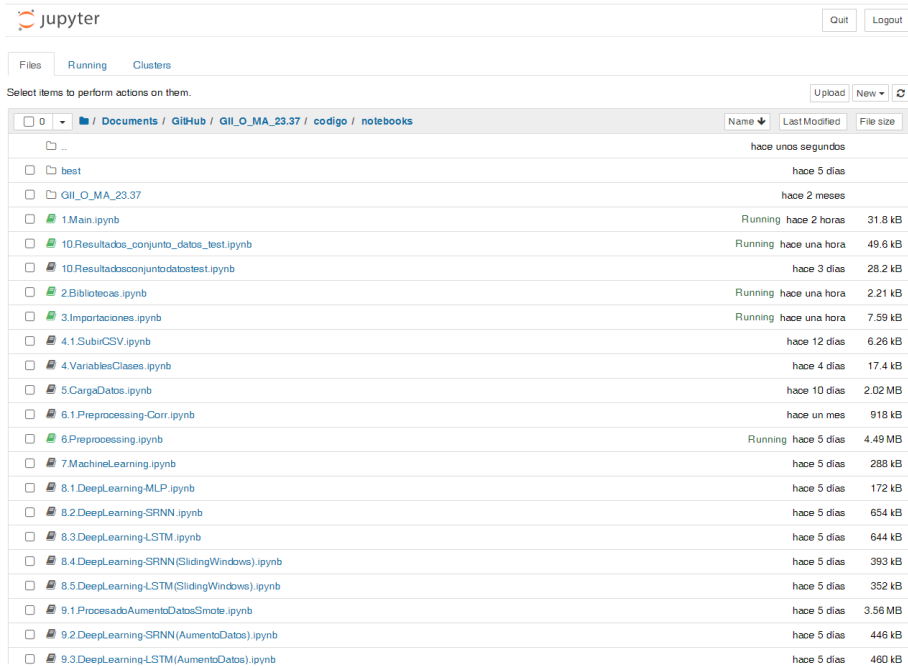
Instalación de software requerido:

■ 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 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 en un solo lugar.

D.5. Pruebas del sistema

Se han realizado varias pruebas de clasificación con el sistema probando con varios modelos de aprendizaje automático:

■ Pruebas simples con modelos:

Los resultados obtenidos para todos los modelos implementados (KNN, Arboles de decisión, Random Forest, MLP, RNN y LSTM) han sido los siguiente:

	KNN_TEST	TREE_TEST	RANDOM_TEST	MLP_TEST	RNN_TEST	LSTM_TEST
Segmento 1	0.692308	0.538462	0.538462	0.615385	0.615385	0.538462
Segmento 2	0.500000	0.375000	0.312500	0.375000	0.312500	0.250000
Segmento 3	0.400000	0.200000	0.200000	0.300000	0.350000	0.350000
Segmento 4	0.200000	0.300000	0.150000	0.300000	0.350000	0.350000
All Segmentos after	0.279412	0.338235	0.250000	0.367647	0.308824	0.338235
All Segmentos before	0.250000	0.308824	0.235294	0.308824	0.294118	0.323529

Figura D.3: Ejemplo resultados Tasa de acierto primer experimento comparativo para subconjunto Test

.9

Las filas corresponden a los conjuntos de datos siguientes:

Segmento del 1 al 4 son conjuntos de datos individuales por sesión o condiciones experimentales. Comentados en la sección Diseño de datos / Division del conjunto de datos.

All Segmentos after es un conjunto de datos que se crea después de haber sido escalados los segmentos individualmente y unificados en un solo conjunto de datos.

All Segmentos before es un conjunto de datos que se crea escalando todo el conjunto de datos global sin haber escalado por separado los segmentos

Las columnas de la tabla corresponden a los siguientes experimentos:

KNN-TEST: Algoritmo de clasificación KNN y validación por Holdout.

TREE-TEST: Algoritmo de clasificación por Árboles de decisión y validación por Holdout.

RANDOM-TEST: Algoritmo de clasificación por random forest y validación por Holdout.

MLP-TEST: Clasificación por red neuronal MLP.

RNN-TEST: Clasificación por red neuronal RNN.

LSTM-TEST: Clasificación por red neuronal LSTM.

Se puede ver a simple vista que los resultados para la métrica Tasa de acierto no han sido buenos para cualquiera de los modelos comparados.

