



UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Grado en Ingeniería Electrónica, Industrial y Automática

**TRABAJO FIN DE GRADO**

# Desarrollo de una aplicación con interfaz multiplataforma para espacios clínicos sobre la progresión de enfermedades neurodegenerativas

Alberto Álvarez López

Tutor: Carlos Platero Dueñas
Departamento: Ingeniería Eléctrica, Electrónica, Automática y Física Aplicada

Madrid, Junio, 2024



UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Grado en Ingeniería Electrónica y Automática Industrial

**TRABAJO FIN DE GRADO**

# Desarrollo de una aplicación con interfaz multiplataforma para espacios clínicos sobre la progresión de enfermedades neurodegenerativas

Firma Autor

Firma Cotutor (si lo hay)

Firma Tutor

**Título:** Desarrollo de una aplicación con interfaz multiplataforma para espacios clínicos sobre la progresión de enfermedades neurodegenerativas
**Autor:** Alberto Álvarez López
**Tutor:** Carlos Platero Dueñas

# EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día ..... de ..... de ... en ....., en la Escuela Técnica Superior de Ingeniería y Diseño Industrial de la Universidad Politécnica de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO PRESIDENTE

# Agradecimientos

En primer lugar, me gustaría expresar mi más sincero agradecimiento a mis padres y a mi hermana por el apoyo constante e incondicional que me han brindado cada día durante mi periodo universitario. Ellos han sido mi principal vía de escape cuando he pasado por los peores momentos, y nunca han fallado al darme el empujón que necesitaba para superar cualquier mínima dificultad que me haya encontrado. Gracias por confiar en mí de forma incondicional.

Quiero agradecer también a todos los amigos que no han dejado de estar a mi lado ni un solo día desde que les conozco y al maravilloso grupo que he tenido de la suerte de conocer en esta universidad. En especial, me gustaría agradecerle su apoyo y confianza a Luis y Sergio, mis eternos compañeros de proyecto que, aunque no han podido participar en la realización de este, no han dejado de apoyarme y de hacerme más ameno todo este proceso en ningún momento.

Por último, y no por ello menos importante, quiero agradecer a Carlos Platero por haber sido un tutor excepcional. Estoy muy agradecido de haber tenido la oportunidad de hacer este trabajo, considero que nunca había aprendido tanto en un periodo tan corto de tiempo y además me ha sido de gran ayuda como orientación para lo que quiero hacer en el futuro. Terminar este proyecto no habría sido posible sin su apoyo y orientación semana a semana.

En resumen, gracias a todas las personas que han confiado en mí durante esta etapa de mi vida.

# Resumen

La enfermedad de Alzheimer es una afección neurodegenerativa progresiva que provoca deterioro cognitivo, pérdida de memoria y cambios en el comportamiento. Es de gran importancia en la actualidad debido al envejecimiento de la población global, lo que lleva a un aumento en el número de personas afectadas. Este incremento plantea significativos desafíos médicos, económicos y sociales, destacando la necesidad urgente de mejores diagnósticos, tratamientos y estrategias de prevención.

Una creciente línea de investigación al respecto es la de los modelos predictivos, que utilizan la información de biomarcadores para generar predicciones sobre el avance de enfermedades neurodegenerativas en un sujeto dado. Sin embargo, mientras que estos modelos se basan en tecnologías como algoritmos paramétricos o el Machine Learning (ML), que pertenecen al ámbito de la ingeniería, los estudios de biomarcadores y el entendimiento más profundo de enfermedades neurodegenerativas como el Alzheimer corresponden a campos como la neurología y la neurobiología. Esto supone un grave entorpecimiento a las investigaciones, ya que los investigadores de estas enfermedades requieren del apoyo constante de profesionales del ámbito de la ingeniería.

Para ayudar a combatir este obstáculo, el presente proyecto propone el desarrollo de una aplicación multiplataforma que, a través de una intuitiva interfaz gráfica de usuario (GUI), facilite a los investigadores el manejo de los modelos predictivos de una forma sencilla e independiente.

**Palabras clave:** Aplicación, GUI, Multiplataforma, Enfermedad de Alzheimer, Modelos predictivos, Biomarcadores.

# Abstract

Alzheimer's disease is a progressive neurodegenerative condition that causes cognitive decline, memory loss, and behavioral changes. It is of great importance today due to the aging global population, which leads to an increase in the number of affected individuals. This increase presents significant medical, economic, and social challenges, highlighting the urgent need for better diagnostics, treatments, and prevention strategies.

A growing line of research in this area involves predictive models that use biomarker information to generate predictions about the progression of neurodegenerative diseases in a given subject. However, while these models are based on technologies such as parametric algorithms or Machine Learning (ML), which belong to the field of engineering, the studies of biomarkers and a deeper understanding of neurodegenerative diseases like Alzheimer's belong to fields such as neurology and neurobiology. This poses a significant impediment to research, as investigators of these diseases require the constant support of professionals from the engineering field.

To help address this obstacle, the present project proposes the development of a cross-platform application that, through an intuitive graphical user interface (GUI), facilitates researchers' handling of predictive models in a simple and independent manner.

**Keywords:** Application, GUI, Cross-Platform, Alzheimer's Disease, Predictive Models, Biomarkers.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

## 1.1. La enfermedad de Alzheimer

La enfermedad de Alzheimer (AD), una forma común de demencia neurodegenerativa, es un trastorno cerebral progresivo e irreversible que afecta principalmente a las funciones cognitivas y la memoria. Esta enfermedad se caracteriza por la acumulación anormal de placas de proteína beta-amiloide y ovillos neurofibrilares en el cerebro, lo que conduce a la degeneración y muerte de las células nerviosas.

Esta degeneración neuronal afecta principalmente a regiones específicas del cerebro asociadas con la memoria, el aprendizaje y otras funciones cognitivas. A medida que la enfermedad avanza, estas áreas se ven comprometidas, lo que resulta en una pérdida gradual de la memoria y del pensamiento, así como en cambios en el comportamiento y la personalidad del individuo afectado.

Uno de los síntomas tempranos más comunes de la enfermedad de Alzheimer es la pérdida de memoria, especialmente la pérdida de memoria reciente. A medida que progresa la enfermedad, los síntomas se vuelven más graves e incluyen confusión, desorientación, dificultad para comunicarse, cambios de humor y problemas para realizar tareas simples de la vida diaria. Además, pueden surgir síntomas neuropsiquiátricos como la depresión, la ansiedad y la agitación, que agravan el estado del paciente.

Aunque actualmente no existe cura para la enfermedad de Alzheimer, hay tratamientos disponibles que pueden ayudar a aliviar los síntomas y mejorar la calidad de vida de los pacientes. Estos tratamientos pueden incluir medicamentos para ayudar a controlar los síntomas cognitivos y del comportamiento, así como terapias de apoyo para brindar soporte emocional y práctico a los pacientes y sus familias. Las terapias ocupacionales y las intervenciones cognitivas también pueden ser beneficiosas para mantener las habilidades funcionales durante más tiempo.

## 1.2. Impacto socioeconómico de la enfermedad de Alzheimer

El Alzheimer, una enfermedad neurodegenerativa progresiva y irreversible, no solo tiene un impacto devastador en la vida de los pacientes y sus familias, sino que también genera importantes repercusiones económicas y sociales a nivel mundial. Esta enfermedad afecta no solo a la salud y el bienestar de los individuos, sino también a los sistemas de atención médica, los servicios sociales y la economía en general.

En términos económicos, el Alzheimer representa una carga significativa para los sistemas de salud y los gastos asociados con el diagnóstico, tratamiento y cuidado de los pacientes. Los costes médicos directos incluyen consultas médicas, hospitalizaciones, medicamentos y terapias especializadas. Además, los costes indirectos, como la pérdida de productividad laboral tanto de los pacientes como de sus cuidadores, también son considerables. A nivel global, se estima que los costes relacionados con el Alzheimer alcanzan aproximadamente 1,3 billones de dólares anuales, una cifra que se espera que se duplique para 2030 debido al envejecimiento de la población [48].

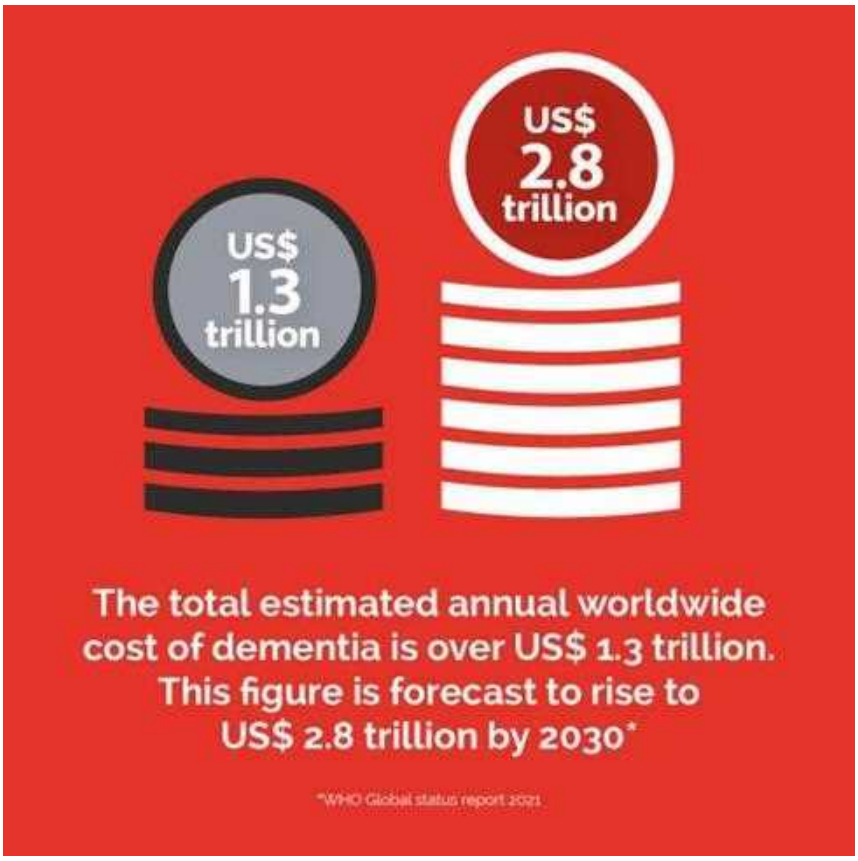


Figura 1.1: Se espera que el coste global de la demencia ascienda hasta los 2,8 billones de dólares anuales para 2030 según la ADI. [5]

En España, la situación es igualmente preocupante. Según datos de la Confederación Española de Alzheimer (CEAFA), se estima que existen más de 1,2 millones de personas que padecen Alzheimer y en total se invierten hasta 36.000 millones

de euros para combatir la enfermedad [11]. Por otro lado, el coste medio anual del cuidado de un paciente de la enfermedad es de 31.890 euros, considerando tanto los costes directos como indirectos, y la mayor parte de estos costes son asumidos por las familias [16]. Este alto coste económico se debe en gran medida a la necesidad de cuidados a largo plazo, que muchas veces recae sobre familiares no remunerados.

A nivel social, el Alzheimer también tiene un impacto profundo en las familias y las comunidades. Los cuidadores familiares, que suelen ser cónyuges, hijos o familiares cercanos, asumen una carga emocional, física y financiera significativa al proporcionar atención a tiempo completo a los pacientes. Esto puede afectar su salud física y mental, sus relaciones personales y sus oportunidades de empleo. Estudios muestran que el 59 % de los cuidadores experimentan estrés significativo, y muchos sufren de depresión y problemas de salud física debido a la carga de cuidado [8].

El Alzheimer puede generar aislamiento social y estigmatización tanto para los pacientes como para sus cuidadores, lo que afecta su calidad de vida y bienestar psicológico. La falta de comprensión y apoyo social agrava el aislamiento, y muchas veces las familias se ven obligadas a asumir solas la totalidad de la carga de cuidado [6]. En este contexto, las redes de apoyo y los servicios comunitarios juegan un papel crucial para mitigar estos efectos negativos.

A nivel macroeconómico, el Alzheimer también afecta la economía en su conjunto. La creciente prevalencia de la enfermedad aumenta la demanda de servicios de atención médica y sociales, lo que puede ejercer presión sobre los sistemas de salud y los presupuestos gubernamentales. En España, la población mayor de 65 años representa un porcentaje significativo de los usuarios de estos servicios, y se espera que este grupo continúe creciendo en las próximas décadas.

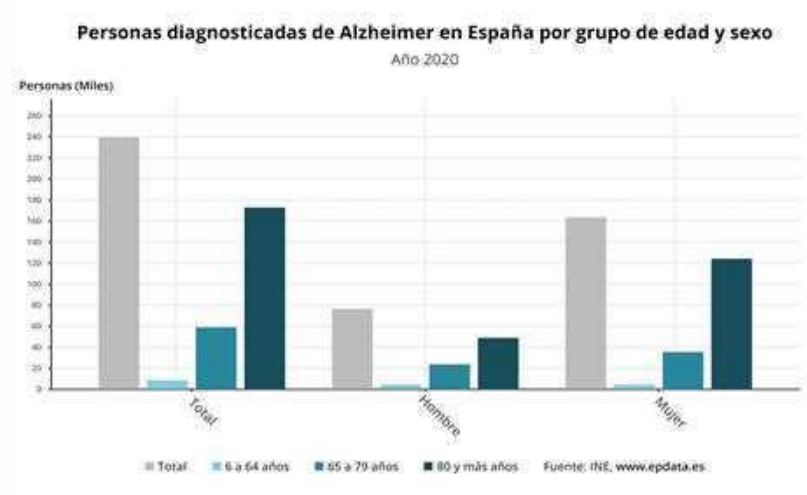


Figura 1.2: Personas diagnosticadas de Alzheimer en España por grupo de edad y sexo en el año 2020. Fuente: INE. [18]

Además, la pérdida de productividad laboral y la disminución de la fuerza laboral activa pueden tener un impacto en el crecimiento económico a largo plazo y en la sostenibilidad de los sistemas de pensiones y seguros sociales. La Organiza-

ción Mundial de la Salud (OMS) estima que, a nivel mundial, la reducción en la productividad laboral debido al Alzheimer y otras demencias asciende a miles de millones de dólares cada año. En países desarrollados, donde el envejecimiento de la población es más pronunciado, esta pérdida de productividad representa un desafío significativo para el desarrollo económico y la sostenibilidad fiscal.

En conclusión, el impacto socioeconómico de la enfermedad de Alzheimer es vasto y complejo, afectando múltiples aspectos de la vida de los individuos y la sociedad en su conjunto. Es crucial que se desarrollen políticas públicas y estrategias de intervención que no solo aborden las necesidades médicas y de cuidado de los pacientes, sino que también proporcionen apoyo a los cuidadores y mitiguen los efectos económicos negativos a largo plazo.

## 1.3. Objetivos

En la actualidad, existen varios modelos predictivos capaces de obtener predicciones cada vez más precisas sobre el avance del AD u otras enfermedades neurodegenerativas, mediante el estudio de biomarcadores en pacientes. Sin embargo, estos modelos suelen ser desarrollados por investigadores del campo de la ingeniería y las matemáticas, ya que se basan en técnicas como el ML (Machine Learning) o algoritmos paramétricos, entre otros. Por otro lado, el estudio de las enfermedades neurodegenerativas pertenece principalmente al ámbito de la neurobiología y la neurología, y los estudios de biomarcadores suelen ser realizados por profesionales de estos campos.

Esta situación presenta un desafío, ya que los profesionales que llevan a cabo estos estudios generalmente carecen de los conocimientos técnicos necesarios para utilizar eficazmente los complejos modelos predictivos, lo que dificulta en gran medida el proceso de obtención y análisis de predicciones de manera eficiente.

A raíz de esta dificultad, surge la motivación de este proyecto: desarrollar una aplicación con una GUI (Graphical User Interface) simple e intuitiva, que permita a los profesionales que realizan estudios de biomarcadores obtener estas predicciones de manera autónoma y sencilla.

Además de la motivación principal, se han fijado una serie de objetivos que pretenden maximizar la calidad del producto final:

- El mecanismo utilizado para la obtención de las predicciones debe tener el menor acoplamiento posible. Debido a la rápida evolución de los modelos con este propósito, se pretende que la aplicación pueda adaptarse fácilmente a cambios en el motor utilizado, en caso de ser necesario. Además, esta medida favorece la actualización y escalabilidad de la aplicación.

- La aplicación desarrollada debe ser multiplataforma. Aunque inicialmente no se cubran gran cantidad de sistemas operativos o entornos, el objetivo es que la aplicación esté preparada para ser ejecutada en más de uno y dejar los cimientos hechos para que, posteriormente, pueda ser migrada a nuevos sistemas con la

mayor facilidad posible. Esto favorece la accesibilidad y permite que el producto llegue a una mayor cantidad de usuarios.

- La interfaz desarrollada debe ser lo más intuitiva posible para el usuario y debe contar con un control de errores robusto. Este objetivo se fija teniendo en cuenta el tipo de usuario al que va dirigida principalmente la aplicación; se aspira a que el usuario se encuentre con el menor número de errores informáticos posible.

Para probar y demostrar que la aplicación final cumple con estos objetivos, se pretende crear distintas versiones de esta que varíen en aspectos como el motor predictivo o el entorno en el que funcionan.

## 1.4. Estructura del documento

A continuación y para facilitar la lectura del documento, se detalla el contenido de cada capítulo:

- Capítulo 1: Introducción. Se trata de una breve introducción al tema del proyecto, indicando el contexto y las motivaciones principales para haberlo llevado a cabo.
- Capítulo 2: Estado del arte. Se indican y explican todos los estudios que han sido consultados para aprender sobre el contexto del proyecto y sobre los métodos necesarios para su realización.
- Capítulo 3: Materiales. Se exponen las herramientas utilizadas para la realización del trabajo y se indica la aportación de cada una.
- Capítulo 4: Métodos. Se presenta y desarrolla toda la metodología llevada a cabo para el desarrollo del proyecto.
- Capítulo 5: Resultados. Se muestran los resultados finales obtenidos tras aplicar los métodos del capítulo cuatro.
- Capítulo 6: Discusiones. Se realiza una valoración general de los resultados obtenidos, comparándolos con los objetivos del proyecto, con otras alternativas del mercado y se sugieren posibles líneas de investigación futuras.
- Anexo. Se añade información adicional para mejorar el entendimiento de algunos aspectos del proyecto.

# Capítulo 2

# Estado del arte

## 2.1. Introducción

En este capítulo se presentan todos los libros y artículos que se han utilizado como fuente de aprendizaje sobre el contexto del estudio y los métodos necesarios para llevarlo a cabo. En la última sección se señalan las aportaciones de cada fuente al proyecto.

## 2.2. Neurodegeneración [21]

El cerebro resulta fascinante al tratarse del órgano más complejo que existe y aprender sobre él supone un inmenso reto biomédico. Entre muchas otras, algunas de las primeras cuestiones que se nos presentan al tratar de aprender sobre este son: ¿por qué perdemos neuronas?, ¿a qué se debe el inevitable deterioro de este órgano? Este concepto se conoce como envejecimiento y el libro se centra en su importancia e influencia con un enfoque particular en enfermedades como el Alzheimer, el Parkinson y la esclerosis lateral amiotrófica. Destaca la importancia del envejecimiento en el desarrollo de estas enfermedades y la necesidad de intervenciones terapéuticas.

Dado el contexto de una sociedad que experimenta un proceso de envejecimiento progresivo, el estudio de las enfermedades relacionadas con este fenómeno se revela como un campo de investigación sumamente atractivo debido a la falta de comprensión completa sobre el mismo. Durante este libro se mencionan posibles mecanismos para promover la longevidad sana, como la restricción calórica, y se discuten los genes y moduladores de la longevidad, que es un ámbito que interesa investigar para mejorar nuestro entendimiento del envejecimiento.