Se han conseguido valores en Tasa de acierto de como máximo el 69,23 en uno de los segmentos individuales y del 36,76 en uno de los conjunto de datos completos.

■ Pruebas con aumento de datos:

La inclusión de datos sintéticos ha demostrado ser beneficioso para mejorar las tasas de acierto en comparación con el uso exclusivo de datos reales.

Este enfoque ha permitido aumentar la diversidad y la cantidad de datos disponibles para el entrenamiento, lo cual es crucial cuando los datos reales son limitados.

	RNN_RS_TEST	LSTM_RS_TEST
All Segmentos after	0.313333	0.463333
All Segmentos before	0.290000	0.346667

Figura D.4: Ejemplo resultados Tasa de acierto experimento con aumento de datos a Test

.6

Las columnas de la tabla corresponden a los siguientes experimentos:

RNN-RS-TEST: Clasificación por red neuronal RNN con aumento de datos.

LSTM-RS-TEST: Clasificación por red neuronal LSTM con aumento de datos.

Estos resultados en comparación con los anteriores son un 10 por ciento mayores pero aun no son buenos resultados para la métrica Tasa de acierto.

■ Pruebas con ventanas deslizantes:

Los datos recogidos tras utilizar las ventanas temporales con datos reales y con el aumento de datos sintéticos han sido los siguientes:

	RNN_SW_TEST	LSTM_SW_TEST
All Segmentos after	0.222222	0.222222
All Segmentos before	0.277778	0.222222

Figura D.5: Ejemplo resultados Tasa de acierto experimento ventanas temporales a Test

.6

	RNN_RS_SW_TEST	LSTM_RS_SW_TEST
All Segmentos after	0.627273	0.836364
All Segmentos before	0.545455	0.609091

Figura D.6: Ejemplo resultados Tasa de acierto experimento ventanas temporales con aumento de datos a Test

.6

Las columnas de la tabla corresponden a los siguientes experimentos:

RNN-SW-TEST: Clasificación por red neuronal RNN con ventanas deslizantes.

LSTM-SW-TEST: Clasificación por red neuronal LSTM con ventanas deslizantes.

RNN-RS-SW-TEST: Clasificación por red neuronal RNN con ventanas deslizantes y aumento de datos.

LSTM-RS-SW-TEST: Clasificación por red neuronal LSTM con ventanas deslizantes y aumento de datos.

El impacto combinado de las dos técnicas para el análisis hace que el aumento en el porcentaje de la tasa de acierto, para el conjunto de datos estandarizado por segmentos y luego unificado en un solo conjunto de datos, alcanza hasta el 83.64 por ciento.

■ **Pruebas con aumento datos contra datos Reales:**

La comparación directa entre conjuntos con aumento de datos y subconjuntos Test de exclusivamente datos reales ha llegado a mostrar que los resultados pueden llegar a superar significativamente, el rendimiento de Tasa de acierto que de los que solo eran datos reales.

Esto sugiere que los datos sintéticos pueden capturar mejor la variabilidad inherente en las señales EEG y mejorar la capacidad de generalización de los modelos.

Unificando de nuevo las técnicas de aumento de datos y ventanas deslizantes se ha llegado hasta el 64,17 por ciento en Tasa de Acierto, algo esperanzador para futuras líneas de investigación puesto que mejora en un casi 30 por ciento de Tasa de acierto con los modelos comparativos con solo datos reales.

	RNN_RS_TEST	LSTM_RS_TEST	RNN_RS_SW_TEST	LSTM_RS_SW_TEST
Test a datos reales	0.446154	0.484615	0.525000	0.641667

Figura D.7: Ejemplo resultados Tasa de acierto experimento aumento de datos a datos reales Test

0.6

Las columnas de la tabla corresponden a los siguientes experimentos:
RNN-RS-TEST: Clasificación por red neuronal RNN con aumento de datos.

LSTM-RS-TEST: Clasificación por red neuronal LSTM con aumento de datos.

RNN-RS-SW-TEST: Clasificación por red neuronal RNN con ventanas deslizantes y aumento de datos.

LSTM-RS-SW-TEST: Clasificación por red neuronal LSTM con ventanas deslizantes y aumento de datos.

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

La instalación se puede realizar manualmente o través de una máquina virtual generada para el proyecto:

- **Instalación de software mediante máquina virtual:**

- Requisitos mínimos para ejecutar la máquina virtual:**

- 4 vCPUs y 4GB de RAM (Se recuerda que estos recursos mínimos son para la máquina virtual, para el sistema operativo que albergue la máquina virtual se ha de reservar unos mínimos recursos de computo)

- Si se puede asignar más CPU y RAM a la máquina virtual las ejecuciones del sistema tardarán menos tiempo en ejecutarse.

- Se ha generado una maquina virtual para evitar posibles problemas de incompatibilidad de versiones de software.

- Para crear la maquina he utilizado virtualbox. [6]

- Para poder instalarla hay que tener instalado el software de virtualización VirtualBox 7.0, descargable desde la pagina web oficial de VirtualBox. [6]

- Una vez instalado VirtualBox 7.0 se han de seguir los siguientes pasos:

- **Descarga archivo AnalisisEEG-TFG.ova:**

- Con este archivo de tipo ova, se puede general la máquina virtual todas las veces que se quiera.

- **Instalar la máquina virtual:** Hacer doble clic sobre el archivo .ova y seguir los siguientes pasos:



Figura E.1: Inicio instalación máquina virtual

Pulsar en Terminar y espere que acabe la importación:

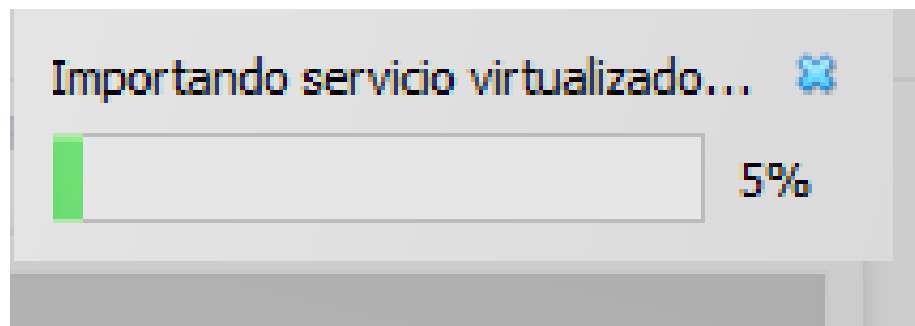


Figura E.2: Importación máquina virtual

.5

Una vez importada, ya se puede arrancar con el botón en forma de flecha verde (Iniciar):

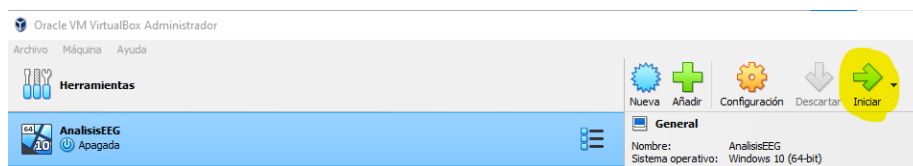


Figura E.3: Arrancar máquina virtual

.5

Una vez arrancada y con acceso al sistema operativo Windows de la máquina virtual las credenciales de la maquina virtual son los siguientes:

Usuario: AnalisisEEG **Contraseña:** Por seguridad, se proporcionará en el enlace al archivo .ova

Una vez aprobadas las credenciales, ya se pueden seguir las indicaciones del Manual de usuario.

■ Instalación de software manualmente:

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

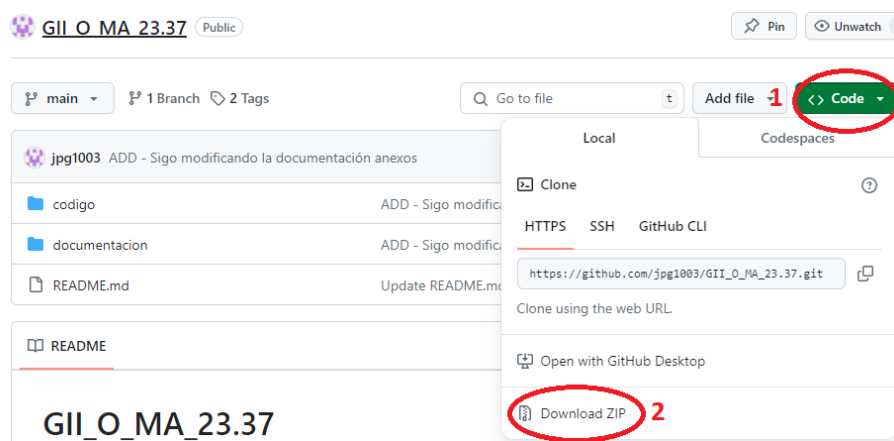


Figura E.4: Descarga repositorio GitHub

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. [\[1\]](#)
 - Descargar la versión más reciente de Python 3.x. para el sistema operativo compatible.
 - Instalar el archivo descargado.