El envejecimiento está relacionado con el daño acumulado en el ADN a lo largo de la vida. La longitud de los telómeros, que son estructuras en los extremos de los cromosomas, y la enzima telomerasa, que puede alargar los telómeros, juegan un importante papel en la longevidad de las células. Los síndromes progeroides y las personas centenarias son modelos de estudio para comprender el envejecimiento. Además de estos factores, la epigenética, que engloba los cambios heredables en la función génica sin cambios en la secuencia del ADN, también ha demostrado influir en gran medida en el envejecimiento de las personas.

Más detalladamente, la epigenética es el estudio de cómo los factores ambientales y de estilo de vida pueden influir en la expresión de los genes. Estos cambios pueden ser causados por factores como la alimentación, el ejercicio, el tabaco o el estrés. Como una posible técnica que se apoya en esto para combatir el envejecimiento celular, se menciona la reprogramación celular, que consiste en la reversión directa de los signos del envejecimiento.

En cuanto a las bases celulares y moleculares detrás del envejecimiento, en primer lugar se explican los tres tipos principales de muerte celular que son relevantes en el contexto de la neurodegeneración: apoptosis, necrosis y autofagia. La apoptosis es un proceso de muerte celular programada que se caracteriza por cambios específicos en la morfología de la célula, como la condensación de la cromatina y la formación de vacuolas en el citoplasma. Este proceso es crucial para el desarrollo normal y el mantenimiento de la homeostasis en los organismos multicelulares.

Por otro lado, la necrosis es un tipo de muerte celular que ocurre como resultado de una lesión o daño celular severo. A diferencia de la apoptosis, la necrosis no es un proceso programado y a menudo conduce a la liberación de sustancias que pueden desencadenar una respuesta inflamatoria, mientras que la autofagia es un proceso celular que degrada y recicla componentes celulares dañados o innecesarios. Aunque la autofagia es esencial para la supervivencia y la homeostasis celular, la disfunción de este proceso puede contribuir a la neurodegeneración.

También se habla del estrés oxidativo, que es una consecuencia del metabolismo aeróbico, es decir, el proceso que utiliza oxígeno para producir energía. Este estrés oxidativo se produce cuando hay un desequilibrio entre la producción de especies reactivas de oxígeno (ROS) y especies reactivas de nitrógeno (RNS) y la capacidad del cuerpo para contrarrestar sus efectos o reparar el daño resultante. El cerebro de los mamíferos es particularmente vulnerable al estrés oxidativo debido a su alto consumo de oxígeno, además, existen numerosos estudios que relacionan el origen y evolución de las enfermedades neurodegenerativas con la excesiva producción de ROS y RNS.

El último elemento de las bases celulares y moleculares que se analiza es la neuroinflamación: esta se refiere a la respuesta inflamatoria del sistema nervioso central como resultado de cualquier tipo de agresión o lesión. Los astrocitos y la microglía son las principales células que, dentro del sistema nervioso, originan y mantienen la respuesta inflamatoria. Durante el envejecimiento, las células gliales, especialmente la microglía, aumentan en número y muestran una elevación en el fenómeno de activación y en la expresión de mecanismos inflamatorios. Este estado de inflamación crónica, también conocido como 'inflamaging', es un buen sustrato para el desarrollo de la neurodegeneración.

A continuación, el libro dedica un elaborado capítulo para explicar en qué consisten las tres enfermedades neurodegenerativas mencionadas previamente:

- Enfermedad de Alzheimer (AD): Es una de las enfermedades neurodegenerati-

vas más destacadas. Se caracteriza por la acumulación y agregación de proteínas mal plegadas que se depositan en forma de agregados e inclusiones de localización intracelular o extracelular y que producen en último término la muerte celular. Los pacientes con AD pueden presentar quejas de memoria y/o un declive en la función cognitiva en comparación a la previa.

- Enfermedad de Parkinson (EP): Descrita por James Parkinson en 1817, es un trastorno neurodegenerativo crónico de curso progresivo y evolución prolongada, caracterizado fundamentalmente por la pérdida de neuronas dopaminérgicas. Es la enfermedad neurodegenerativa más frecuente en personas mayores de 65 años tras la enfermedad de Alzheimer y afecta entre el 1 y el 2 por ciento de la población. Su incidencia y prevalencia aumenta con la edad, y llega a producirse hasta entre el 4 y el 5 por ciento de los mayores.
- Esclerosis Lateral Amiotrófica (ELA): Esta enfermedad se manifiesta raramente antes de la tercera década de vida (solo un 10 por ciento de los casos se producen antes de los cuarenta años), se incrementa con la edad y decae pasada la séptima década. Es más frecuente en hombres que en mujeres, y la supervivencia media es de tres años, pero depende de la precocidad con que afecte a la capacidad respiratoria y, sobre todo, a la calidad de los cuidados.

En el contexto de las enfermedades neurodegenerativas, se han identificado alteraciones proteicas características de cada uno de estas: la enfermedad de Alzheimer se asocia con la producción de un péptido de 42 aminoácidos denominada 'proteína beta amiloide'; la enfermedad de Parkinson con los depósitos de alfa-sinucleína, componente importante de los cuerpos de Lewy; la proteína Tau y tauopatías y la producción de ovillos nenrofibrilares con la demencia frontotemporal; y los desórdenes de las poliglutaminas con la enfermedad de Huntington.

El artículo también discute la importancia de la función de los elementos celulares y cómo su pérdida puede llevar a la neurodegeneración. Esto subraya la importancia de mantener la salud celular para prevenir enfermedades neurodegenerativas.

Concretamente, se menciona la asociación entre el nivel educativo y la reserva cognitiva, y cómo esta puede proteger contra la neurodegeneración. Esto sugiere que la educación y el aprendizaje continuo pueden ser estrategias efectivas para prevenir el deterioro cognitivo relacionado con la edad.

Finalmente, el artículo discute la variabilidad en la progresión de la enfermedad de Alzheimer, con algunos casos progresando rápidamente y otros más lentamente. Esto destaca la necesidad de enfoques de tratamiento personalizados que tengan en cuenta las diferencias individuales en la progresión de la enfermedad.

## 2.3. Cuando el cerebro envejece [9]

El envejecimiento es un proceso complejo y progresivo que implica la pérdida de las funciones de los tejidos y órganos del cuerpo debido al daño acumulado en las células. Aunque no existe un gen que 'programe' el envejecimiento, ciertos genes pueden aumentar la susceptibilidad a enfermedades relacionadas con la vejez. Sin

embargo, el peso de la genética en el ritmo de envejecimiento tan solo es del orden del 20-25 por ciento, mientras que el ambiente y el estilo de vida, que forman parte de la epigenética, tienen un mayor impacto.

El envejecimiento varía completamente entre distintas personas e incluso entre órganos de una misma, por lo que cuantificarlo y medirlo supone un gran desafío. Sin embargo, se han desarrollado métodos para estimar la edad biológica de una persona. El envejecimiento del cerebro es especialmente interesante, ya que puede mostrar tanto un declive como procesos de desarrollo inversos. El estado cognitivo de las personas mayores puede variar significativamente.

El envejecimiento de la población es un tema importante en la sociedad actual y plantea desafíos en diversas disciplinas. Existen varias teorías sobre el envejecimiento, como la del programa genético, los radicales libres, la senescencia celular y la desincronización de los ritmos circadianos. La esperanza de vida varía según el país y el género, pero en líneas generales ha aumentado durante los últimos años.

El objetivo principal de los estudios sobre el envejecimiento es retrasar el inicio de enfermedades crónicas incapacitantes típicas del envejecimiento. En el libro se menciona que la duración de vida ha aumentado en los últimos años y se calcula que las personas nacidas a partir de 1999 tendrán la posibilidad de alcanzar los cien años de edad con más facilidad. Además, se menciona que la longevidad está determinada en parte por el potencial genético y en parte por las condiciones ambientales.

El envejecimiento cerebral es un proceso complejo que afecta a nivel molecular, celular y estructural. Durante el envejecimiento, se producen cambios en la expresión génica, tanto en defecto como en exceso, lo que puede llevar a una disminución de las proteínas sintetizadas, pero también a un aumento de las mismas. Estas modificaciones pueden resultar en una mayor accesibilidad al material genético, lo que puede llevar a una expresión de genes anómala e inestable con el aumento de la edad.

El cerebro es un órgano único y complejo que tiene la capacidad de remodelarse constantemente en respuesta a estímulos tanto internos como externos, un fenómeno conocido como plasticidad cerebral. Este proceso de remodelación implica la formación o pérdida de conexiones entre las células, el fortalecimiento o debilitamiento de las conexiones existentes, y la creación o eliminación de circuitos neuronales.

Por otro lado, el cerebro está compuesto por neuronas: estas son células especializadas que forman parte del sistema nervioso y son responsables de transmitir información a través de impulsos eléctricos e intercambios químicos. Cada neurona está compuesta por un cuerpo celular y dos tipos de prolongaciones: las dendritas y el axón. Las dendritas son ramificaciones que se extienden desde el cuerpo celular y reciben información de otras neuronas, mientras que el axón es una prolongación única que puede recorrer largas distancias y transmite información a otras neuronas o a estructuras diana, como los músculos.

Las neuronas se comunican entre sí a través de un proceso especializado llamado sinapsis. Cada sinapsis está formada por un terminal presináptico, que pertenece

a la neurona que envía la información, y una membrana postsináptica, que pertenece a la neurona que recibe la información. Estos dos componentes están separados por un pequeño espacio llamado hendidura sináptica. Cuando un impulso nervioso llega al terminal presináptico, provoca la liberación de sustancias químicas llamadas neurotransmisores desde vesículas presentes en el terminal. Los neurotransmisores se liberan en la hendidura sináptica y se unen a receptores específicos en la membrana postsináptica, lo que permite la transferencia de la información.

En el caso del envejecimiento, se ha observado en cerebros de ratas una disminución en el número de nuevas neuronas (el proceso de nacimiento de éstas se conoce como neurogénesis). Aunque en humanos la cuestión sigue en debate, experimentos con Carbono-14 en sujetos ancianos han mostrado un modesto declive del recambio anual de las neuronas en el hipocampo en comparación con lo que ocurre en adultos jóvenes, lo que podría ser una causa de degeneración neuronal.

Las nuevas neuronas que se generan de forma continua en el giro dentado del hipocampo, que son cruciales para los mecanismos de memoria, están sujetas a un lento pero continuo recambio, lo que sugiere la renovación de la totalidad de esta población de neuronas entre el nacimiento y los cuarenta años.

En cuanto a la circulación cerebral, sabemos que el cerebro, a pesar de ser un órgano relativamente pequeño, consume una cantidad significativa de energía y oxígeno. Para satisfacer esta alta demanda de oxígeno, el cerebro requiere una cantidad mayor de flujo sanguíneo que otros órganos. Hasta el 15 por ciento de la sangre que el corazón bombea desde el ventrículo izquierdo se dirige hacia el cerebro.

A pesar de ello, los estudios mencionan que la microcirculación cerebral envejece con la edad, lo que resulta en una reducción de su extensión debido a la degeneración de los capilares. Por suerte, la circulación cerebral conserva la función de autorregulación incluso en la vejez avanzada, a través de mecanismos compensatorios que aseguran niveles suficientes de oxigenación para garantizar el desarrollo de las actividades normales.

Otro elemento importante en relación a la circulación cerebral es la barrera hematoencefálica, una estructura distintiva de la red de microcirculación que protege al cerebro de lo que circula por el flujo sanguíneo. Sin embargo, esta barrera también experimenta alteraciones dependientes de la edad, lo que puede afectar la eficiencia del acoplamiento metabólico y 'energético' en el interior de la unidad neurovascular, que es importante para la actividad neural.

Por estas razones y muchas otras, es importante mantener unos buenos hábitos que favorezcan el mejor envejecimiento del cerebro posible; cuando el cerebro envejece bien, se mantiene un buen estado cognitivo a pesar del avance de la edad. Esto puede variar significativamente de una persona a otra, con algunos individuos mayores de sesenta y cinco años llegando a mostrar un estado cognitivo comparable al de los jóvenes.

Factores como la actividad física y cognitiva, una dieta equilibrada y espacios de

socialización pueden contribuir a un envejecimiento cerebral saludable, provocando una estimulación de la neurogénesis, que ayuda creando una especie de reserva neural que compensa las pérdidas funcionales asociadas al envejecimiento. Además, a pesar de la reducción de volumen que puede ocurrir en el cerebro con la edad, esto no necesariamente se traduce en un declive cognitivo. En igualdad de reducción de volumen, algunos ancianos pueden mantener un mejor estado cognitivo.

En contraparte, cuando el cerebro envejece mal pueden darse una serie de consecuencias negativas. Esto podría incluir un declive cognitivo más rápido, la aparición de enfermedades neurodegenerativas como el Alzheimer o el Parkinson y, en definitiva, una disminución en la calidad de vida. El envejecimiento cerebral no saludable puede ser el resultado de una variedad de factores. Por ejemplo, factores genéticos, falta de actividad física y mental, una dieta pobre, estrés crónico y falta de interacción social.

Además, el envejecimiento cerebral no saludable puede estar asociado con cambios en la estructura y función del cerebro. Esto puede incluir la pérdida de neuronas y sinapsis, la degeneración de las vainas de mielina, que ayudan a transmitir señales entre las células cerebrales, y la reducción del flujo sanguíneo al cerebro.

Por último, se habla sobre los ritmos circadianos y el sueño, y el gran impacto que tienen sobre la salud y el envejecimiento del cerebro. Los ritmos circadianos son oscilaciones naturales y regulares en los procesos biológicos que ocurren en un ciclo de aproximadamente 24 horas. Estos ritmos son generados por un grupo de neuronas en la base del cerebro llamado núcleo supraquiasmático (NSQ) y se adaptan a la luz ambiental. Por otro lado, el sueño es una función biológica esencial que permite al cerebro descansar y recuperarse. Durante el sueño, el cerebro realiza una serie de funciones importantes, como la consolidación de la memoria y la eliminación de los desechos metabólicos.

Las alteraciones en los ritmos circadianos y el sueño pueden tener efectos negativos en la salud del cerebro y pueden acelerar el proceso de envejecimiento. Por ejemplo, la falta de sueño o la mala calidad de éste se han asociado con un mayor riesgo de enfermedades neurodegenerativas, como la enfermedad de Alzheimer. Además, las alteraciones en los ritmos circadianos pueden afectar la función cognitiva y el estado de ánimo, y se han asociado con una variedad de trastornos mentales y neurológicos. Por lo tanto, mantener ritmos circadianos saludables y una buena higiene del sueño puede ser importante para promover un envejecimiento cerebral saludable.

## 2.4. Toward a biological definition of Alzheimer's disease [24]

Este artículo discute el marco de investigación desarrollado por el Instituto Nacional sobre el Envejecimiento y la Asociación de Alzheimer (NIA-AA) para la enfermedad de Alzheimer (AD). Este marco define la AD basándose en biomarcadores que pueden ser detectados a través de exámenes post mortem o pruebas in vivo, estos biomarcadores se dividen en aquellos relacionados con la deposición de beta amiloide, los que se basan en la deposición de tau patológico y los relacionados con

la neurodegeneración [AT(N)].

### 2.4.1. Motivación y antecedentes

En 2011, el Instituto Nacional sobre el Envejecimiento y la Asociación de Alzheimer (NIA-AA) crearon guías diagnósticas separadas para las etapas clínicas de AD. El progreso científico desde entonces ha llevado a una actualización para unificar estas guías en un 'marco de investigación' que se enfoca en el uso de biomarcadores para definir la AD en personas vivas. Este marco busca proporcionar un lenguaje común para investigadores, facilitando la generación y prueba de hipótesis sobre las interacciones entre los procesos patológicos y los síntomas cognitivos.

Durante muchos años, la AD se concebía como una construcción clínico-patológica, lo que suponía que se asumiese que si un individuo presentaba síntomas típicos de la AD, tendría cambios neuropatológicos de AD en la autopsia y si, por el contrario, no presentaba síntomas, no presentaría signos de AD en la autopsia. Esta perspectiva se demostró incorrecta, o no suficientemente precisa, cuando estos diagnósticos clínicos de AD presentaban un 50 por ciento de tasa de error en los ancianos. Por esta razón, el NIA-AA propone definir la AD por biomarcadores en lugar de síntomas, que pueden ser independientes los unos de los otros. Además, introducen un sistema de etapas para medir el avance y la gravedad de esta condición basado en biomarcadores y el deterioro cognitivo del paciente.

### 2.4.2. Sistema de clasificación AT(N)

El sistema de clasificación AT(N) agrupa los biomarcadores en función del proceso patológico que cada uno mide. Los biomarcadores de beta amiloide (A), tau patológico (T) y neurodegeneración (N) se consideran de manera flexible, permitiendo la adición de nuevos biomarcadores en el futuro.

| **Grupo de biomarcadores** | **Biomarcadores incluidos** |
|---|---|
| A: Depósito de beta amiloide | CSF Ab42, PET amiloide |
| T: Tau patológico | CSF tau fosforilado, PET tau |
| (N): Neurodegeneración | MRI, FDG PET, CSF tau total |

Tabla 2.1: Agrupación de biomarcadores AT(N)

### 2.4.3. Perfiles de biomarcadores

Este sistema de etapas se basa en diferentes estados cognitivos por los que puede pasar el paciente hasta llegar al estado de demencia, que es el que más se asocia tradicionalmente con la enfermedad de Alzheimer. Para poder determinar la etapa en la que se encuentra un determinado individuo, se proponen los perfiles de biomarcadores; el resultado de combinar de forma binaria la presencia o ausencia de los tres biomarcadores presentados anteriormente. Al tratarse de tres biomarcadores con dos posibles estados para cada uno, nos encontramos con ocho posibles perfiles de biomarcadores representados de la siguiente manera: A[+/-]T[+/-](N)[+/-], donde el signo situado a la derecha de cada biomarcador supone la presencia (+) o ausencia (-) de este en el paciente.

Tras estudiar el estado de pacientes con cada uno de los posibles perfiles de biomarcadores, se han podido distinguir las distintas categorías en las que puede encontrarse cada individuo y el perfil al que está asociada cada una. Se distinguen, principalmente, cinco casos:

- Estado normal de los biomarcadores: Se encuentra en esta categoría cualquier persona con el perfil A-T-(N)-, no presenta ninguna afección cognitiva.
- Cambio patológico de Alzheimer: Se encuentran en este estado todos los pacientes con el perfil A+T-(N)-, se trata de una etapa muy temprana de la AD.
- Alzheimer posiblemente acompañado de un cambio patológico no asociado con Alzheimer: Se encuentran en este estado todos los pacientes con el perfil A+T-(N)+.
- Enfermedad de Alzheimer: Se encuentran en esta categoría todos los individuos con los perfiles A+T+(N)- y A+T+(N)+.
- Cambio patológico no asociado con Alzheimer: Se encuentran en esta categoría todos los pacientes con cualquier perfil que no incluya presencia de beta amiloide (A+). Los pacientes en esta categoría, además de la enfermedad de Alzheimer, pueden desarrollar otras enfermedades neurodegenerativas al dar positivo en otros biomarcadores.

| **Perfil AT(N)** | **Categoría de biomarcadores** |
|---|---|
| A-T-(N)- | Biomarcadores normales de AD |
| A+T-(N)- | Cambio patológico de Alzheimer |
| A+T+(N)- | Enfermedad de Alzheimer |
| A+T+(N)+ | Enfermedad de Alzheimer |
| A+T-(N)+ | Alzheimer con cambio patológico no asociado |
| A-T+(N)- | Cambio patológico no asociado con Alzheimer |
| A-T-(N)+ | Cambio patológico no asociado con Alzheimer |
| A-T+(N)+ | Cambio patológico no asociado con Alzheimer |

Tabla 2.2: Perfiles de biomarcadores y categorías

### 2.4.4. Estados cognitivos y aplicaciones del sistema

Por otro lado, también es esencial distinguir entre los distintos estados cognitivos que puede mostrar un paciente. Al combinar las categorías asociadas a los perfiles de biomarcadores con estos posibles estados cognitivos, obtenemos una clasificación final mucho más realista e interesante para el ámbito de la investigación que la que había tradicionalmente. A grandes rasgos, se distinguen los estados cognitivos: cognitivamente intacto, deterioro cognitivo leve (MCI) y demencia. En la actualidad, se evita la noción de estos estados como 'casos separados' y se habla del 'Continuum Cognitivo', que hace alusión al avance tradicional de la enfermedad de Alzheimer. Consideramos que pertenecen a este grupo todos los pacientes con cualquiera de los perfiles de biomarcadores que indican una deposición de beta amiloide (A+).