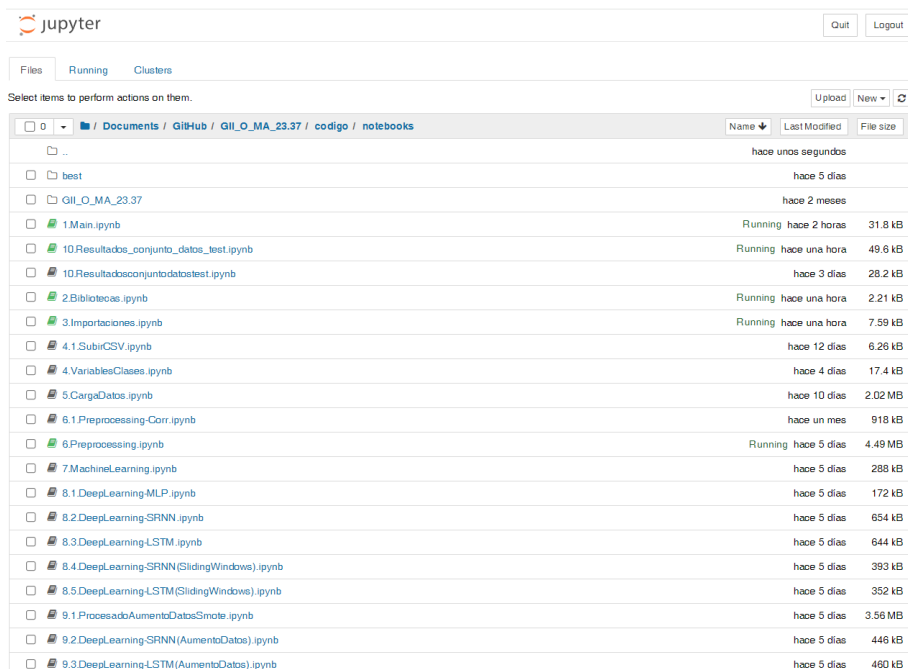
Una vez instalado el paquete de aplicaciones Anaconda, en el sistema operativo aparecerá una aplicación llamada **Jupyter Notebook**.

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:



Name	Last Modified	File size
..	hace unos segundos	
best	hace 5 días	
Gil_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 E.5: Listado de Notebooks en Jupyter Notebook

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

■ Automática:

Esta ejecución dura aproximadamente con los recursos mínimos para la maquina virtual.

- 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 sería el notebook ya abierto:

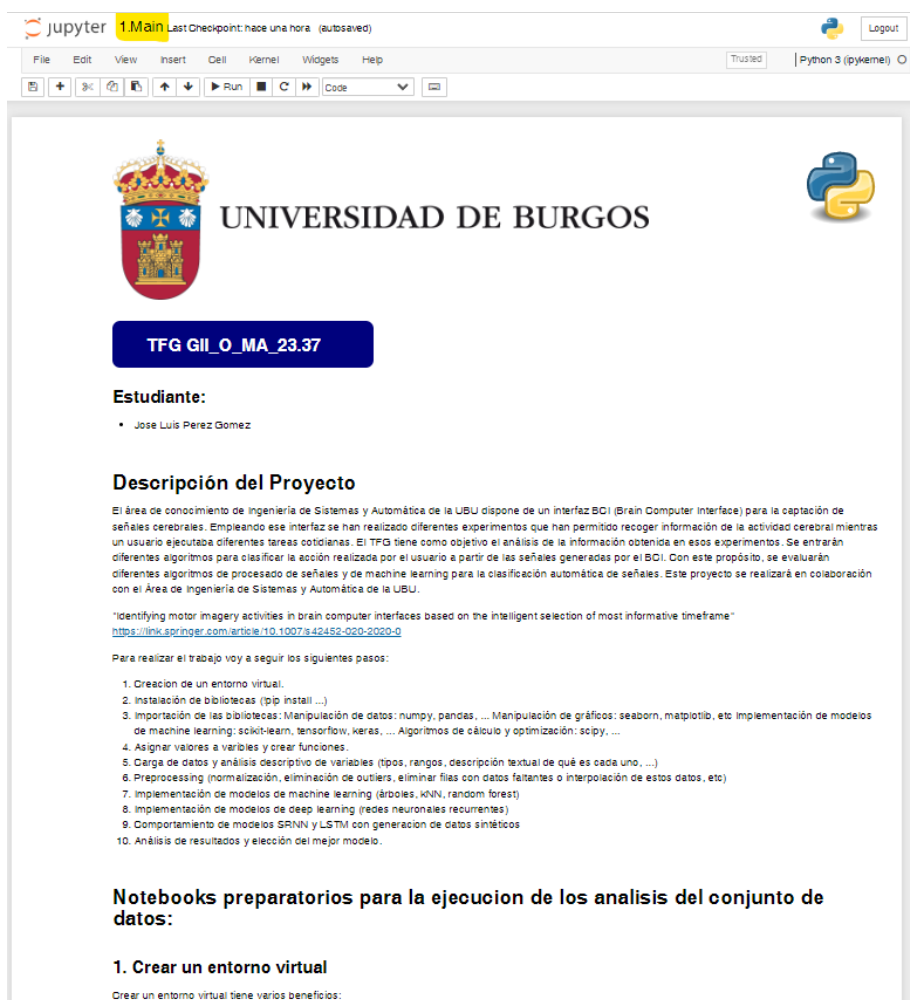


Figura E.6: Ejemplo Notebook Main

- **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 declaración 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 análisis 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 podrá 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:

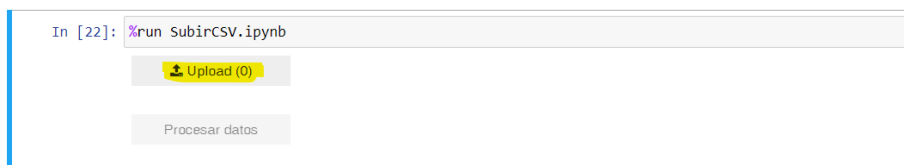


Figura E.7: Botón Upload

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.

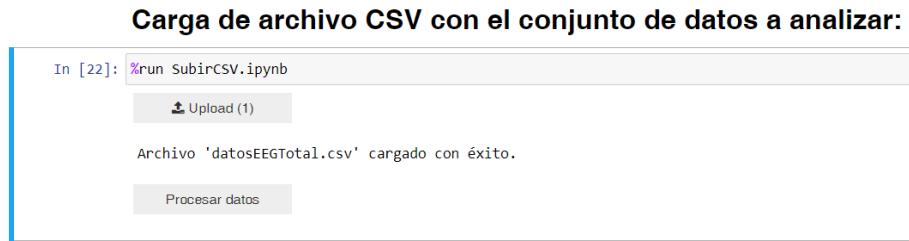


Figura E.8: Subida correcta de archivo csv.

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

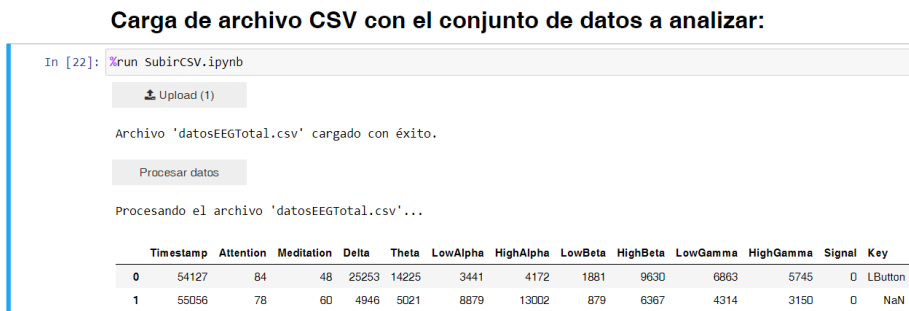


Figura E.9: Datos después de subirlos al entorno correctamente.