Además de esto, se describe un sistema de 'estadio clínico numérico' que se aplica solo a individuos dentro del 'Continuum Cognitivo' de Alzheimer. Este sistema

de etapas refleja la evolución secuencial de la enfermedad de Alzheimer desde una etapa inicial caracterizada por la aparición de biomarcadores anormales de AD en pacientes asintomáticos. A medida que las anomalías de los biomarcadores progresan, los primeros síntomas sutiles se vuelven detectables. La progresión adicional de las anomalías de los biomarcadores va acompañada de un empeoramiento progresivo de los síntomas cognitivos, culminando en demencia.

Es importante mencionar que el comité no proporcionó recomendaciones específicas para los detalles de implementación, sino que esbozó un marco de investigación general que puede ser adaptado por grupos de investigación individuales. Esto fue algo prioritario para ellos para facilitar otras investigaciones paralelas y permitir que puedan tomarse la mayor cantidad de libertades posible, por ejemplo utilizando una nomenclatura alternativa para los perfiles de biomarcadores. El uso de la genética en el marco de investigación no fue incluido porque las variantes genéticas indican el riesgo de un individuo de desarrollar cambios patológicos, en lugar de medir el cambio patológico en sí mismo.

Para finalizar, el artículo sugiere que, en el futuro, las pruebas de biomarcadores basadas en sangre, junto con la información genética, clínica y demográfica, probablemente desempeñarán un papel importante en la selección de individuos para pruebas de biomarcadores más invasivas y costosas. Además, se discute la idea de que ciertos biomarcadores de imagen y de líquido cefalorraquídeo son proxies válidos para los cambios neuropatológicos de la AD.

## 2.5. Biomarcadores

Un biomarcador se define como una sustancia utilizada para indicar un estado biológico, patogénico o de respuesta a un tratamiento farmacológico. En el caso de la enfermedad de Alzheimer (AD), se realizan múltiples pruebas o medidas para la evaluación clínica de la demencia. A continuación, se explican varios de estos biomarcadores, incluyendo ADAS-13, CDR-SB, RAVLT-Learning, TRABSCOR, PACC, así como los marcadores de CSF (Líquido cefalorraquídeo) A$\beta$, pTau, tTau y EveryCognition.

### 2.5.1. ADAS-13 [12]

El nombre de este marcador (Alzheimer's Disease Assessment Scale - Cognitive Subscale 13) hace referencia a las distintas pruebas que se le realizan al paciente para comprobar su rendimiento cognitivo y su función mental en el ámbito del AD. En este caso concreto se realizan 13 de ellas. Estas tareas evalúan habilidades como la memoria, la atención, el lenguaje y la capacidad para realizar algunos trabajos específicos, y su resultado final será la suma de la puntuación obtenida en cada una de ellas, indicando un mayor deterioro cognitivo cuanto más alto sea el resultado. Aunque las interpretaciones de los resultados dependen de factores como la población estudiada o la gravedad de los síntomas cognitivos, generalmente se interpretan de la siguiente forma: entre 0 y 4 puntos suponen ausencia de o muy poco deterioro cognitivo; entre 5 y 9 puntos, deterioro cognitivo leve; entre 10 y 19 puntos, deterioro cognitivo moderado y, por encima de 20 puntos, deterioro cognitivo grave

o demencia. Cabe destacar que, normalmente, se realizan varias iteraciones de la prueba y los resultados se evalúan sobre la media de todas ellas.

### 2.5.2. CDR-SB [13]

El marcador (Clinical Dementia Rating Scale Sum of Boxes) se emplea para medir la gravedad de la demencia. De manera similar al ADAS-13, el CDR-SB también evalúa el desempeño del paciente en tareas varias como orientación, memoria, juicio y resolución de problemas, función en la comunidad, actividades domésticas o cuidado personal. Normalmente, el desempeño en estas áreas se evalúa mediante entrevistas con el paciente, asignándose una nota numérica de 0 a 3 y, o bien se hace la media de los 6 resultados, o se muestra directamente con valores de 0 a 18. Cuanto mayor sea el resultado, mayor gravedad de la demencia supone y los valores se desglosan de esta forma: entre 0 y 0.5 hay ausencia de o muy leve grado de demencia, entre 0.5 y 1.5 hay declive cognitivo y por encima de 1.5 se considera que el paciente tiene demencia.

### 2.5.3. RAVLT-Learning [23]

Este marcador se especializa en el contexto de la memoria verbal y auditiva y la capacidad de aprendizaje (Rey Auditory-Verbal Learning Test). Durante esta prueba, a los pacientes se les presenta una lista de palabras que deben memorizar para que, posteriormente, las repitan durante varias rondas. Para este test se evalúa la capacidad de aprender a lo largo de las fases y se establece el resultado en la cantidad de palabras que se hayan conseguido recordar, poniendo a prueba especialmente la capacidad de aprendizaje verbal y la memoria a corto plazo. La cantidad de palabras que el paciente debe recordar varía según lo que decida el realizador de la prueba, pero normalmente se establecen 15 palabras y 5 repeticiones, por lo que el resultado numérico variará entre 0 y 75, suponiendo un peor desempeño cuanto menor sea el resultado y por tanto un mayor deterioro cognitivo, al contrario del resto de marcadores presentados anteriormente.

### 2.5.4. TRABSCORE o TRABSCOR [25]

Este test es una variación del 'Trail Making Test', donde el objetivo es que el paciente conecte 25 puntos por orden lo más rápido posible, manteniendo un mínimo de precisión. La prueba se utiliza para evaluar el desempeño en áreas como la velocidad y flexibilidad mental, la velocidad de búsqueda o la exploración visual. Consiste en dos fases: en la primera, al paciente se le solicita que conecte los 25 puntos, que vienen todos marcados con su respectivo número, mientras que en la segunda se marcan los 13 primeros con un número y los 12 siguientes con una letra de la 'A' a la 'L' y el paciente debe conectarlos alternando entre números y letras (1, A, 2, B, ...). El resultado de esta prueba se obtiene del tiempo que tarde el paciente en completarla, por lo que, cuanto mayor sea el resultado, mayor grado de demencia supone. El tiempo medio que se tarda en completar la tarea es de 29 segundos para la primera fase y de 75 para la segunda. Si el paciente alcanza los 300 segundos sin haberla completado, normalmente se da la prueba por finalizada.

### 2.5.5. PACC [17]

El biomarcador conocido como 'Preclinical Alzheimer Cognitive Composite' se caracteriza principalmente por ser utilizado en la investigación para evaluar el rendimiento cognitivo en etapas preclínicas de la enfermedad de Alzheimer. El PACC se obtiene a partir de varias pruebas cognitivas y tiene el objetivo de detectar cambios sutiles en la función cognitiva antes de que aparezcan síntomas clínicos. La comprensión de las etapas tempranas del AD sigue siendo un área activa de estudio en campos como la medicina o la neurociencia, por lo que las pruebas realizadas pueden variar, pero generalmente incluyen evaluaciones de la memoria, la atención, la velocidad de procesamiento o la función ejecutiva. El resultado de este biomarcador se obtiene sumando las puntuaciones de estas pruebas, por lo que, al igual que en el RAVLT, una menor puntuación supone un peor resultado y por tanto un mayor deterioro cognitivo, lo que en este caso se asocia con una mayor tendencia o probabilidad de desarrollar AD en el futuro.

### 2.5.6. EveryCognition (ECog) [19]

El marcador EveryCognition (ECog) se utiliza para evaluar la capacidad funcional diaria de individuos mayores, y es especialmente útil para predecir la conversión de un estado de cognición normal a deterioro cognitivo leve (MCI) y, potencialmente, a demencia. El ECog mide las habilidades cognitivas diarias en seis dominios relevantes: memoria, lenguaje, habilidades visuoespaciales, planificación, organización y atención dividida. Tanto el propio paciente como un informante pueden completar el cuestionario, proporcionando una evaluación integral del estado cognitivo del individuo. Un incremento de una unidad en el puntaje total del ECog se asocia con el doble de riesgo de desarrollar MCI en un período de seguimiento promedio de tres años. Los dominios de memoria y funciones ejecutivas diarias son los predictores más fuertes de la conversión a MCI.

### 2.5.7. A$\beta$1-42 [22]

El beta-amiloide 1-42 (A$\beta$1-42) es un péptido derivado del metabolismo de la proteína precursora amiloidea (APP). La reducción de los niveles de A$\beta$1-42 en el CSF es un indicador temprano de la acumulación de placas amiloides en el cerebro, una característica distintiva del Alzheimer. Los estudios han establecido que niveles de A$\beta$1-42 inferiores a 1100 pg/mL están altamente correlacionados con una carga amiloide positiva observada mediante PET. Este marcador tiene una alta sensibilidad y especificidad para la detección de la enfermedad en sus etapas tempranas.

### 2.5.8. pTau [22]

La tau fosforilada (pTau) refleja la alteración en el procesamiento de la proteína tau, lo cual conduce a la formación de ovillos neurofibrilares, otra característica patológica del Alzheimer. Los niveles elevados de pTau en el CSF son indicativos de daño neuronal y degeneración. Se ha determinado que una relación pTau/A$\beta$1-42 mayor a 0.022 es un indicador confiable de la patología tau en el cerebro. Este ratio ofrece una mejor precisión diagnóstica en comparación con los niveles individuales de A$\beta$1-42 o pTau.

### 2.5.9. tTau [22]

La tau total (tTau) mide la cantidad total de proteína tau presente en el CSF, que aumenta significativamente en respuesta a la neurodegeneración. La elevación de tTau está asociada con el daño axonal y la pérdida neuronal. Un valor de tTau/A$\beta$1-42 mayor a 0.26 se asocia con una alta probabilidad de Alzheimer. Este marcador, al igual que pTau, proporciona información crítica sobre el estado neurodegenerativo del paciente.

## 2.6. Robust parametric modeling of Alzheimer's disease progression [20]

El artículo propone un método robusto de modelado paramétrico para la progresión de la enfermedad de Alzheimer utilizando datos longitudinales. El método relaciona linealmente la edad de un paciente con un medidor de la progresión de una enfermedad (DPS) y ajusta funciones logísticas generalizadas como funciones de dicho DPS, mediante el método de máxima verosimilitud. La robustez de las estimaciones se cuantifica mediante 'bootstrapping', y los puntos de inflexión estimados de las funciones ajustadas se utilizan para ordenar temporalmente los biomarcadores modelados en el curso de la enfermedad. El modelo propuesto logra un alto rendimiento en la predicción de valores de biomarcadores y la clasificación del estado clínico, superando los resultados del estado del arte.

El estudio también discute un enfoque de optimización multiobjetivo para ajustar curvas de biomarcadores y estimar parámetros de progresión de la enfermedad específicos del sujeto. El enfoque implica el uso de funciones logísticas (curvas en forma de S) para modelar las trayectorias de los biomarcadores y optimizar los parámetros utilizando un enfoque alternativo. El artículo también analiza la predicción de valores de biomarcadores, la clasificación del estado clínico y el diseño experimental utilizando los valores de las bases de datos 'ADNI' y 'NACC'. Estos datos se filtran para eliminar valores atípicos y garantizar la precisión, todo esto con el objetivo de proporcionar un método robusto y eficiente para analizar resultados de biomarcadores en el contexto de la enfermedad de Alzheimer (AD).

### 2.6.1. Modelo matemático

A continuación, se explica en detalle en qué consiste el método utilizado para modelar las dinámicas de los marcadores. En primer lugar, se estiman dos conjuntos de parámetros: parámetros observables específicos de biomarcadores, que se asignan para ajustar las curvas de biomarcadores y parámetros ocultos específicos del paciente sobre la progresión de su enfermedad, que se usan para ajustar las trayectorias. Es importante destacar que estos últimos son dependientes de la edad del sujeto. Una vez tenemos esta información, se define el resultado de los biomarcadores de la siguiente forma:

$$y_{i,j,k} = f(s_{i,j}; \theta_k) + \sigma_k \epsilon_{i,j,k}, \tag{2.1}$$

donde $\sigma_k$ es la desviación típica del biomarcador k, cuyos parámetros son $\theta_k$, $\epsilon_{i,j,k}$ es el ruido blanco gaussiano añadido y $s_{i,j}$ es el DPS del sujeto i en su visita número j

que, además, se define de la siguiente forma:

$$s_{i,j} = \alpha_i t_{i,j} + \beta_i, \tag{2.2}$$

donde $t_{i,j}$ es la edad del paciente i durante su visita j, mientras que $\alpha_i$ y $\beta_i$ son la tasa y el comienzo de la enfermedad de dicho paciente, respectivamente.

Finalmente, se obtiene la función de optimización multiobjetivo mencionada previamente:

$$\{\hat{\alpha}, \hat{\beta}, \hat{\theta}\} = \min_{i,j,k} \sum_{i,j,k} \omega_i \rho(\frac{y_{i,j,k} - f(\alpha_i t_{i,j} + \beta_i; \theta_k)}{\sigma_k}), \tag{2.3}$$

donde $p(\cdot)$ es una función de máxima verosimilitud y $\omega_i = 1/N_i$ es un factor utilizado para normalizar la función con el número de resultados (puntos de la función) disponibles para el sujeto estudiado.

Por otro lado, el estudio utilizó datos de 'ADNI' y 'NACC' para modelar la dinámica de biomarcadores utilizados para 'AD'. El método propuesto encontró el mayor rendimiento utilizando una combinación de la función modificada de Stannard para el ajuste de biomarcadores y un análisis de regresión logística. Además, el orden temporal de los biomarcadores mostró que los biomarcadores de LCR y PET, así como RAVLT-IR, preceden a todos los demás biomarcadores en la progresión de la enfermedad de Alzheimer. La distribución de los puntos de inflexión de los biomarcadores puede utilizarse para comprender cómo avanzan sus valores durante el curso de la enfermedad de Alzheimer. También se presenta un análisis sobre la distribución de los 'DPS' y la varianza de los puntos de inflexión estimados por distintos biomarcadores.

Seguidamente, el estudio presenta un modelo para predecir valores de biomarcadores y clasificar el estado clínico en la enfermedad de Alzheimer. Este modelo supera a los métodos anteriores en términos de precisión y muestra resultados prometedores en comparación con los resultados del estado del arte. El estudio también discute el uso potencial de técnicas de aprendizaje de conjunto para mejorar el rendimiento de predicción.

### 2.6.2. Validación del modelo y conclusiones

Para validar el modelo propuesto, los autores utilizaron datos longitudinales de las bases de datos 'ADNI' y 'NACC'. Estos datos se sometieron a un proceso de filtrado riguroso para eliminar valores atípicos y asegurar la precisión de las estimaciones. El modelo mostró un rendimiento superior en la predicción de valores de biomarcadores y en la clasificación del estado clínico en comparación con los métodos existentes. Los resultados del estudio sugieren que el modelo propuesto no solo es robusto y preciso, sino que también es aplicable a diferentes tipos de datos longitudinales y biomarcadores.

En conclusión, el artículo propone un modelo paramétrico robusto para la progresión de la enfermedad de Alzheimer (AD) basado en el método de máxima verosimilitud utilizando el análisis de regresión logística y dicho modelo resulta superar a los métodos anteriores en cuanto al rendimiento de las predicciones. Además, el enfoque

propuesto puede aplicarse a diferentes datos de series temporales y, principalmente, a biomarcadores con diversas características.

## 2.7. Learning Multimodal Digital Models of Disease Progression [28]

Esta tesis aborda el modelado de la progresión de enfermedades neurodegenerativas utilizando herramientas matemáticas avanzadas y técnicas de análisis de datos longitudinales. A lo largo del texto, se presentan los fundamentos teóricos de la geometría Riemanniana y los modelos de efectos mixtos, se desarrollan algoritmos de estimación y simulación, y se describen herramientas de software implementadas para estos fines. Para este proyecto, el estudio se ha centrado exclusivamente en los capítulos 1, 2 y 7, que cubren los conceptos matemáticos básicos, los métodos de estimación del modelo, y las herramientas de software desarrolladas.

### 2.7.1. Fundamentos teóricos

El primer capítulo proporciona una introducción a los conceptos matemáticos fundamentales sobre los cuales se basa el modelo presentado, específicamente la geometría Riemanniana y los modelos de efectos mixtos. Estos conceptos son esenciales para comprender los capítulos posteriores, aunque no se abordan de manera exhaustiva en esta sección. La geometría Riemanniana se centra en la estructura y propiedades de las variedades, que son espacios topológicos que localmente se asemejan al espacio euclidiano. Una métrica en una variedad permite definir distancias y ángulos, dotando a la variedad de una estructura geométrica. Las geodésicas, que son las curvas que representan el camino más corto entre dos puntos en una variedad, y el mapa exponencial, que asocia a cada punto un vector tangente, son conceptos clave en esta área. Además, el transporte paralelo permite trasladar vectores a lo largo de curvas en la variedad, manteniendo su direccionalidad relativa.

En cuanto a los modelos de efectos mixtos, estos combinan efectos fijos, que son comunes a todos los individuos, y efectos aleatorios, que representan variaciones individuales. Los modelos lineales de efectos mixtos permiten modelar relaciones simples, mientras que los modelos no lineales de efectos mixtos pueden capturar relaciones más complejas entre variables. Los datos longitudinales, que son observaciones repetidas de los mismos individuos a lo largo del tiempo, son especialmente útiles para estudiar fenómenos biológicos. El modelo de progresión de la enfermedad descrito en el documento utiliza la geometría Riemanniana para modelar trayectorias promedio e individuales de progresión de la enfermedad. Este modelo se descompone en componentes de efectos fijos y aleatorios, y las condiciones de identificabilidad establecen restricciones necesarias para asegurar que los parámetros del modelo se puedan estimar de manera única y precisa.

### 2.7.2. Algoritmos de aprendizaje estadístico

El segundo capítulo se centra en los algoritmos de aprendizaje estadístico utilizados para la estimación del modelo de progresión de la enfermedad. El algoritmo E-M, por ejemplo, es utilizado para encontrar estimaciones de parámetros en modelos de

efectos mixtos. El SAEM (Stochastic Approximation Expectation Maximization) es una extensión del algoritmo E-M que incorpora aproximaciones estocásticas para manejar mejor la variabilidad y complejidad de los datos. El MCMC-SAEM (Monte Carlo Markov Chain SAEM) combina técnicas de cadenas de Markov y el algoritmo SAEM para mejorar la estimación en modelos complejos, y el muestreo de Gibbs facilita la generación de distribuciones de probabilidad para parámetros individuales. Los procedimientos de estimación incluyen la calibración, que ajusta el modelo para que coincida con los datos observados; la personalización, que estima los efectos aleatorios individuales a partir de los datos; y la reconstrucción e imputación de valores faltantes, que utiliza el modelo para predecir datos faltantes y futuros puntos temporales. La simulación se utiliza para generar datos simulados y evaluar la precisión y robustez del modelo.

### 2.7.3. Herramientas de software

El séptimo capítulo describe las herramientas de software desarrolladas para implementar el modelo de progresión de la enfermedad. Leasp es un paquete de software en C++ que permite el análisis de datos longitudinales estructurados espacialmente, y se utiliza para estimar la progresión de la enfermedad a partir de datos de imágenes médicas. Leaspy, por otro lado, es una caja de herramientas en Python que proporciona un marco general para el análisis de datos longitudinales, abarcando desde la estimación de la progresión a largo plazo de la enfermedad hasta la imputación de valores faltantes y la simulación de cohortes virtuales. Estas herramientas de software ofrecen una base sólida para futuros estudios y aplicaciones en el campo de la modelización de enfermedades neurodegenerativas, como el Alzheimer.

### 2.7.4. Conclusiones

En conclusión, estos capítulos ofrecen una visión técnica y detallada de los fundamentos matemáticos y estadísticos utilizados para modelar la progresión de enfermedades, destacando las herramientas y métodos desarrollados para la estimación y simulación de datos longitudinales. Las herramientas de software presentadas en el documento proporcionan un soporte robusto para la investigación continua y las aplicaciones prácticas en el campo de la modelización de la progresión de enfermedades.

## 2.8. Desarrollo de una aplicación multiplataforma con GUI en Python

El desarrollo de aplicaciones multiplataforma es un aspecto esencial en la ingeniería de software moderno, especialmente debido a la diversidad de dispositivos y sistemas operativos utilizados por los usuarios [37]. Una aplicación multiplataforma permite que el software funcione en múltiples sistemas operativos, como Windows, macOS y Linux, sin necesidad de modificar el código base de manera significativa. En este contexto, el lenguaje de programación Python, combinado con la biblioteca Tkinter, ofrece una solución robusta y eficiente para el desarrollo de interfaces gráficas de usuario (GUI) multiplataforma.

### 2.8.1. ¿Qué significa que una aplicación sea multiplataforma?

Una aplicación multiplataforma es aquella que puede ejecutarse en diversos sistemas operativos sin necesidad de modificar el código fuente de manera significativa. Las principales características que definen una aplicación multiplataforma incluyen:

- **Compatibilidad de sistema operativo**: La aplicación debe funcionar correctamente en múltiples sistemas operativos, como Windows, macOS y Linux.
- **Interfaz de usuario consistente**: La interfaz gráfica debe ser coherente y funcional en todos los sistemas soportados.
- **Gestión de dependencias**: La aplicación debe manejar sus dependencias de manera que estas estén disponibles en todas las plataformas de destino.
- **Mantenimiento y actualización simplificados**: El código base debe ser único, facilitando el mantenimiento y las actualizaciones sin necesidad de gestionar múltiples versiones para diferentes plataformas.

### 2.8.2. Consideraciones para el desarrollo multiplataforma

Para lograr una aplicación multiplataforma efectiva, se deben tener en cuenta varios aspectos durante el desarrollo:

1. **Abstracción de plataforma**: Utilizar bibliotecas que abstraigan las diferencias entre sistemas operativos, permitiendo escribir código que funcione en todos ellos.
2. **Gestión de dependencias**: Asegurarse de que todas las dependencias de la aplicación estén disponibles y funcionen correctamente en cada plataforma.
3. **Pruebas en múltiples sistemas operativos**: Realizar pruebas exhaustivas en todos los sistemas operativos soportados para garantizar la funcionalidad y la experiencia de usuario consistentes.
4. **Distribución y empaquetado**: Utilizar herramientas que faciliten la distribución y el empaquetado de la aplicación para diferentes sistemas operativos.

#### Distribución de la aplicación

Para distribuir la aplicación en diferentes sistemas operativos, se pueden utilizar herramientas como `PyInstaller` [34] para empaquetar la aplicación en un solo ejecutable. Esto garantiza que todas las dependencias estén incluidas y que la aplicación sea fácil de instalar y ejecutar en cualquier plataforma.

### 2.8.3. Conclusiones

El desarrollo de aplicaciones GUI multiplataforma en Python es una tarea alcanzable gracias a bibliotecas como Tkinter. Estas herramientas proporcionan los medios necesarios para crear aplicaciones que funcionan en múltiples sistemas operativos, permitiendo a los desarrolladores alcanzar una audiencia más amplia sin duplicar esfuerzos. La clave del éxito radica en abstraer las diferencias entre plataformas, gestionar las dependencias de manera efectiva y realizar pruebas exhaustivas en todos los sistemas operativos soportados.

## 2.9. Contribuciones de la literatura al proyecto

En esta sección se presentan las principales contribuciones de cada investigación al desarrollo del proyecto.

### 2.9.1. Neurodegeneración [21]

Este libro explora el complejo proceso de envejecimiento cerebral y su relación con enfermedades neurodegenerativas como el Alzheimer, el Parkinson y la esclerosis lateral amiotrófica. Discute la importancia de la longevidad, el papel del ADN y la epigenética, y cómo factores como el estrés oxidativo y la neuroinflamación contribuyen a la neurodegeneración. Además, se describen las bases celulares y moleculares del envejecimiento, incluyendo los tipos de muerte celular (apoptosis, necrosis y autofagia) y los cambios asociados con el envejecimiento cerebral.

Esta sección de la literatura ha aportado una base teórica fundamental para facilitar el entendimiento del gran problema al que se enfrenta este proyecto en última instancia.

### 2.9.2. Cuando el cerebro envejece [9]

El envejecimiento cerebral es un proceso complejo influenciado por factores genéticos, ambientales y de estilo de vida. Esta sección aborda cómo el envejecimiento varía entre individuos y órganos, destacando la plasticidad cerebral y la importancia de la neurogénesis en el hipocampo. También se examinan la microcirculación cerebral, la barrera hematoencefálica y los ritmos circadianos, y cómo su alteración puede afectar la salud cerebral. Se enfatiza la importancia de mantener hábitos saludables para un envejecimiento cerebral óptimo.

De forma similar al libro anterior, 'Cuando el cerebro envejece' ha aportado un contexto de suma importancia para entender y asentar los objetivos del proyecto en el ámbito de la neurología.

### 2.9.3. Toward a biological definition of Alzheimer's disease [24]

Este estudio presenta el marco de investigación del Instituto Nacional sobre el Envejecimiento y la Asociación de Alzheimer (NIA-AA) para definir la enfermedad de Alzheimer mediante biomarcadores en lugar de síntomas clínicos. Se describen los biomarcadores relacionados con la beta-amiloide, tau patológico y neurodegeneración, y se introduce un sistema de etapas basado en biomarcadores y deterioro cognitivo. También se discuten las implicaciones de este enfoque para la investigación y el tratamiento personalizados.

El artículo amplía la base aportada por los dos libros anteriores. Aporta un análisis más profundo y cubre completamente toda la teoría necesaria para entender el marco neurológico del trabajo.

### 2.9.4. Biomarcadores

Los biomarcadores son cruciales para la evaluación clínica de la demencia y la enfermedad de Alzheimer. Esta sección detalla varios biomarcadores, como ADAS-13, CDR-SB, RAVLT-Learning, TRABSCOR y PACC, junto con marcadores de CSF como A$\beta$1-42, pTau y tTau. Cada biomarcador tiene su propio método de evaluación y relevancia en la detección y monitoreo de la progresión de la enfermedad, ofreciendo herramientas esenciales para el diagnóstico y la investigación.

Aunque no ha sido necesario profundizar en gran medida en el fundamento de cada biomarcador para este proyecto, el uso de estos marcadores ha resultado indispensable para la elaboración del proyecto.

### 2.9.5. Robust parametric modeling of Alzheimer's disease progression [20]

El artículo describe un método robusto de modelado paramétrico para la progresión de la enfermedad de Alzheimer utilizando datos longitudinales. Este modelo relaciona la edad con un medidor de progresión de la enfermedad y ajusta funciones logísticas mediante máxima verosimilitud. Se utilizan técnicas de optimización y 'bootstrapping' para mejorar la precisión. Los resultados muestran un alto rendimiento en la predicción de biomarcadores y clasificación del estado clínico, destacando la eficacia del modelo propuesto.

Este artículo contiene toda la base teórica y el desarrollo del algoritmo RPDPM. Este algoritmo se ha utilizado como motor principal del proyecto para obtener las predicciones de los sujetos, por lo que su aportación al trabajo ha sido inmensa.

### 2.9.6. Learning Multimodal Digital Models of Disease Progression [28]

La tesis aborda el modelado de la progresión de enfermedades neurodegenerativas utilizando geometría Riemanniana y modelos de efectos mixtos. Se desarrollan algoritmos de estimación y simulación, y herramientas de software como Leasp y Leaspy para el análisis de datos longitudinales. Estos modelos permiten estimar trayectorias de progresión de la enfermedad y personalizar los tratamientos basados en datos de imágenes médicas y otros biomarcadores.

De forma similar al artículo anterior, esta tesis presenta la teoría necesaria para desarrollar un algoritmo predictivo mediante las herramientas provistas por la biblioteca Leaspy. Este algoritmo ha sido utilizado como alternativa del RPDPM para una de las variantes de la aplicación principal.

### 2.9.7. Desarrollo de una aplicación multiplataforma con GUI en Python

Esta sección explica la importancia del desarrollo de aplicaciones multiplataforma utilizando Python y la biblioteca Tkinter. Se destacan las características esenciales de una aplicación multiplataforma, como la compatibilidad con diferentes sistemas operativos y la gestión de dependencias. Se discuten las consideraciones clave para el desarrollo multiplataforma, incluyendo la abstracción de plataforma, pruebas

exhaustivas y herramientas de distribución como PyInstaller, que facilitan el empaquetado y distribución de la aplicación.

La investigación presentada en esta sección se realizó durante una fase más avanzada del trabajo para aportar un contexto adicional necesario para la creación de las variantes del programa principal.

# Capítulo 3

# Materiales

## 3.1. Introducción

En este capítulo se muestran todas las herramientas que han sido utilizadas a lo largo del desarrollo del proyecto. Para cada una, se incluye una breve explicación y algunas aplicaciones por las que han sido interesantes para el desarrollo del proyecto.

## 3.2. MATLAB [46]

MATLAB es un entorno de computación numérica y programación que permite realizar análisis y cálculos matemáticos de manera eficiente. Se destaca por su amplia gama de funciones integradas y su capacidad para trabajar con matrices y vectores de forma sencilla. MATLAB se utiliza en una variedad de campos, como ingeniería, ciencias naturales, economía y más, para realizar tareas como modelado, simulación, visualización de datos y desarrollo de algoritmos. Ofrece un entorno interactivo que permite escribir y ejecutar código de manera rápida, así como herramientas para la creación de interfaces gráficas y la integración con otros lenguajes de programación.

Para este trabajo, sus funciones para el manejo de matrices, sus herramientas para el modelado y su integración con otros lenguajes de programación son las características que más útiles han resultado.

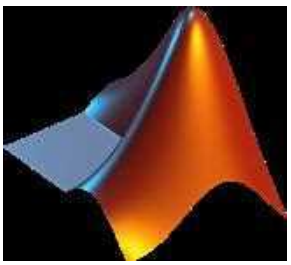

Figura 3.1: Logo de MATLAB.

## 3.3. MATLAB APP Designer [31]

MATLAB App Designer es una herramienta integrada en MATLAB que permite a los usuarios diseñar y desarrollar aplicaciones interactivas con interfaces gráficas de usuario (GUI) de manera sencilla y eficiente. Proporciona un entorno de diseño visual donde los componentes de la interfaz, como botones, menús, gráficos y controles de entrada, pueden ser arrastrados y colocados en la ventana de la aplicación. Esto facilita la creación de aplicaciones complejas sin necesidad de un conocimiento profundo de la programación de interfaces gráficas.

App Designer también incluye un editor de código que permite a los usuarios escribir scripts y funciones MATLAB directamente vinculados a los componentes de la GUI. Además, proporciona herramientas para la depuración y el ajuste fino de la aplicación, asegurando que el rendimiento sea óptimo. Esta combinación de diseño visual y edición de código hace que MATLAB App Designer sea una herramienta poderosa para ingenieros, científicos y profesionales que necesitan desarrollar aplicaciones personalizadas para análisis de datos, simulaciones y otras tareas computacionales avanzadas.

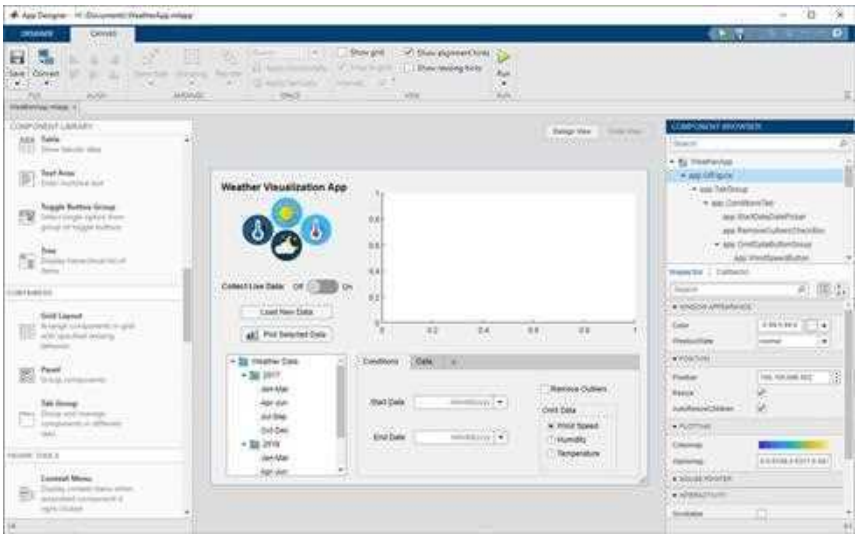


Figura 3.2: Interfaz de MATLAB App Designer.

## 3.4. MATLAB Compiler [29]

MATLAB Compiler es una herramienta avanzada que permite la conversión de aplicaciones y funciones de MATLAB en programas ejecutables, bibliotecas compartidas, componentes de software y aplicaciones web. Este proceso de compilación facilita la distribución de aplicaciones desarrolladas en MATLAB a usuarios que no disponen de una licencia de MATLAB, manteniendo la funcionalidad y el rendimiento del código original. MATLAB Compiler soporta la creación de ejecutables independientes que incluyen todos los archivos necesarios para ejecutar la aplicación, permitiendo una fácil implementación en distintos entornos.

Para este proyecto, MATLAB Compiler ha resultado especialmente útil por su capacidad para generar aplicaciones autónomas y distribuibles. Esta herramienta permite que las aplicaciones creadas en MATLAB se ejecuten en sistemas que no tienen MATLAB instalado, ampliando significativamente el alcance y la accesibilidad del software desarrollado. La capacidad de empaquetar y distribuir aplicaciones de forma eficiente ha sido crucial para facilitar el uso y la implementación del programa en diversos entornos operativos, garantizando al mismo tiempo la integridad y funcionalidad del código MATLAB.

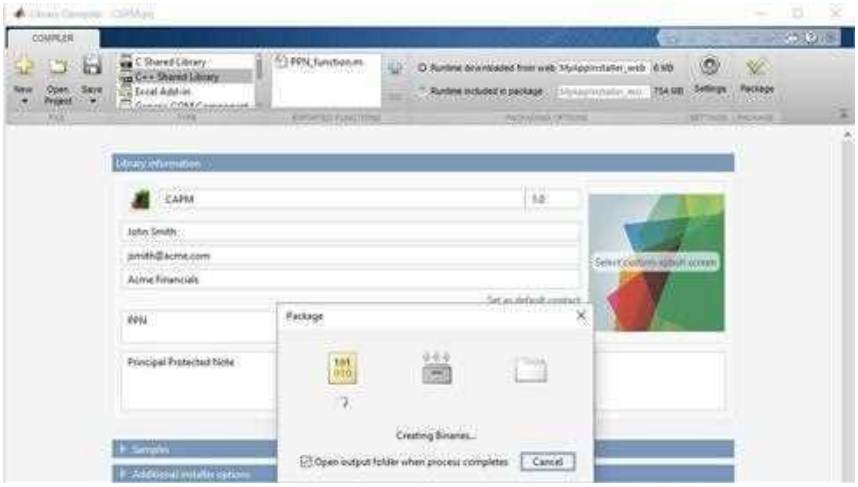


Figura 3.3: Interfaz de MATLAB Compiler.

## 3.5. Python [35]

Python es un lenguaje de programación de alto nivel conocido por su sintaxis clara y legible. Es interpretado y de tipado dinámico, lo que significa que el código se ejecuta línea por línea y las variables no requieren declaraciones explícitas de tipo. Python cuenta con una amplia biblioteca estándar que ofrece una variedad de módulos y funciones para realizar tareas comunes, y su naturaleza multiplataforma lo hace compatible con diferentes sistemas operativos. Gracias a su comunidad activa y a su popularidad en la industria y la academia, Python se ha convertido en una herramienta fundamental para el desarrollo de una amplia gama de aplicaciones, desde desarrollo web y análisis de datos hasta inteligencia artificial y automatización de tareas.

La mayoría del código desarrollado para este proyecto está escrito en este lenguaje. Su simplicidad, su amplia biblioteca estándar y su gran soporte para el desarrollo de aplicaciones e interfaces o GUI son las características que más se han tenido en cuenta a la hora de desarrollar la aplicación en este lenguaje.

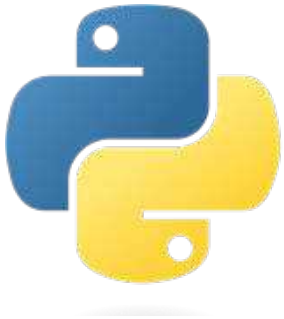

Figura 3.4: Logo de Python.

## 3.6. Anaconda [7]

Anaconda es una distribución de Python y R diseñada para simplificar la gestión de paquetes y la implementación de entornos de desarrollo. Es especialmente popular en los campos de la ciencia de datos, la inteligencia artificial y el análisis de datos debido a su capacidad para manejar bibliotecas y dependencias de manera eficiente. Anaconda incluye más de 1,500 paquetes de código abierto y herramientas como Jupyter Notebook, que facilitan la creación y ejecución de código interactivo.

Además, Anaconda ofrece Anaconda Navigator, una interfaz gráfica de usuario que permite a los usuarios instalar, actualizar y administrar paquetes y entornos con facilidad, sin necesidad de utilizar la línea de comandos. Esta combinación de gestión integral de paquetes y entornos, junto con herramientas de desarrollo interactivas, hace que Anaconda sea una elección ideal para desarrolladores y científicos de datos que buscan un entorno de trabajo robusto y eficiente.


Figura 3.5: Logo de Anaconda.

## 3.7. Spyder [2]

Spyder es un entorno de desarrollo integrado (IDE) diseñado específicamente para la programación en Python. Destaca por su enfoque en la ciencia de datos y la computación numérica, proporcionando herramientas especializadas para análisis, visualización y manipulación de datos. Spyder ofrece características como la exploración interactiva de datos, la integración con bibliotecas populares de Python como

NumPy, SciPy y Matplotlib, así como la capacidad de ejecutar y depurar código de manera eficiente. Además, cuenta con un editor de texto con resaltado de sintaxis, completado automático y otras características que facilitan la escritura de código Python. Spyder es ampliamente utilizado por científicos, ingenieros y analistas de datos para llevar a cabo proyectos de análisis de datos y desarrollo de aplicaciones en Python.

Las herramientas de depuración de Spyder han resultado especialmente útiles para el desarrollo de este proyecto.


Figura 3.6: Logo de Spyder.

## 3.8. PuTTY [47]

PuTTY es un programa de código abierto que proporciona una terminal de red, un cliente SSH y Telnet, así como herramientas de transferencia de archivos SCP y SFTP. Es compatible con varios protocolos de red, lo que lo convierte en una herramienta versátil para administrar sistemas remotos y realizar tareas de administración de redes. PuTTY es ampliamente utilizado en entornos de tecnología de la información y administración de sistemas, permitiendo a los usuarios acceder de forma segura a servidores remotos, administrar dispositivos de red y transferir archivos de manera eficiente. Ofrece una interfaz simple y fácil de usar, lo que lo convierte en una opción popular entre administradores de sistemas y desarrolladores que necesitan acceder y controlar equipos remotos de manera efectiva.

Figura 3.7: Logo de PuTTY.