- **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-fold 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) 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.Resultadosconjuntodatostest.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.10: 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.11: Ejemplo instalación manual

IMPORTANTE: Si se quisiera ejecutar el proyecto en manera AUTOMATICA 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.12: 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.13: 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.14: 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:

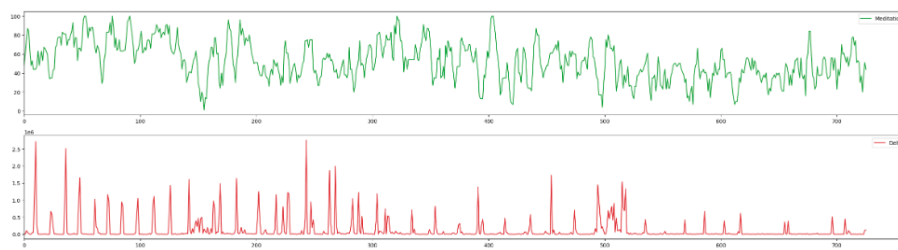


Figura E.15: Ejemplo gráfica en análisis de datos

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, eliminación de outliers, estandarización de datos. Si el usuario dispone de conocimiento para modificar este código 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 sería consultar con el desarrollador.

Los resultados que se imprimirán son resultados de tipo tasa de acierto, comparándola 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.16: Ejemplo tabla comparativa de tasa de acierto entre modelos

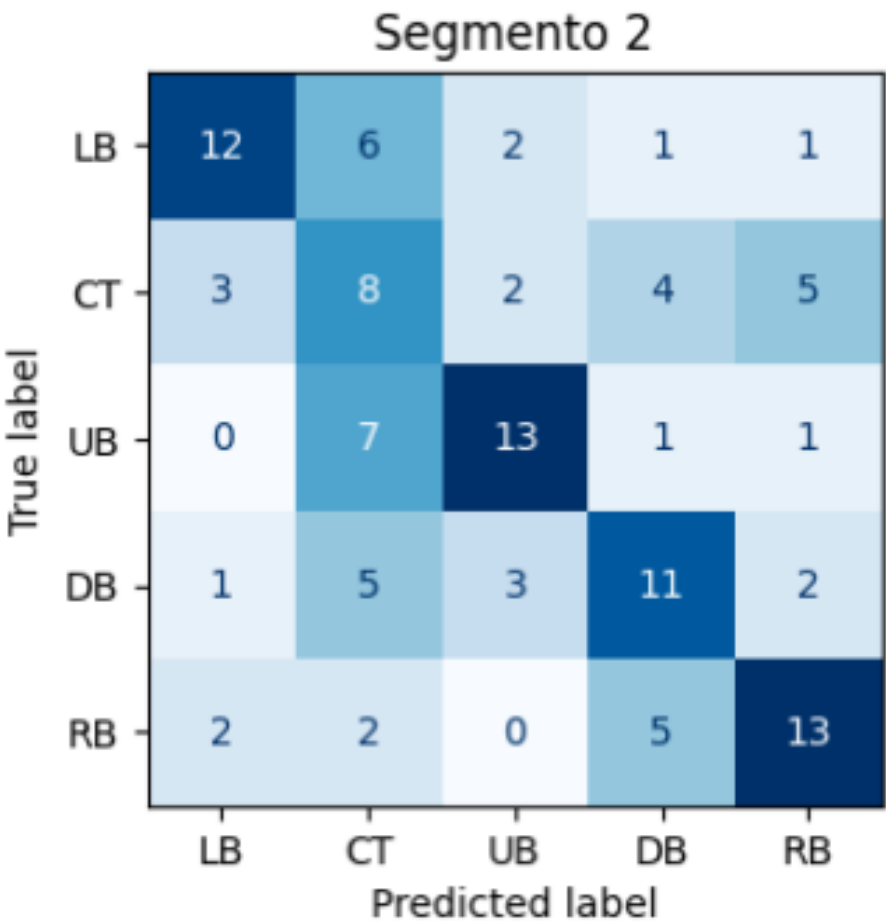


Figura E.17: Ejemplo matriz de confusión para un modelo

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.18: Ejemplo tabla comparativa de resultados

Después de la ejecución automática o manual se ha de realizar la interpretación de los datos obtenidos por el usuario.

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

Durante el desarrollo de este proyecto, he tenido la oportunidad de profundizar en los principios de sostenibilidad delineados por CRUE, los cuales proporcionan un marco esencial para integrar prácticas sostenibles en nuestras actividades académicas y prácticas. Esta experiencia no solo ha enriquecido mi comprensión teórica sobre el análisis de datos EEG, sino que también ha fortalecido mis habilidades prácticas para abordar los desafíos ambientales y sociales desde una perspectiva integradora y proactiva.