## 3.9. Xming [3]

Xming es un servidor de pantalla de código abierto que permite ejecutar aplicaciones gráficas de forma remota en sistemas basados en Windows. Funciona como un servidor X para el entorno de escritorio X Window System, que es comúnmente utilizado en sistemas operativos basados en Unix y Linux para manejar la interfaz gráfica de usuario (GUI). Xming permite a los usuarios ejecutar aplicaciones gráficas en un servidor remoto y mostrarlas en su escritorio de Windows de manera transparente, lo que facilita el acceso y la ejecución de programas que requieren una interfaz gráfica, incluso cuando el sistema remoto no tiene un entorno de escritorio completo. Esto lo hace especialmente útil en entornos de administración de sistemas y desarrollo de software, donde se necesitan herramientas gráficas en sistemas remotos basados en Unix o Linux desde una computadora con Windows.

Figura 3.8: Logo de Xming.

## 3.10. FileZilla [1]

FileZilla es un cliente de software de código abierto que proporciona funcionalidades para transferir archivos de y hacia servidores remotos a través de protocolos como FTP (File Transfer Protocol), SFTP (SSH File Transfer Protocol), y FTPS (FTP sobre SSL/TLS). Permite a los usuarios gestionar fácilmente la transferencia de archivos entre su computadora local y un servidor remoto, lo que lo hace útil para administradores de sitios web, desarrolladores web y cualquier persona que necesite transferir archivos de manera segura y eficiente a través de una red.

Figura 3.9: Logo de FileZilla.

## 3.11. LaTeX [45]

LaTeX es un sistema de composición de textos de alta calidad ampliamente utilizado en la creación de documentos científicos y técnicos, como artículos de investigación, informes, tesis, libros y presentaciones. A diferencia de los procesadores de texto convencionales, LaTeX se centra en la estructura y el formato del documento, permitiendo a los usuarios concentrarse en el contenido mientras LaTeX se encarga de la presentación.

En LaTeX, los documentos se crean utilizando un lenguaje de marcado que incluye comandos y entornos para especificar el formato, la estructura y el estilo del texto. Esto permite a los usuarios realizar tareas avanzadas de composición de texto, como la creación de ecuaciones matemáticas complejas, la generación de tablas y gráficos profesionales, la gestión de bibliografías y referencias cruzadas, y la personalización detallada del diseño del documento.

Todo el presente documento ha sido elaborado haciendo uso de este sistema.

Figura 3.10: Logo de LaTeX.

## 3.12. TeXstudio [43]

TeXstudio es un editor de LaTeX de código abierto que facilita la escritura de documentos en el lenguaje de marcado LaTeX. Este entorno de desarrollo integrado (IDE) proporciona una amplia gama de funciones para mejorar la eficiencia y la comodidad de los usuarios al redactar documentos científicos, técnicos y académicos. Entre sus características destacadas se incluyen la sintaxis destacada, la corrección ortográfica, la autocompletación de comandos y una vista previa integrada que permite ver los resultados en tiempo real.

Además, TeXstudio ofrece herramientas de navegación y referencia que simplifican la gestión de grandes proyectos de LaTeX. Su interfaz amigable permite a los usuarios acceder rápidamente a plantillas, bibliografías y gráficos, facilitando así la organización y estructuración del documento. La capacidad de personalización y las extensiones adicionales hacen de TeXstudio una herramienta versátil y poderosa para cualquier persona que trabaje regularmente con LaTeX.


Figura 3.11: Logo de TeXstudio.

## 3.13. ADNI [4]

El Alzheimer's Disease Neuroimaging Initiative (ADNI) es un proyecto de investigación colaborativa a gran escala que tiene como objetivo comprender mejor la progresión de la enfermedad de Alzheimer. Fundado en 2004, el ADNI recopila y analiza datos de imágenes cerebrales, biomarcadores y pruebas neuropsicológicas de participantes a lo largo del tiempo. Este proyecto se centra en identificar los cambios tempranos en el cerebro asociados con la enfermedad de Alzheimer, con el fin de desarrollar métodos de diagnóstico más precisos y tratamientos efectivos.

El ADNI incluye a investigadores de universidades, centros médicos y la industria farmacéutica, promoviendo una colaboración interdisciplinaria. Los datos recopilados por el ADNI están disponibles públicamente para la comunidad científica, lo que facilita una amplia gama de estudios y descubrimientos en el campo de la neurociencia. Esta iniciativa ha contribuido significativamente al avance del conocimiento sobre el Alzheimer, acelerando la investigación y desarrollo de nuevas terapias y enfoques diagnósticos.


Figura 3.12: Logo de ADNI.

# Capítulo 4

# Métodos

## 4.1. Introducción

En este capítulo se reúne y explica toda la metodología utilizada para desarrollar la aplicación en la que se enfoca este proyecto y sus dos variantes.

En primer lugar, se muestra la estructura de la aplicación principal realizada y sus dos variaciones. A continuación, se explican las funciones relacionadas con graficar los resultados generados por el algoritmo utilizado. Luego, se describe la adaptación del modelo RPDPM realizada en MATLAB para obtener las predicciones. Posteriormente, se indica el modelo de Leaspy que ha sido utilizado en una de las variaciones. Después, se exponen los métodos utilizados para conectar los algoritmos con el resto del programa. Por último, se explican todas las clases y funciones asociadas con la interfaz de la aplicación: la parte más importante del proyecto.

Todo el código desarrollado para la aplicación principal y sus dos variantes se encuentra alojado en la carpeta *demo*. A lo largo del capítulo se hará referencia a este directorio para mostrar dónde se puede encontrar el código desarrollado para cada sección.

## 4.2. Estructura de la aplicación principal y sus variaciones

La aplicación principal desarrollada como objetivo de este trabajo consiste en una GUI creada en Python que permite la introducción de datos individuales de un paciente con el propósito de estudiar la evolución de una enfermedad neurodegenerativa, específicamente el Alzheimer (AD). La interfaz está diseñada para ser lo más intuitiva posible y se comunica con un modelo capaz de realizar predicciones, las cuales se muestran en forma de gráficos a través de la misma interfaz.

Dos objetivos principales de este trabajo eran que la aplicación fuera multiplataforma y que se pudiera cambiar el modelo utilizado sin mucha dificultad, facilitando la escalabilidad y la actualización del programa. Para demostrar esto, se ha creado una versión para Windows64 y otra para Linux, y se han desarrollado dos variantes de la aplicación principal: una que utiliza un motor distinto y otra realizada en otro lenguaje de programación.

Mientras que la aplicación principal está desarrollada mayoritariamente en Python, el motor que utiliza, RPDPM, está programado en MATLAB. Esto implica una tediosa comunicación entre los dos lenguajes de programación, por lo que, para evitar esto, se decidió crear una variante desarrollada en su totalidad en MATLAB que mantenga el mismo algoritmo para la obtención de resultados y que cuente con una interfaz lo más similar posible a la original. En cuanto a la segunda variación, esta comparte el código para la interfaz y la visualización con el programa principal, pero utiliza un motor basado en Leaspy como medio de obtención de los resultados.

A continuación, se muestra un esquema sencillo de la estructura de estos tres programas para facilitar su entendimiento:

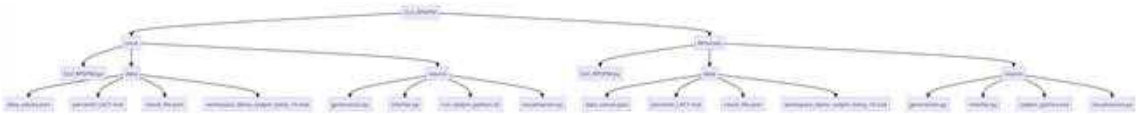

Figura 4.1: Estructura de 'GUI_RPDPM', la aplicación principal.

Figura 4.2: Estructura de 'GUI_Leaspy', la versión que utiliza un algoritmo de Leaspy como motor.

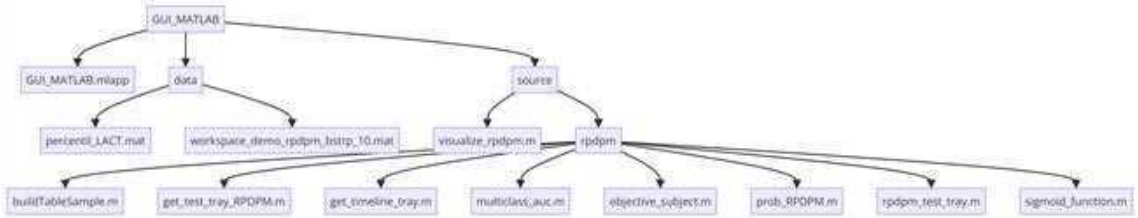


Figura 4.3: Estructura de 'GUI_MATLAB', la variante del programa principal desarrollada en MATLAB.

Por otro lado, de forma complementaria a esta memoria se ha elaborado una demo del funcionamiento de todas las aplicaciones desarrolladas para el presente proyecto. Esta demo se compone de los códigos fuente de las tres aplicaciones además de material adicional que también ha sido creado para el trabajo o va a ser utilizado para alguna explicación realizada durante el documento. La estructura de la carpeta *demo* es la siguiente:

```
+---Anexo
|   |   paciente_ejemplo.xlsx
|   |   rpdpm_python.m
|   |   Informe_Turnitin.pdf
|   |
|   +---Ejemplo_AppDesigner
|   |       app_mldesigner.mlapp
|   |
|   +---Ejemplo_Tkinter
```

```
|   |       app_tkinter.py
|   |
|   +---MatlabEngine_MATLAB
|   |       llamar_funcion_python.m
|   |       mi_modulo.py
|   |
|   \---MatlabEngine_Python
|       |   MatlabEngine.py
|       |   sumColumns.m
|       |
|       \---data
|               miArchivo.mat
|
+---GUI_Leaspy
|   |   GUI_Leaspy.py
|   |
|   +---data
|   |       dps_train1.csv
|   |       ...
|   |       dps_train10.csv
|   |       idx_train1.csv
|   |       ...
|   |       idx_train10.csv
|   |       id_train1.csv
|   |       ...
|   |       id_train10.csv
|   |       leaspy_btstrp_1.json
|   |       ...
|   |       leaspy_btstrp_10.json
|   |       train.csv
|   |
|   \---source
|           bayes_train.py
|           generacion.py
|           interfaz.py
|           leaspy_alg.py
|           leaspy_engine.py
|           percentile_interp.py
|           visualizacion.py
|
+---GUI_MATLAB
|   |   GUI_MATLAB.mlapp
|   |
|   +---data
|   |       percentil_LACT.mat
|   |       workspace_demo_rpdpm_bstrp_10.mat
|   |
|   \---source
```

```
|       |   visualize_rpdpm.m
|       |
|       \---rpdpm
|               buildTableSample.m
|               get_test_tray_RPDPM.m
|               get_timeline_tray.m
|               multiclass_auc.m
|               objective_subject.m
|               prob_RPDPM.m
|               rpdpm_test_tray.m
|               sigmoid_function.m
|
+---GUI_RPDPM
|   +---Linux
|   |   |   GUI_RPDPM.py
|   |   |
|   |   +---data
|   |   |       data_values.json
|   |   |       percentil_LACT.mat
|   |   |       result_file.json
|   |   |       workspace_demo_rpdpm_bstrp_10.mat
|   |   |
|   |   \---source
|   |           generacion.py
|   |           interfaz.py
|   |           run_rpdpm_python.sh
|   |           visualizacion.py
|   |
|   \---Windows
|       |   GUI_RPDPM.py
|       |
|       +---data
|       |       data_values.json
|       |       percentil_LACT.mat
|       |       result_file.json
|       |       workspace_demo_rpdpm_bstrp_10.mat
|       |
|       \---source
|               generacion.py
|               interfaz.py
|               rpdpm_python.exe
|               visualizacion.py
|
\---Instaladores
    +---GUI_Leaspy
    |   |   GUI_Leaspy.spec
    |   |
    |   +---build
```

```
|   |
|   \---dist
|
+---GUI_MATLAB
|   +---Linux
|   |   |   GUI_MATLAB.prj
|   |   |
|   |   \---GUI_MATLAB
|   |       |
|   |       +---for_redistribution
|   |       |       MyAppInstaller_mcr.install
|   |       |
|   |       +---for_redistribution_files_only
|   |       |       GUI_MATLAB
|   |       |       run_GUI_MATLAB.sh
|   |       |
|   |       \---for_testing
|   |
|   \---Windows
|       |   GUI_MATLAB.prj
|       |
|       \---GUI_MATLAB
|           |
|           +---for_redistribution
|           |       MyAppInstaller_mcr.exe
|           |
|           +---for_redistribution_files_only
|           |       GUI_MATLAB.exe
|           |
|           \---for_testing
|
\---GUI_RPDPM
    |   GUI_RPDPM.spec
    |
    +---build
    |
    \---dist
```

Además de los códigos fuente de las aplicaciones, en la carpeta *demo* también se encuentran la carpeta *Instaladores*, donde se pueden consultar los ejecutables e instaladores creados para los tres programas, y la carpeta *Anexo*, donde se han colocado todos los scripts que se tratan en el capítulo con el mismo nombre. Estos archivos se encuentran en una carpeta aparte porque no pertenecen al desarrollo principal del proyecto.

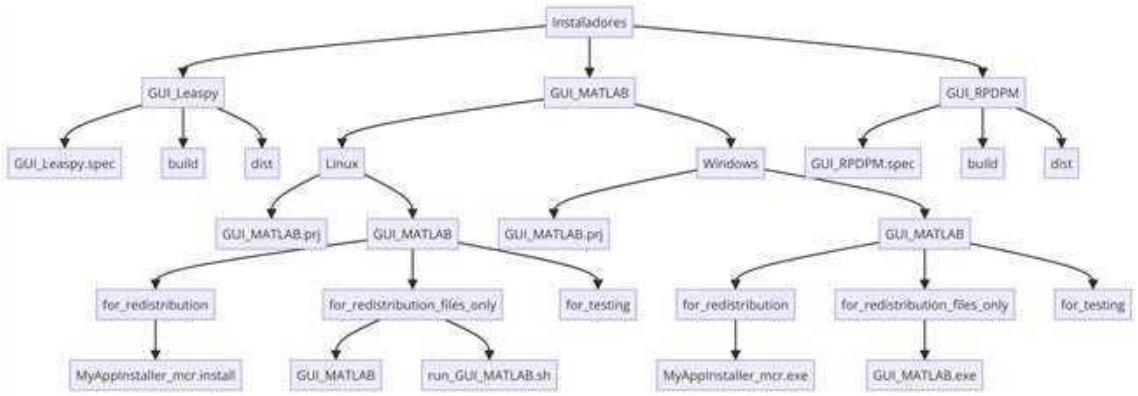


Figura 4.4: Carpeta 'Instaladores'.

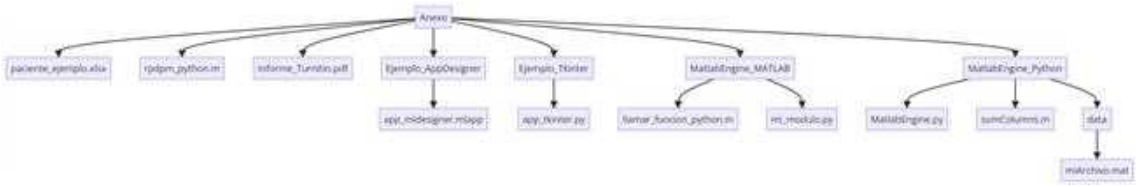


Figura 4.5: Carpeta 'Anexo'.

## 4.3. Visualización de las trayectorias

Uno de los primeros objetivos a la hora de realizar la aplicación consistió en la elaboración de un programa capaz de presentar gráficamente los resultados del algoritmo, es decir, las predicciones del avance de la enfermedad en el sujeto. Para ello, en primer lugar se partió de una demo en MATLAB que realiza esta tarea de forma simple y a partir de una información previamente establecida para cada paciente. En este caso, la información proviene de 1518 sujetos de ADNI, por lo que el programa se limita a permitir al usuario elegir entre uno de estos para mostrarle sus predicciones mediante figuras de MATLAB.

Como el objetivo era incluir esta funcionalidad en el programa final de Python, el primer paso fue obtener una versión funcional de esa demo en este lenguaje. Para llevar esto a cabo, la librería matplotlib.pyplot resultó ideal. Esta biblioteca incluye funciones que tratan de asemejarse a las utilizadas en MATLAB para la creación de gráficos y visualización de datos. De forma similar, se basa en crear figuras y modificarlas con gran libertad para después presentar el resultado de la forma que el usuario desee. Esto se consideró muy conveniente por dos razones: facilitaba en gran medida la migración de las funciones de MATLAB a Python manteniendo una buena compatibilidad con otras bibliotecas de Python como pandas o numpy y permitía una buena interacción con la interfaz que iba a ser implementada.

Todas las funciones asociadas a la visualización de las trayectorias se encuentran en *demo/GUI_RPDPM/Windows/source/visualizacion.py*. Estas mismas funciones

se utilizan también para la versión de Linux del programa principal en *demo/-GUI_RPDPM/Linux/source/visualizacion.py* y para la variante que utiliza el motor de Leaspy en *demo/GUI_Leaspy/source/visualizacion.py*. Seguidamente, se muestra un pseudocódigo de la migración que se realizó de la demo de MATLAB:

**Algorithm 1** Visualización de las trayectorias del avance de enfermedades neurodegenerativas

**Require:** Los archivos percent.xlsx, point_feat.xlsx, tau.xlsx, tSample.xlsx y tTrayec.xlsx y la selección por teclado del individuo cuya predicción se desea obtener.

**Ensure:** Presentación de la puntuación de los biomarcadores del paciente a lo largo del tiempo y del avance esperado de estos a lo largo de los próximos años, según las predicciones del algoritmo. También se muestra información de interés como el MAE y el diagnóstico del paciente en cada punto del tiempo.

**Importar** numpy, pandas y matplotlib.pyplot

**function** PLOT_INDIVIDUAL_ORIGINAL_LACT($tau, tSample, tTrayec, markers\_name$)
Genera un gráfico individual para cada marcador específico, comparando los datos de muestra con las estimaciones correspondientes.
**end function**

**function** PLOT_INDIVIDUAL_NORM_LACT($tau, tSample, tTrayec, markers\_name, change\_sign\_feat, ...$)
$..., percent, point\_feat$)
Representa las trayectorias normalizadas de los biomarcadores presentados por la función anterior en relación con la edad.
**end function**

**function** CALCULATE_MAE($samples, estimate$)
Se encarga de calcular y devolver el MAE (error absoluto medio) de los datos recibidos como argumento, es llamada por las dos funciones anteriores.
**end function**

**function** MAIN
Obtiene los datos iniciales necesarios a partir de los archivos excel, realiza unos pequeños ajustes en estos datos y ejecuta un bucle simple que, en cada iteración, solicita al usuario el id del paciente deseado y llama a las dos funciones anteriores. Este bucle se ejecuta hasta que el usuario lo interrumpa introduciendo el id '0'.
**end function**

A continuación, se explica más detalladamente el funcionamiento de las funciones `plot_individual_original_LACT` y `plot_individual_norm_LACT`.

### 4.3.1. `plot_individual_original_LACT`

- **Parámetros de entrada**
  - **tau**: Vector que contiene los valores de los marcadores.
  - **tSample**: Tabla que almacena los datos de muestra.
  - **tTrayec**: Tabla que contiene las estimaciones correspondientes a los marcadores.
  - **markers_name**: Vector que contiene los nombres de los marcadores que se desean visualizar.
- **Cálculo de estadísticas**
  - Se calcula la media y la desviación estándar de los valores de tau para determinar el punto medio (meanAge_MCI_RPDPM) y la dispersión de los datos.

- Si la media calculada es NaN (no es un número), se asigna el valor máximo de la edad de tTrayec a meanAge_MCI_RPDPM y se establece la desviación estándar como cero.

- **Diagnóstico clínico**
  - Se obtiene la información de diagnóstico clínico (DX) de la tabla tSample.
  - Si hay dos diagnósticos diferentes, se asigna 'pCU' como diagnóstico.
  - Se crean máscaras para distinguir entre diagnósticos de Cognitivamente Intacto (CU), Deterioro Cognitivo Leve (MCI) y datos faltantes.

- **Selección de marcadores**

  Se identifican las columnas correspondientes a los marcadores especificados en markers_name en las tablas tSample y tTrayec.

- **Generación del gráfico**
  - Se crea una figura con cuatro subgráficos (uno por cada marcador).
  - Para cada marcador:
    1. Se extraen los datos de muestra y estimación.
    2. Se trazan los puntos de muestra para CU (Cognitivamente Intacto), MCI y datos faltantes (NaN).
    3. Se trazan las estimaciones obtenidas de tTrayec.
    4. Se calcula el Error Absoluto Medio (MAE) entre los datos de muestra y las estimaciones.
    5. Se ajusta el rango de visualización del eje y para incluir todos los datos relevantes.
    6. Se traza una línea punteada que representa el punto medio calculado (meanAge_MCI_RPDPM) en el eje x.
    7. Se muestra el título del gráfico con información relevante como el ID de la muestra (RID), el diagnóstico clínico (diag) y el MAE calculado.
    8. Se etiquetan los ejes x e y con la edad y el nombre del marcador respectivamente.
    9. Se muestra una leyenda que indica los grupos de diagnóstico (CU, MCI) y las estimaciones.

- **Resultados**

  La función devuelve un gráfico que compara los datos de muestra con las estimaciones para cada marcador especificado.

### 4.3.2. `plot_individual_norm_LACT`

- **Parámetros de entrada**
  - **tau**: Vector que contiene los valores de los marcadores.
  - **tSample**: Tabla que almacena los datos de muestra.
  - **tTrayec**: Tabla que contiene las estimaciones correspondientes a las trayectorias de los marcadores.

- **markers_name**: Vector que contiene los nombres de los marcadores que se desean visualizar.
- **change_sign_feat**: Lista booleana que indica si se debe cambiar el signo de los rasgos.
- **percent**: Lista de vectores que especifican los porcentajes de normalización.
- **point_feat**: Lista de vectores que especifican los puntos de características.

- **Cálculo de estadísticas**
  - Se calcula la media de los valores de tau para determinar el punto medio (mean_age_MCI_RPDPM) de la edad de los individuos.
  - Si la media calculada es NaN (no es un número), se asigna el valor máximo de la edad de tTrayec a mean_age_MCI_RPDPM.

- **Diagnóstico clínico**
  - Se obtiene la información de diagnóstico clínico (DX) de la tabla tSample.
  - Si hay dos diagnósticos diferentes, se asigna 'pCU' como diagnóstico. De lo contrario, se asigna el diagnóstico encontrado.

- **Normalización de trayectorias**
  - Se obtienen los nombres de las columnas de tTrayec y se procesan para obtener los nombres base de los marcadores.
  - Para cada marcador especificado:
    1. Se identifican las columnas correspondientes en tTrayec.
    2. Se calcula la trayectoria promedio del marcador.
    3. Si change_sign_feat[k] es False, se interpola la trayectoria promedio entre los puntos de características y los porcentajes especificados.
    4. Si change_sign_feat[k] es True, se invierte el signo de la trayectoria promedio antes de la interpolación.
    5. Se traza la trayectoria normalizada contra la edad.

- **Configuración del gráfico**
  1. Se establece el rango del eje y entre 0 y 1 para representar la normalización.
  2. Se agrega una línea punteada para indicar el punto medio de la edad (mean_age_MCI_RPDPM).
  3. Se establece el título del gráfico con información relevante como el ID del individuo (RID) y el diagnóstico clínico (diag).
  4. Se etiquetan los ejes x e y con la edad y el nombre del marcador respectivamente.
  5. Se muestra una leyenda con los nombres de los marcadores.

- **Resultados**

  La función muestra un gráfico que representa las trayectorias normalizadas de los marcadores específicos en función de la edad para un individuo dado.

Después de completar esta migración, el siguiente paso consistió en desarrollar un programa capaz de calcular las mismas trayectorias utilizando cualquier conjunto de datos proporcionado por el usuario sobre el paciente. A pesar de que la complejidad del programa aumentó considerablemente, las funciones `plot_individual_original_LACT` y `plot_individual_norm_LACT`, responsables de generar los gráficos de los resultados, se mantuvieron sin cambios. Sin embargo, al integrar la interfaz, fue necesario realizar un ligero ajuste a la parte encargada de la visualización de las trayectorias para poder mostrarlas a través de la GUI. Estos ajustes se detallan en la sección *Desarrollo de la interfaz*.

Por otro lado, para realizar la GUI desarrollada completamente en MATLAB, fue necesario crear una versión de estas funciones que realizara la misma tarea utilizando funciones de MATLAB. Esto resultó ser una tarea relativamente sencilla, ya que se pudo partir de la demo de MATLAB mencionada al inicio de la sección para obtener unas funciones completamente equivalentes a las de Python. Estas se pueden consultar con el mismo nombre en el fichero *demo/GUI_MATLAB/source/visualize_rpdpm.m*.

## 4.4. Algoritmo RPDPM

Para lograr el objetivo de este proyecto, es fundamental la utilización de un algoritmo capaz de obtener las predicciones del avance del AD en un sujeto a partir de sus datos, concretamente la puntuación de una selección de biomarcadores a lo largo del tiempo. El algoritmo elegido para este fin es el 'Robust Parametric Disease Progression Modeling' (RPDPM).

Partiendo de un modelo de este algoritmo realizado en MATLAB, el primer paso fue obtener una adaptación en este mismo lenguaje del modelo. El resultado se encuentra en *demo/Anexo/rpdpm_python.m* y se explica a continuación:

- **Configuración inicial**

  Esta sección incluye el establecimiento de las rutas de acceso y la carga de datos iniciales necesarios para el algoritmo. Esto implica la carga de datos de porcentajes y características puntuales desde un archivo MATLAB (`percentil_LACT.mat`).

- **Decodificación de datos de entrada**

  La función lee un archivo JSON (`data_values.json`), decodifica su contenido y lo convierte en una tabla de MATLAB (data). Los nombres de las columnas se almacenan en la variable column_names. Posteriormente, se ajustan los tipos de datos de las columnas para garantizar la consistencia y la correcta interpretación de los datos.

- **Preparación de datos**

  En esta sección, se extraen campos específicos de los datos longitudinales y se organizan en una estructura adecuada para su procesamiento posterior. Se extraen identificadores de sujetos, etiquetas de estado de visita, edades de visita y valores de biomarcadores. Estos datos se organizan en matrices y arreglos tridimensionales para su posterior análisis.

- **Entrenamiento del modelo**

  Se cargan parámetros previamente optimizados (Sigma, A, B, C, D, G) y se realiza el entrenamiento del modelo RPDPM utilizando los datos preparados anteriormente. Se definen opciones de optimización para el proceso de ajuste.

- **Pruebas del modelo**

  Se realizan pruebas del modelo utilizando los datos de prueba preparados anteriormente. Se obtienen predicciones y se ajustan a una estructura de datos adecuada para su posterior procesamiento.

- **Clasificación utilizando el modelo**

  Se realizan pruebas de clasificación utilizando el modelo RPDPM. Se obtienen puntuaciones de predicción para cada sujeto y se comparan con las etiquetas de clase reales para evaluar el rendimiento del modelo.

- **Cálculo de la reserva cognitiva**

  Se calcula un índice de reserva cognitiva para cada sujeto utilizando los resultados obtenidos del modelo RPDPM y se almacena en la variable tau.

- **Escritura de resultados**

  Se estructuran los resultados finales en un formato adecuado y se convierten a formato JSON. Los resultados se guardan en un archivo JSON (`result_file.json`) para su posterior uso o análisis externo.

Para lograr una buena comunicación entre este script y el resto del programa, que está desarrollado en Python, fue necesario utilizar un sistema de comunicación por pipes y compilar el archivo para obtener `rpdpm_python.exe` y `run_rpdpm_python.sh` para la versión de Linux, que se pueden consultar en *demo/GUI_RPDPM/Windows/-source/rpdpm_python.exe* y *demo/GUI_RPDPM/Linux/source/run_rpdpm_python.sh*, respectivamente. Esto se explica detenidamente en la sección *Comunicación entre Python y MATLAB*.

En el caso de la variante de MATLAB, no es necesario compilar el archivo `rpdpm_python.m`; sin embargo, para esta aplicación es necesario añadir el código encargado de graficar los resultados, por lo que se creó el script alojado en *demo/GUI_MATLAB/source/-visualize_rpdpm.m*, que combina el motor RPDPM explicado en esta sección con las funciones de la sección *Visualización de las trayectorias*. Por esta razón, aunque el archivo `rpdpm_python.m` no se usa directamente en el proyecto, se ha incluido en la carpeta *Anexo* para que pueda ser consultado.

## 4.5. Algoritmo basado en Leaspy

Este algoritmo se realizó con el objetivo de encontrar una alternativa al RPDPM que se pueda desarrollar completamente en el lenguaje Python. Para ello, la biblioteca Leaspy resultó ser una opción ideal. Esta biblioteca proporciona herramientas avanzadas para la modelización y simulación de trayectorias longitudinales, permitiendo el ajuste de modelos a datos temporales de manera eficiente y precisa.

El motor desarrollado está diseñado para predecir la evolución de la enfermedad de Alzheimer utilizando modelos de aprendizaje automático personalizados con la biblioteca Leaspy. A continuación, se describe el funcionamiento técnico del motor, desglosando sus componentes clave y su interacción.

### 4.5.1. Componentes Principales

El motor de predicción se compone principalmente de tres módulos:

- `leaspy_engine.py`: Archivo principal que coordina el flujo de datos y las predicciones.
- `leaspy_alg.py`: Contiene funciones específicas para personalizar y ajustar modelos Leaspy.
- `percentile_interp.py`: Implementa funciones para interpolación de percentiles y balanceo de muestras.
- `bayes_train.py`: Archivo encargado del entrenamiento bayesiano de los modelos Leaspy.

Todos ellos se pueden consultar en *demo/GUI_Leaspy/source* junto con el resto de módulos de la aplicación.

### 4.5.2. `leaspy_engine.py`

Este módulo actúa como el núcleo del motor, coordinando la carga de modelos, la personalización de parámetros individuales y la generación de predicciones. Se encuentra en el directorio *demo/GUI_Leaspy/source/leaspy_engine.py*.

- **Carga de modelos**: Los modelos entrenados se cargan utilizando `Leaspy.load()`, permitiendo la reutilización de modelos preentrenados en diferentes bootstrap.
- **Personalización de parámetros**: La función `leaspyPersonalize` se utiliza para obtener parámetros individuales a partir de los datos de entrada.
- **Interpolación y estimaciones**: Utiliza interpolación lineal (`interp1d`) para ajustar las predicciones a los percentiles especificados. Esta técnica permite la extrapolación de datos y la transformación de características específicas.
- **Generación de la tabla de resultados**: Los resultados se organizan en un `DataFrame` de Pandas, que almacena las predicciones para cada individuo en diferentes puntos temporales.

A continuación, el desarrollo de la función `leaspy_table`:

**Algorithm 2** Función `leaspy_table`

```
Input: df_test, btstrp, timepoints, percent, point_feat
Output: leaspy_matrix
ids = df_test['ID'].unique()
leaspy_matrix = pd.DataFrame(columns=[...], index=range(len(ids) *
len(timepoints)))
for n in range(1, btstrp + 1) do
   leaspy_model = Leaspy.load(f'./data/leaspy_btstrp_n.json')
   _, ip = leaspyPersonalize(df_test, leaspy_model)
   for i, id in enumerate(ids) do
      estimation = leaspy_model.estimate({str(id): timepoints},
ip._individual_parameters)[str(id)]
      for j in range(estimation.shape[1]) do
         interp_func = interp1d(percent[j], point_feat[j], kind='linear',
fill_value='extrapolate')
         estimation[:, j] = interp_func(estimation[:, j])
         if j == 2 then
            estimation[:, j] = -estimation[:, j]            ▷ Ajuste específico para
RAVLT_learning
         end if
      end for
   end for
end for
return leaspy_matrix
```

### 4.5.3. `leaspy_alg.py`

Este módulo se centra en la personalización y ajuste de los modelos Leaspy. Está alojado en la carpeta *demo/GUI_Leaspy/source/leaspy_alg.py*.

- **Personalización del modelo**: La función `leaspyPersonalize` permite adaptar los modelos generales de Leaspy a los datos específicos de cada individuo.
- **Funciones auxiliares**: Se incluyen funciones para el ajuste de parámetros y la mejora del rendimiento del modelo.

### 4.5.4. `percentile_interp.py`

Este módulo proporciona funciones para la interpolación de percentiles y el balanceo de muestras, cruciales para la normalización y transformación de datos. Se puede consultar en *demo/GUI_Leaspy/source/percentile_interp.py*.

- **Interpolación de percentiles**: `percentile_interp` convierte características en percentiles, facilitando la comparación y el análisis estadístico.
- **Balanceo de muestras**: `balanceSamples` asegura que las muestras utilizadas estén equilibradas entre diferentes grupos, mejorando la robustez del modelo.
- **Función de distribución empírica**: `ecdf` calcula la función de distribución empírica para los datos proporcionados.

A continuación, el desarrollo de la función `percentile_interp`:

**Algorithm 3** Función `percentile_interp`

```
Input: features, group, percent=None, point_feat=None
Output: feat2per, percent, point_feat
num_features = features.shape[1]
feat2per = pd.DataFrame(np.zeros_like(features))
if percent is not None and point_feat is not None then
   for i in range(num_features) do
      interp_func = interp1d(point_feat[i], percent[i], kind='linear',
fill_value='extrapolate')
      feat2per.iloc[:, i] = interp_func(features.iloc[:, i])
   end for
else
   percent, point_feat = [], []
   for i in range(num_features) do
      mask_nan = ~features.iloc[:, i].isna()
      feat, prob = ecdf(balanceSamples(np.array(features.loc[mask_nan, i]),
group[mask_nan]))
      point_feat.append(feat)
      percent.append(prob)
      interp_func = interp1d(point_feat[i], percent[i], kind='linear',
fill_value='extrapolate')
      feat2per.iloc[:, i] = interp_func(features.iloc[:, i])
   end for
end if
return feat2per, percent, point_feat
```

### 4.5.5. `bayes_train.py`

Este módulo se encarga del entrenamiento bayesiano de los modelos Leaspy. Concretamente, se utiliza para obtener la probabilidad de que el sujeto pase a MCI 'probMCI' para cada punto de las predicciones. Está alojado en la carpeta *demo/-GUI_Leaspy/source/bayes_train.py*.

- **Inicialización del entrenamiento**: Configura los parámetros iniciales para el entrenamiento bayesiano, incluyendo las distribuciones previas y los hiperparámetros del modelo.

- **Proceso de entrenamiento**: Implementa el algoritmo de inferencia bayesiana para ajustar los parámetros del modelo a los datos observados, utilizando métodos como el muestreo de Gibbs y MCMC.

- **Evaluación del modelo**: Valida el modelo entrenado mediante técnicas de validación cruzada y calcula métricas de rendimiento para asegurar la precisión y robustez del modelo.

A continuación, se presenta una vista general del proceso de entrenamiento bayesiano en `bayes_train.py`:

**Algorithm 4** Proceso de entrenamiento bayesiano

```
Entrada: datos, distribuciones_previas, hiperparámetros
Salida: modelo_entrenado
Inicializar el modelo con distribuciones previas e hiperparámetros
for cada iteración do
    Actualizar parámetros del modelo usando muestreo de Gibbs
    Evaluar la verosimilitud del modelo con los parámetros actuales
end for
devolver modelo_entrenado
```

## 4.6. Comunicación entre Python y MATLAB

Aunque en primera instancia se trató con demos programadas en MATLAB, el objetivo en todo momento ha sido desarrollar la interfaz en el lenguaje de Python. No obstante, el algoritmo utilizado está desarrollado y entrenado en MATLAB, por lo que migrarlo a Python representaba un desafío considerable.

Ante esta dificultad, se pensó en buscar una solución que permitiera ejecutar el algoritmo, programado en MATLAB, desde la interfaz de Python. En este contexto, la API matlab.engine [30] surgió como la opción ideal. Esta solución no solo permite la ejecución paralela de funciones de MATLAB desde un entorno de Python, sino que también habilita una comunicación fluida entre ambos lenguajes. Esta comunicación resulta fundamental para enviar los datos requeridos por el algoritmo, así como para la recepción de sus resultados.

A continuación, se muestra un ejemplo básico de cómo se puede utilizar esta API:

**Algorithm 5** Comunicación entre MATLAB y Python usando matlab.engine

```
1: Importar matlab.engine, pandas y scipy.io
2: eng ← matlab.engine.start_matlab()                    ▷ Creación del motor de MATLAB
3: Crear un DataFrame data con algunas columnas
4: df ← pd.DataFrame(data)
5: scipy.io.savemat('data/miArchivo.mat', {'miTabla': df})
6: resultTable ← eng.sumColumns('data/miArchivo.mat')    ▷ Llamada a la función de MATLAB
7: resultCell ← eng.table2cell(resultTable) ▷ Función de matlab.engine para conversión de datos
8: result_df ← pd.DataFrame(resultCell, columns=['Resultado'])
9: Imprimir result_df
10: eng.quit()                                           ▷ Se limpia la memoria del motor de MATLAB
```

El ejemplo mostrado se puede consultar en *demo/Anexo/MatlabEngine_Python/-MatlabEngine.py*.

Cabe destacar que matlab.engine requiere de una instalación funcional de MATLAB con una licencia válida en el equipo. Esto no supuso ningún inconveniente durante el desarrollo del programa, sin embargo, a la hora de crear el ejecutable final aparecieron una serie de dificultades que provocaron el planteamiento de un cambio de estrategia. Ya que el programa estaba planteado para poder ser instalado en equipos con Microsoft Windows, surgió la idea de compilar la adaptación ya realizada del

algoritmo en un archivo `.exe` y ejecutarlo de forma similar desde el programa de Python.

Para llevar a cabo esta nueva estrategia, ahora la comunicación se realiza a través de pipes, donde se trata la información en formato JSON para facilitar la conversión de las variables de MATLAB a Python. En ambos lenguajes se debe hacer y deshacer esta conversión antes de enviar o recibir los datos. El módulo subprocess de Python es uno de los más extendidos para realizar tareas como esta, y resultó ideal en el caso de este programa, ya que el uso de las pipes simplificó significativamente la comunicación entre MATLAB y Python, logrando que solo sea necesario ejecutar el `.exe` sin interactuar mediante variables de entrada o de salida.

Esta implementación de subprocess se encuentra en la función `generate_graphs()` declarada en el archivo *demo/GUI_RPDPM/Windows/source/generación.py*. Por otro lado, la versión compilada de esta función que ejecuta subprocess, se encuentra en el directorio de la carpeta demo *demo/GUI_RPDPM/Windows/source/rpdpm_python.exe*.

En el caso de la versión para Linux, la adaptación fue muy similar, ya que el módulo subprocess permite hacer llamadas desde bash. Para esta versión, el motor compilado se encuentra en el archivo *demo/GUI_RPDPM/Linux/source/run_rpdpm_python.sh* y la gestión de subprocess se puede consultar en *demo/GUI_RPDPM/Linux/source/-generación.py*.