Desde el inicio del proyecto, a través de la investigación y el análisis crítico, he llegado a apreciar la interconexión entre los aspectos sociales y económicos de la sostenibilidad. Este enfoque me ha permitido entender que la sostenibilidad no se limita únicamente a la conservación del medio ambiente, sino que también implica asegurar la equidad social y promover la viabilidad económica a largo plazo.

F.2. Aplicación práctica

Una parte central de mi aprendizaje ha sido la aplicación práctica de los principios de sostenibilidad en este proyecto en concreto. A medida que avanzaba en la elaboración de estrategias y soluciones, tuve la oportunidad de evaluar el ciclo de vida de productos y procesos, identificar áreas de mejora y proponer medidas para minimizar nuestro impacto ecológico. Este

enfoque práctico no solo me ha equipado con habilidades técnicas, sino que también me ha inculcado un sentido de responsabilidad hacia la integración de prácticas sostenibles en mi vida personal y profesional.

Además, el uso de licencias gratuitas o de uso libre permisivo para las herramientas y recursos utilizados en este estudio subraya el compromiso con la accesibilidad y la equidad. Esta elección no solo facilita la replicabilidad y la colaboración, sino que también promueve un acceso más amplio a los resultados y beneficios del proyecto.

F.3. Pensamiento crítico

Otro aspecto clave de mi desarrollo ha sido el fomento de un pensamiento crítico y creativo en relación con la sostenibilidad. A través del análisis de diversas perspectivas, he aprendido a cuestionar supuestos establecidos y a explorar soluciones innovadoras que puedan contribuir significativamente a la sostenibilidad global. Este proceso me ha enseñado a pensar de manera estratégica y a considerar no solo los desafíos inmediatos, sino también las implicaciones a largo plazo de nuestras acciones y decisiones.

F.4. Impacto Personal

Una faceta fascinante de este proyecto ha sido explorar cómo la tecnología de análisis de datos EEG puede mejorar significativamente la calidad de vida de las personas con discapacidades, como aquellas personas con discapacidad motora que utilizan sillas de ruedas. Al estudiar la posibilidad de utilizar conjunto de datos con señales EEG para ser utilizadas en la vida diaria o incluso en aplicaciones de entretenimiento como videojuegos accesibles, se puede llegar a vislumbrar un futuro donde la tecnología no solo facilita la inclusión, sino que también empodera a individuos con capacidades diversas para participar plenamente en la sociedad.

Este aspecto del proyecto no solo me ha inspirado personalmente, sino que también refuerza la importancia de considerar las implicaciones éticas y sociales de la tecnología en nuestro trabajo. La promoción de soluciones innovadoras y accesibles no solo beneficia a individuos con discapacidades, sino que también enriquece nuestra comprensión colectiva de cómo la tecnología puede ser utilizada para el bien común y el progreso social.

Este proyecto ha tenido un impacto profundo en mi compromiso personal. Me ha inspirado a adoptar un enfoque más consciente y reflexivo en mi vida

diaria, buscando constantemente oportunidades para promover prácticas sostenibles en mi entorno. Además, me ha motivado a explorar carreras y oportunidades profesionales que me permitan contribuir de manera significativa a la construcción de un futuro más sostenible y equitativo para todos.

F.5. Conclusión

El proyecto no solo ha ampliado mis conocimientos y habilidades en sostenibilidad y tecnología de datos EEG, sino que también ha transformado mi perspectiva personal y profesional. Estoy convencido de que las competencias adquiridas aquí no solo serán valiosas en mi desarrollo académico y profesional, sino que también me capacitarán para enfrentar los desafíos globales con una mentalidad innovadora y comprometida con el cambio positivo.

Espero seguir explorando y aplicando estas competencias a lo largo de mi vida, contribuyendo activamente a la creación de un mundo más sostenible y resiliente para las generaciones futuras.

Bibliografía

- [1] Anaconda. Descarga anaconda. <https://www.anaconda.com/download/success>.
- [2] José Luis Pérez Gómez. Analisis eeg. <https://github.com/jpg1003/AnalisisEEG>.
- [3] Plataforma Kaggle. Trabajo con series temporales. <https://www.kaggle.com/learn/time-series>.
- [4] Plataforma Kaggle. Tutorial de inicio de pandas. <https://www.kaggle.com/learn/pandas>.
- [5] Plataforma Kaggle. Visualización de datos. <https://www.kaggle.com/learn/data-visualization>.
- [6] VirtualBox.org. Welcome to virtualbox.org. <https://www.virtualbox.org/>.