## 4.7. Desarrollo de la interfaz

La finalidad principal de este proyecto es desarrollar una interfaz o GUI que permita al usuario interactuar de forma simple e intuitiva con un algoritmo predictivo sin la necesidad de contar con conocimientos de programación o del funcionamiento de estos algoritmos. Ante este propósito, el paquete Tkinter de Python se presentó como la mejor herramienta por distintas razones:

- **Facilidad de uso:** Tkinter es conocido por su simplicidad y facilidad de uso. Ofrece una interfaz intuitiva y fácil de aprender, lo que hace que sea ideal para desarrolladores que deseen crear interfaces gráficas sin una curva de aprendizaje pronunciada. Para este proyecto en concreto, la libertad creativa que ofrece Tkinter es más que suficiente, por lo que su simplicidad se convierte en un factor de mucho peso.
- **Flexibilidad y capacidad de personalización:** A pesar de su simplicidad, Tkinter es una biblioteca flexible que permite la creación de interfaces gráficas altamente personalizadas. Ofrece una amplia gama de widgets y herramientas de diseño que permiten adaptar la interfaz a las necesidades específicas del proyecto.
- **Incorporación en Python:** Tkinter viene incluido con la instalación estándar de Python, lo que significa que no es necesario instalar bibliotecas adicionales para empezar a trabajar con él. Esto facilita la distribución y la portabilidad del programa, ya que no es necesario preocuparse por las dependencias externas. Durante el proceso de compilación del programa completo y la creación de

su instalador, esta ventaja resulta especialmente útil. De hecho, fue un factor fundamental en la elección de Tkinter como herramienta para desarrollar la interfaz.

Tras desarrollar esta interfaz utilizando Tkinter, se decidió crear una adaptación de la GUI en MATLAB, como se explicó previamente en este capítulo. Esta sección se ha dividido en dos partes para explicar el desarrollo de ambas: primero se aborda la interfaz en Python y, posteriormente, la de MATLAB.

### 4.7.1. Interfaz de Python

En el programa final, las clases y funciones relacionadas con la interfaz representan la mayor parte del código. En la función `main()`, que se encuentra en el archivo con la dirección *demo/GUI_RPDPM/Windows/GUI_RPDPM.py* o *demo/-GUI_RPDPM/Linux/GUI_RPDPM.py* en la versión para Linux, se declaran la ventana principal y una serie de widgets generales para el manejo de la interfaz, como un botón para finalizar la ejecución del programa y todos los widgets asociados a la tabla de datos sin estar directamente incluidos en esta. Por otro lado, se declara una instancia de la clase DataTable, que incluye la mayor parte de la lógica de la GUI. Esta clase proporciona una estructura para introducir y manipular los datos de los pacientes que más tarde serán enviados al algoritmo. La declaración de la clase DataTable se encuentra en el archivo *demo/GUI_RPDPM/Windows/source/interfaz.py* o *demo/GUI_RPDPM/Linux/source/interfaz.py* en la versión para Linux. A continuación, se muestra su estructura de forma esquemática mediante un diagrama UML y, posteriormente, se explica detalladamente su funcionamiento:

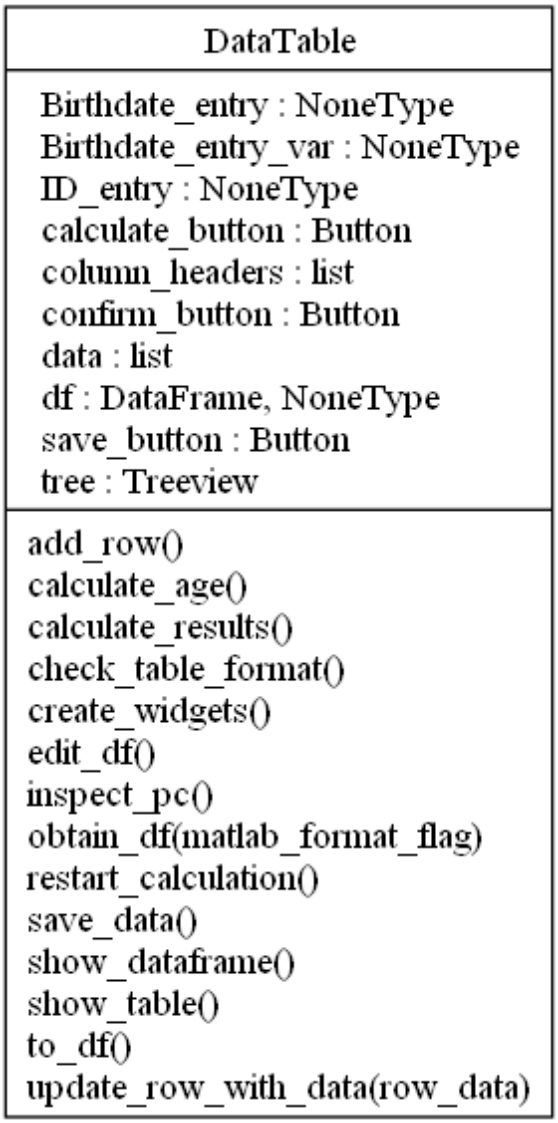


Figura 4.6: Diagrama UML de la clase DataTable.

**Explicación detallada del funcionamiento de DataTable:**

- **Inicialización**

  La clase se inicializa con varios parámetros, incluyendo un widget maestro (master), entradas para el ID y la fecha de nacimiento (ID_entry y Birthdate_entry, respectivamente), y una variable de entrada asociada con la fecha de nacimiento (Birthdate_entry_var). Esta última es la única variable de entrada incluida en la tabla que hace falta declarar, ya que su columna asociada debe ser completada con un formato de fecha correcto y esta variable es necesaria para verificarlo. Por otro lado, en la inicialización se llama al método `create_widgets()`, que se encarga de crear todos los elementos visuales necesarios para la tabla, incluyendo etiquetas, entradas, botones y el encabezado de la tabla y al método `add_row()` para agregar una fila inicial a la tabla.

- **Atributos**

- **df:** Un DataFrame de pandas que almacena los datos introducidos en la tabla.
- **column_headers:** Una lista de nombres de columnas que define las columnas de la tabla.
- **data:** Una lista que almacena los datos de la tabla en forma de filas, donde cada fila contiene una lista de entradas, una para cada celda en esa fila.
- **ID_entry, Birthdate_entry, Birthdate_entry_var:** Entradas y variable de entrada asociadas con el ID y la fecha de nacimiento del paciente. Son necesarias para el funcionamiento de la clase pero están asociadas a widgets que gráficamente no están incluidos en la tabla, por lo que están declarados en la función `main()` y sus handlers son pasados a la clase como atributos.
- **calculate_button, save_button, confirm_button:** Botones para calcular resultados, guardar datos y confirmar la información introducida, respectivamente. Declarados como atributos para poder acceder a ellos en el método `check_table_format()`.

▪ **Estructura de la tabla**

La tabla se compone de una matriz de entradas de Tkinter, junto con sus variables asociadas y etiquetas que indican de qué columna o fila se trata a través de la interfaz. Las columnas, determinadas por los elementos en la lista column_headers, son fijas y contienen los siguientes encabezados: **Exam date**, **Diagnosis**, **ADAS13**, **CDRSB**, **RAVLT**, **TRABSCOR** y **Age**. Modificar las columnas de las que consta la tabla sería tan simple como cambiar esta última lista. Por otro lado, las filas representan las visitas que realiza el sujeto a lo largo del estudio y se añaden dinámicamente mediante el método `add_row()` asociado al botón 'Add visit'. Esto permite que la tabla tenga un número variable de filas, según la cantidad de visitas que el usuario decida añadir. Mientras que el método `create_widgets()` se encarga de generar los encabezados de las columnas, es `add_row()` quien se ocupa de crear todos los elementos mencionados anteriormente (entradas, variables asociadas y etiquetas) y de agregarlos al atributo data.

▪ **Verificación del formato de la tabla**

El método `check_table_format()` verifica que los datos introducidos en la tabla cumplan con ciertos criterios, como el formato de fecha correcto, que el ID sea numérico, etc. Este método se ejecuta cada vez que se produce un cambio en los datos de la tabla para garantizar la integridad de los datos introducidos en todo momento. Hasta que esta función no compruebe que los datos encontrados en la tabla no cumplan con unos requisitos mínimos para ser enviados al algoritmo, no se muestran los botones 'Save data' y 'Confirm', utilizados para llevar a cabo dicha tarea. Los requisitos que valida son los siguientes:

- Verifica que las casillas de ID, fecha de nacimiento y todas las de fecha de la visita que existan en la tabla no estén vacías y tengan el formato correcto: el ID debe ser un valor numérico y tanto la fecha de nacimiento como las fechas de examen deben tener un formato de fecha 'DD/MM/AAAA' adecuado. Si se valida la condición anterior, el formato de los datos se considera correcto y se muestran los botones 'Save data' y 'Confirm'.

- Comprueba que el formato del resto de columnas de la tabla sea el adecuado: todas deben tener valores numéricos excepto la columna de diagnóstico, que debe tener un formato de cadena de texto. Estas columnas, a diferencia de las mencionadas en el primer punto, pueden estar vacías. Sin embargo, no pueden tener un formato incorrecto: si se validan las condiciones del primer punto pero alguna del resto de columnas tiene un formato incorrecto, se desactivan los botones 'Save data' y 'Confirm' hasta que esto cambie.

- **Cálculo de los resultados**

  El método `calculate_results()` se activa cuando se hace clic en el botón 'Calculate results'. Utiliza los datos de la tabla para obtener los resultados de las predicciones a través de la función `generate_graphs()`. Esta función, además de incluir la comunicación con el algoritmo explicada en su correspondiente sección, llama a `plot_individual_original_LACT` y a `plot_individual_norm_LACT`, explicadas en la sección de visualización de las trayectorias. Estas fueron modificadas para que hagan uso de FigureCanvasTkAgg, una clase en la biblioteca Matplotlib que proporciona una integración entre las figuras de Matplotlib y la interfaz de usuario de Tkinter. De esta forma, es posible insertar los resultados gráficos del algoritmo en la interfaz sin mucha dificultad.

- **Guardado de datos**

  El método `save_data()` permite al usuario guardar los datos introducidos en la tabla como un archivo Excel cuando se hace clic en el botón 'Save data'. Este método hace uso de la clase filedialog de Tkinter, que habilita cuadros de diálogo para que el usuario pueda inspeccionar su equipo y elegir la ubicación y nombre deseados para el archivo Excel.

- **Obtención del DataFrame**

  El método `obtain_df()` convierte los datos de la tabla en un DataFrame de pandas, listo para ser utilizado en otras partes de la aplicación. Esto resulta fundamental para tareas como obtener los datos iniciales del algoritmo. El método incluye una serie de transformaciones para que el DataFrame termine con el formato idóneo para ser utilizado en otras partes del programa.

- **Confirmación de datos**

  El método `to_df()` se activa cuando se hace clic en el botón 'Confirm'. Inicializa el DataFrame con los datos de la tabla y muestra la información para su revisión.

- **Cálculo de edad**

  El método `calculate_age()` calcula la edad del paciente en cada visita, utilizando la fecha de nacimiento y la fecha de cada visita, y muestra esta información en la columna 'Age' de la tabla.

- **Visualización del DataFrame**

  El método `show_dataframe()` muestra el DataFrame generado a partir de los datos de la tabla en un widget Treeview para su visualización y revisión. Los datos presentados de esta forma están cargados y listos para ser enviados al algoritmo mediante el botón 'Calculate Results'.

- **Edición de la tabla mediante un archivo Excel**

  De la misma forma que el método `save_data()`, el método `inspect_pc()`, hace uso de la clase filedialog para permitir al usuario inspeccionar su equipo en busca de un archivo Excel con la información de un paciente. Una vez elegido un archivo con un formato válido, se completa la tabla con su información para ofrecer la posibilidad de editarla antes de hacer clic en el botón 'Confirm' para confirmar los datos.

### 4.7.2. Interfaz de MATLAB

El objetivo al crear esta adaptación en MATLAB fue realizarla lo más similar posible a la original en Python, manteniendo la esencia de una interfaz simple y amigable para el usuario. Al buscar la herramienta idónea para este trabajo en MATLAB, App Designer resultó ser perfecta. MATLAB App Designer permite la creación de aplicaciones profesionales ofreciendo una gran facilidad de uso, ya que solo hace falta arrastrar y colocar los componentes deseados y editar sus parámetros hasta lograr el resultado buscado. La implementación final desarrollada se puede consultar en el directorio *demo/GUI_MATLAB/GUI_MATLAB.mlapp*. A continuación, se explican todos los componentes de diseño que forman la aplicación. Posteriormente, se detalla el funcionamiento desde la perspectiva del código, destacando los componentes más importantes y cómo están programados.

#### Diseño de la aplicación

Debido a que la herramienta se basa en una edición principalmente gráfica, para explicar esta parte se utilizarán capturas de pantalla del editor de MATLAB App Designer. El resultado final de la interfaz, incluyendo todos sus componentes, es el siguiente:

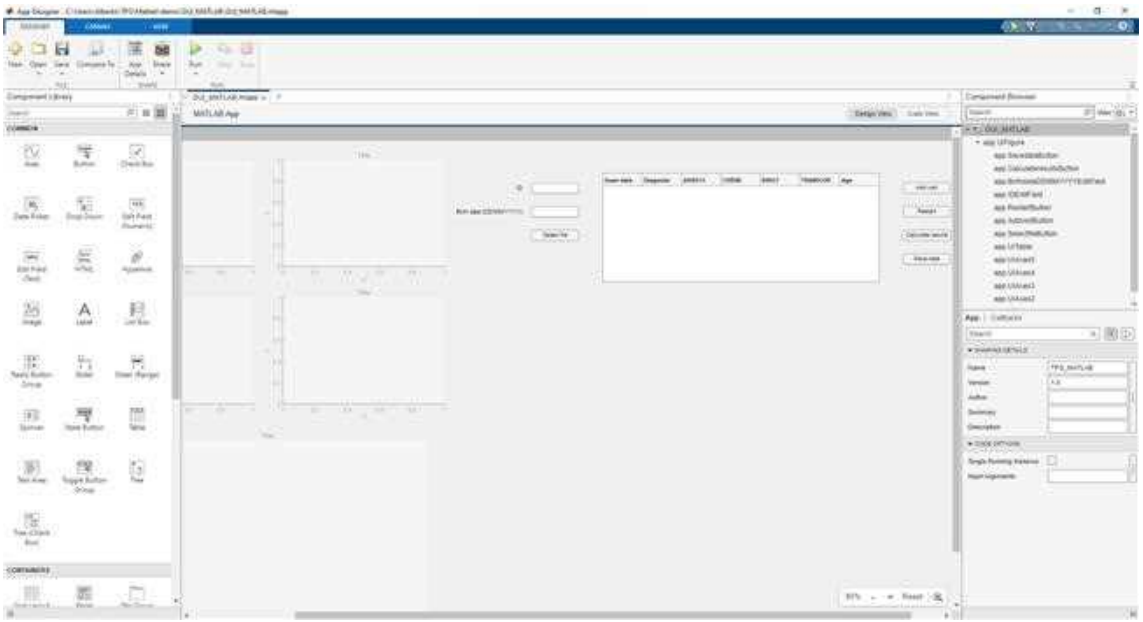

Figura 4.7: Editor de MATLAB App Designer mostrando todos los componentes de la interfaz.

En la interfaz de App Designer destacan los siguientes elementos:

- **Barra superior**: Esta sección permite configurar el espacio de trabajo principal en términos generales. Para realizar esta aplicación, se han utilizado principalmente sus funciones para ejecutar y depurar el programa.

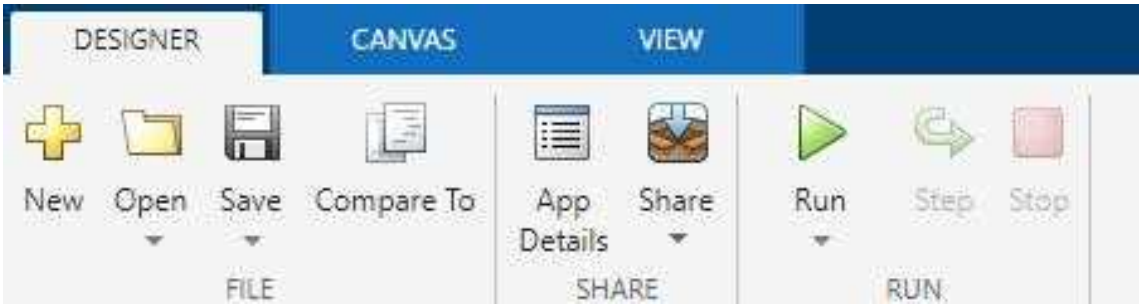


Figura 4.8: Barra superior de MATLAB App Designer.

- **Biblioteca de componentes**: En esta sección se encuentran los diferentes componentes y controles que pueden ser arrastrados y soltados en el área de diseño de la GUI. Incluye elementos como botones, campos de texto, cuadros desplegables, tablas, etc. Es una especie de çaja de herramientas"para construir la interfaz de usuario. Cuenta con una amplia variedad de componentes que permiten lograr una alta calidad de diseño y personalización, manteniendo la facilidad y rapidez que caracterizan a la herramienta.

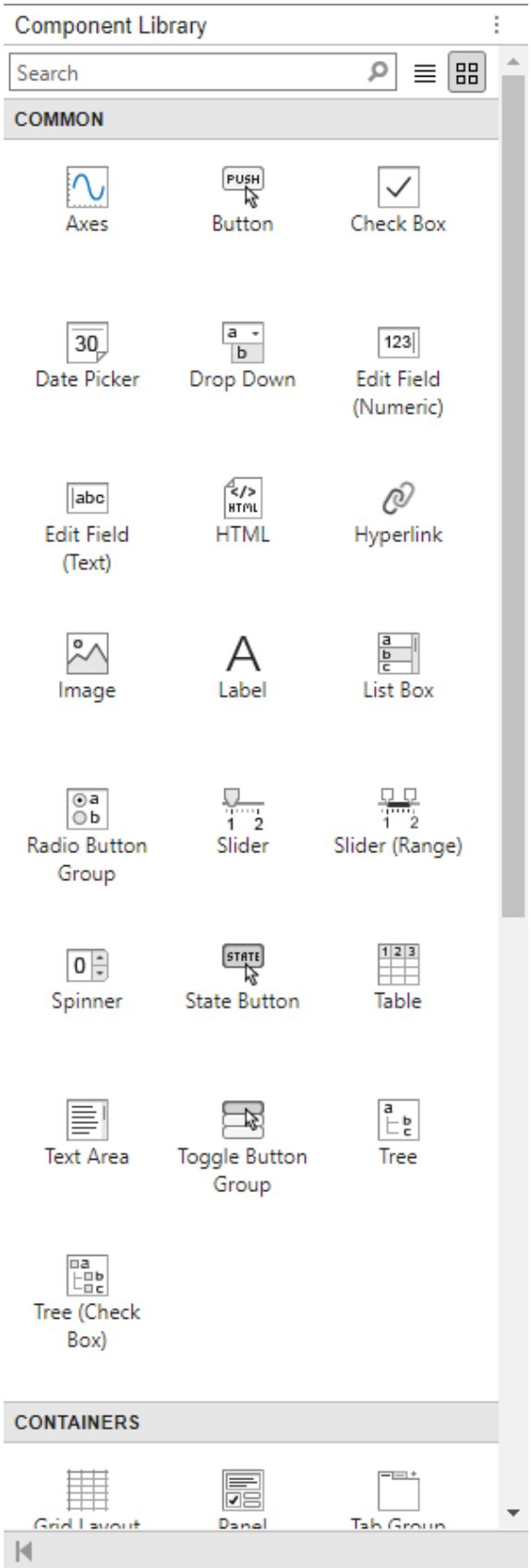


Figura 4.9: Biblioteca de componentes de MATLAB App Designer.

- **Navegador de componentes**: En esta parte se muestra la estructura jerárquica de todos los componentes que han sido agregados a la GUI. A través de esta vista, se puede acceder a una lista organizada de todos los controles y componentes dentro de la aplicación. Además, permite seleccionar cualquier componente para modificar sus propiedades, ajustar sus callbacks, y gestionar la organización y el agrupamiento de los elementos en la interfaz.

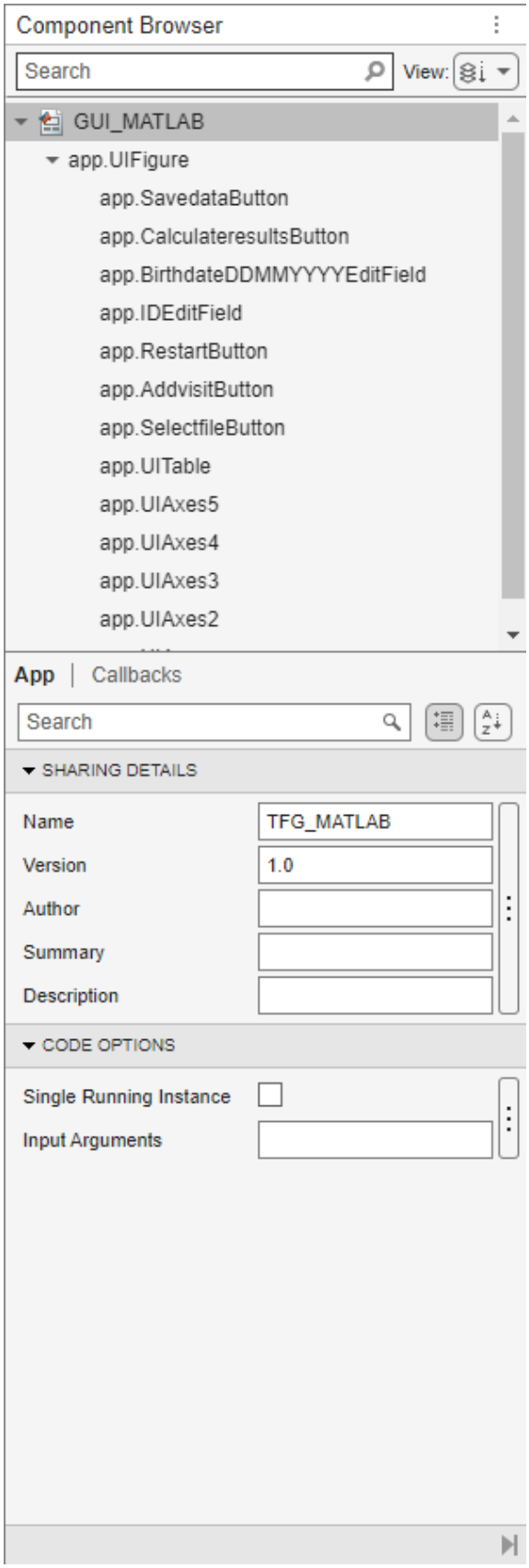


Figura 4.10: Navegador de componentes de MATLAB App Designer.

Al seleccionar un componente de esta lista, aparece un menú para editar sus propiedades con gran detalle. A continuación, se muestra un ejemplo:

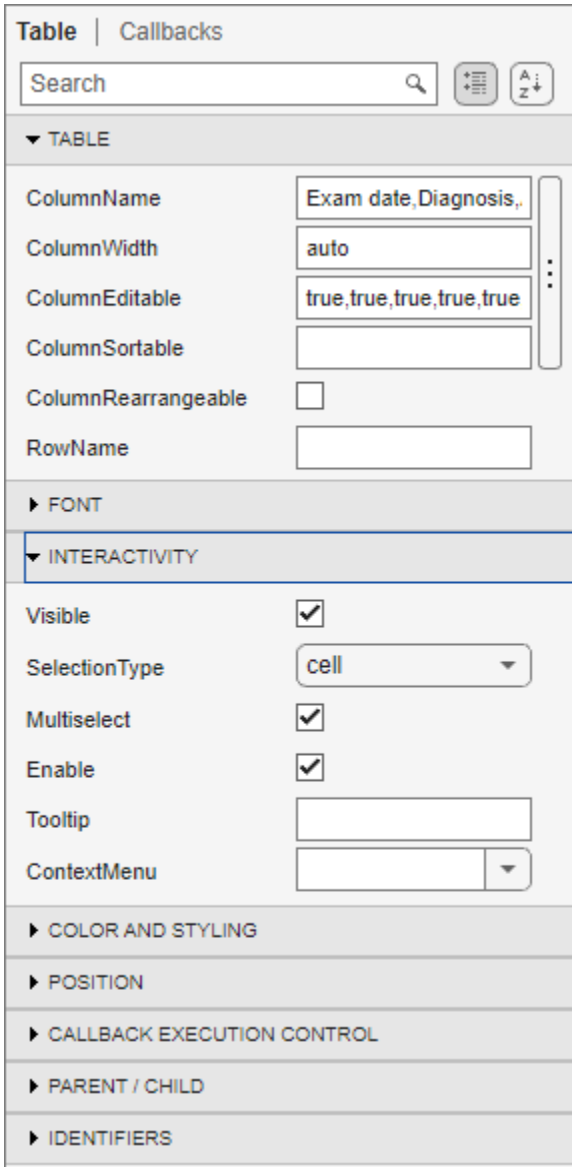


Figura 4.11: Edición de propiedades de un componente de MATLAB App Designer.

A continuación, se nombran y explican en detalle todos los componentes que forman esta aplicación.

#### Componentes principales

- **UIFigure**: Es la ventana principal de la aplicación donde se alojan todos los componentes gráficos. La posición y el tamaño de esta ventana se establecen inicialmente, y se configura para maximizarse al abrirse.

- **Botones de interacción** (SavedataButton, CalculateresultsButton, AddvisitButton, SelectfileButton, RestartButton): Estos botones permiten al usuario

realizar diversas acciones, como guardar datos, calcular resultados, agregar visitas, seleccionar archivos y reiniciar la aplicación. Cada botón está asociado a una función de callback específica que define su comportamiento.

- **Campos de entrada de texto** (IDEditField, BirthdateDDMMYYYYEditField): Permiten al usuario ingresar datos básicos del paciente, como el ID y la fecha de nacimiento.
- **UITable**: Es una tabla interactiva que muestra los datos médicos longitudinales del paciente. Esta tabla puede ser editada por el usuario para ingresar o modificar datos de visitas.
- **Componentes de representación** (UIAxes, UIAxes2, UIAxes3, UIAxes4, UIAxes5): Son áreas de gráficos donde se visualizan los resultados y análisis de los datos médicos. Cada uno de estos se utiliza para mostrar gráficos específicos relacionados con la progresión de la enfermedad. Están configurados para permanecer ocultos al iniciar el programa y solo se muestran una vez que se han terminado de calcular los resultados de las predicciones.

Aunque para gran parte del desarrollo con App Designer, concretamente para el diseño gráfico, no es necesario tener conocimientos de programación, sí es necesario escribir código para completar los callbacks asociados a cada botón de interacción. A continuación, se explican las acciones programadas en cada uno de estos, además de otras funciones importantes del programa.

#### Programación y funcionalidad

- **Inicialización de la aplicación**: Al crear la GUI, se llama a la función `startupFcn`, que inicializa los datos de la tabla. Se establece una matriz vacía como datos iniciales para UITable.
- **Agregar visitas**: La función `AddvisitButtonPushed` se ejecuta cuando se presiona el botón 'Add visit'. Incrementa un contador de visitas, crea una nueva fila vacía en la UITable y actualiza el encabezado de las filas para reflejar el número de visitas.
- **Reiniciar la información introducida**: La función `RestartButtonPushed` se ejecuta cuando se presiona el botón 'Restart'. Esta función vacía los campos de entrada de texto, reinicia el contador de visitas y restablece los datos de la tabla a su estado inicial.
- **Seleccionar y cargar archivo**: La función `SelectfileButtonPushed` se ejecuta cuando se presiona el botón 'Select file'. Permite al usuario seleccionar un archivo Excel, leer sus datos y actualizar UITable con los datos del archivo. También establece los valores de los campos de entrada de texto según los datos del archivo.
- **Calcular resultados**: La función `CalculateresultsButtonPushed` se ejecuta cuando se presiona el botón 'Calculate results'. Esta función obtiene los datos de la tabla, realiza cálculos específicos y visualiza los resultados en los ejes gráficos (UIAxes), que se mantienen ocultos hasta que los muestra esta función. También organiza y prepara los datos para un análisis posterior.

- **Guardar datos**: La función `SavedataButtonPushed` se ejecuta cuando se presiona el botón 'Save data'. Obtiene los datos de UITable, junto con los valores de ID y fecha de nacimiento, y los guarda en un archivo Excel en la ubicación especificada por el usuario.

## 4.8. Creación de los ejecutables y los instaladores

Durante el desarrollo de este proyecto, se han desarrollado aplicaciones en Python y en MATLAB. Para asegurar que los usuarios finales pudieran ejecutar estos programas sin necesidad de configurar entornos de desarrollo o instalar dependencias adicionales, se optó por convertir los scripts en ejecutables autónomos. Este proceso se realizó utilizando herramientas específicas: PyInstaller para los programas en Python y MATLAB Compiler para el programa en MATLAB. El resultado se puede consultar en la carpeta *demo/Instaladores*. A continuación, se describe el proceso seguido para crear los ejecutables e instaladores de cada uno de estos programas.

### 4.8.1. Programas en Python

Para los dos programas desarrollados en Python, se utilizó PyInstaller, una herramienta que convierte scripts de Python en ejecutables autónomos.

- **Instalación de PyInstaller**: Para instalar el paquete PyInstaller en el entorno de desarrollo, se puede utilizar el gestor de paquetes `pip` de forma sencilla:

  ```
  pip install pyinstaller
  ```

- **Generación del ejecutable**: Una vez instalado el paquete en el sistema, para obtener el instalador se debe ejecutar el siguiente comando en la terminal, especificando el script principal de cada programa:

  ```
  pyinstaller --onefile nombre_del_script.py
  ```

  - El comando permite personalizar el tipo de instalador que va a ser creado mediante numerosas opciones, en este caso se utilizó `--onefile`; esta opción genera un único archivo ejecutable.

- **Archivos generados**: Tras ejecutar este comando, PyInstaller crea una carpeta `dist` que contiene el archivo ejecutable, y una carpeta `build` con archivos temporales utilizados durante el proceso de creación. El ejecutable en la carpeta `dist` puede distribuirse directamente a los usuarios. Los resultados completos para la aplicación que utiliza RPDPM como motor y para la que usa el algoritmo basado en Leaspy se encuentran en los directorios *demo/Instaladores/GUI_RPDPM* y *demo/Instaladores/GUI_Leaspy*, respectivamente.

### 4.8.2. Programa en MATLAB

Para el programa desarrollado en MATLAB, se utilizó MATLAB Compiler, una herramienta que permite empaquetar aplicaciones de MATLAB para su distribución. Esta opción resultó de gran utilidad debido a su sencillez y facilidad para crear instaladores y ejecutables para múltiples sistemas operativos. Aunque se han generado

instaladores tanto para Windows como para Linux, a continuación se muestra únicamente el procedimiento para Windows. En Linux se deben seguir exactamente los mismos pasos, pero MATLAB Compiler debe ejecutarse desde un sistema operativo basado en Linux.

- **Preparación del archivo**: El programa de MATLAB está desarrollado mediante MATLAB App Designer, por lo que fue necesario asegurar que el archivo principal `.mlapp` estuviera listo para ser compilado, incluyendo el motor `visualize_rpdpm.m` y todas las dependencias necesarias: las carpetas *source* y *data*. Es importante tener una estructura del programa bien organizada en el caso de que el script principal necesite acceder a archivos externos. A continuación, se muestra la estructura de este programa:

```
GUI_MATLAB
|   GUI_MATLAB.mlapp
|
+---data
|       percentil_LACT.mat
|       workspace_demo_rpdpm_bstrp_10.mat
|
\---source
    |   visualize_rpdpm.m
    |
    \---rpdpm
        buildTableSample.m
        get_test_tray_RPDPM.m
        get_timeline_tray.m
        multiclass_auc.m
        objective_subject.m
        prob_RPDPM.m
        rpdpm_test_tray.m
        sigmoid_function.m
```

  Además, para asegurar que las rutas de estos archivos se manejen correctamente independientemente del equipo en el que se ejecute el programa, la carga de datos en `visualize_rpdpm.m` se realiza de la siguiente forma:

**Algorithm 6** Carga de datos en `visualize_rpdpm.m`

```
%% Input data file
currentFolder = fileparts(mfilename('fullpath'));
addpath(fullfile(currentFolder, 'rpdpm'));
load(fullfile(currentFolder, '../data/percentil_LACT.mat'), 'percent',
'point_feat');
load(fullfile(currentFolder, '../data/workspace_demo_rpdpm_bstrp_10.mat'),
'Sigma', 'A', 'B', 'C', 'D', 'G', 'posterior_bayes');
```

- **Uso de MATLAB Compiler**: Para abrir la herramienta, se puede ejecutar el siguiente comando en la línea de comandos de MATLAB:

```
deploytool
```

A partir de aquí, se deben seguir los siguientes pasos para obtener el instalador:

- **Nuevo proyecto**: Se crea un nuevo proyecto seleccionando 'Application Compiler'.

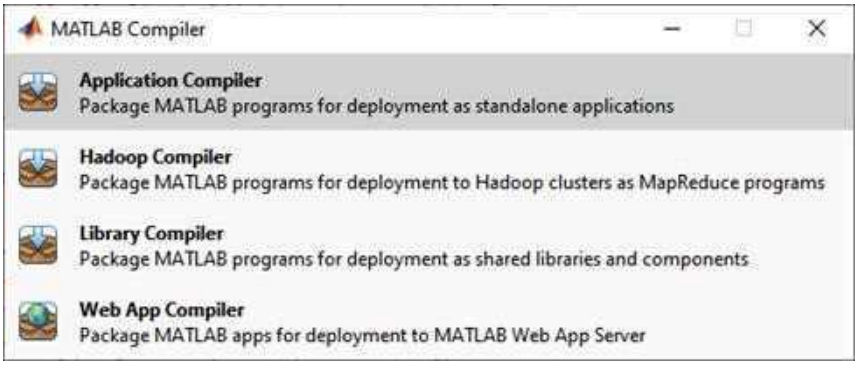


Figura 4.12: Creación de un nuevo proyecto de MATLAB Compiler.

- **Selección del script principal**: Se añade el script principal del programa. La opción 'Standalone Application' debe estar seleccionada.

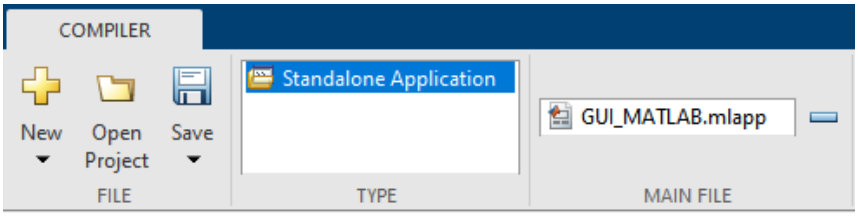


Figura 4.13: Selección del archivo principal del programa que se desea compilar con MATLAB Compiler.

- **Configuración de parámetros**: Se configuran parámetros adicionales, como la inclusión de archivos adicionales y opciones de configuración. Para el caso concreto de este programa, todas las dependencias están incluidas en las carpetas *source* y *data*, por lo que se deben añadir en la sección 'Files required for your application to run'.


Figura 4.14: Sección de MATLAB Compiler para añadir las dependencias del programa.

- **Compilación**: Por último, se debe decidir entre incluir el Runtime de MATLAB necesario para la ejecución del proyecto o mantenerlo alojado en la web, lo cual hace que los archivos resultantes sean mucho más ligeros. Con el objetivo de favorecer la distribución del programa, en este caso se ha elegido incluirlo. Una vez seleccionada una de estas dos opciones, se debe hacer clic en el botón 'Package' para iniciar el proceso de compilación.


Figura 4.15: Barra superior de MATLAB Compiler con el botón 'Package'.

- **Archivos generados**: MATLAB Compiler genera un instalador que incluye el ejecutable y todas las dependencias necesarias, así como un archivo de instalación para el MATLAB Runtime, que los usuarios deben instalar si no tienen MATLAB instalado en su sistema. El resultado se puede consultar en el directorio *demo/Instaladores/GUI_MATLAB*.

# Capítulo 5

# Resultados

## 5.1. Introducción

La aplicación final de este proyecto consta con una interfaz intuitiva y fácil de entender para el usuario. Esta GUI permite una introducción y manejo de los datos muy completa con varias funcionalidades que favorecen esto. Posteriormente, se exponen los resultados de las predicciones de forma clara sin que el usuario sea consciente de que se está ejecutando un complejo algoritmo en segundo plano. Además de esta aplicación, se han creado dos variantes: una que funciona con un motor distinto para la obtención de los resultados y otra que conserva el algoritmo original pero cuenta con una adaptación de la interfaz en otro lenguaje de programación. A pesar de esto, solo se va a utilizar la aplicación principal para explicar el funcionamiento, ya que la primera variación conserva la misma GUI y su funcionamiento es exactamente el mismo y la segunda se trata de una adaptación que se ha realizado tratando de parecerse lo máximo posible a la original. Los pequeños matices en los que se diferencia respecto de la original, serán indicados igualmente en su sección correspondiente.

Al iniciar el programa, el aspecto de la GUI es el siguiente:

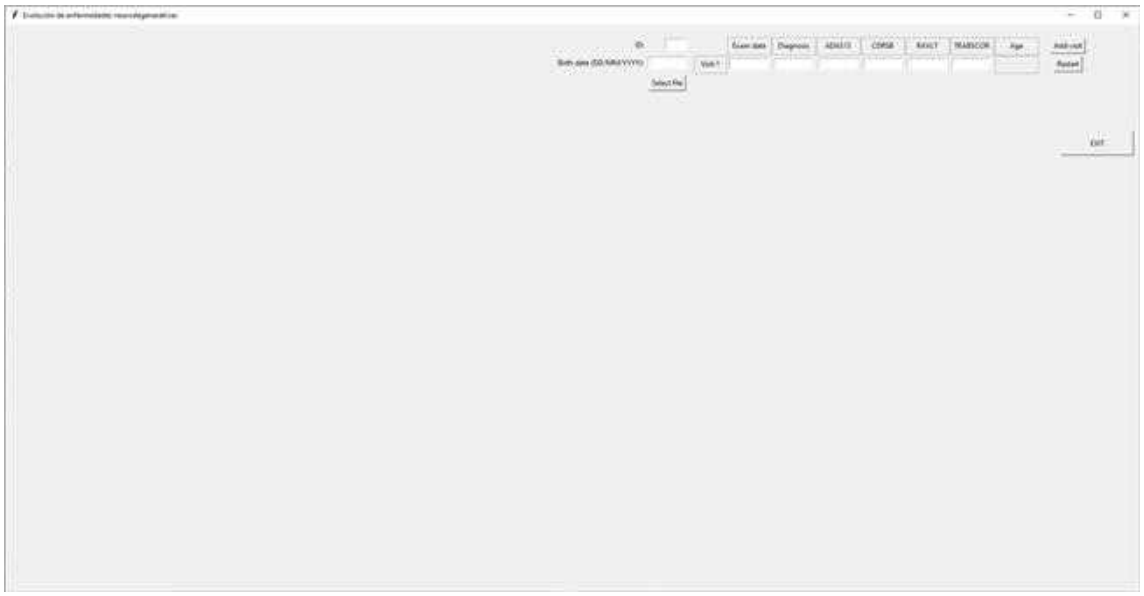

Figura 5.1: Apariencia de la interfaz tras iniciar la aplicación.

Se aprecia arriba a la derecha la sección para la introducción de datos con una serie de botones para interactuar con esta y un botón 'EXIT' para finalizar la ejecución del programa. Por otro lado, se observa un gran espacio vacío a la izquierda y abajo a la derecha de la interfaz: estos espacios existen para mostrar los resultados de las predicciones y una presentación de los datos del paciente que están listos para ser enviados al algoritmo, respectivamente. El funcionamiento de estas secciones y todas sus funcionalidades van a ser explicados a lo largo de este capítulo, que se ha dividido en *Introducción de los datos* y *Presentación de los resultados.*

Tras avanzar por todas las fases de la aplicación que se explican en dichas secciones, el aspecto final de la GUI se observa en el siguiente ejemplo:

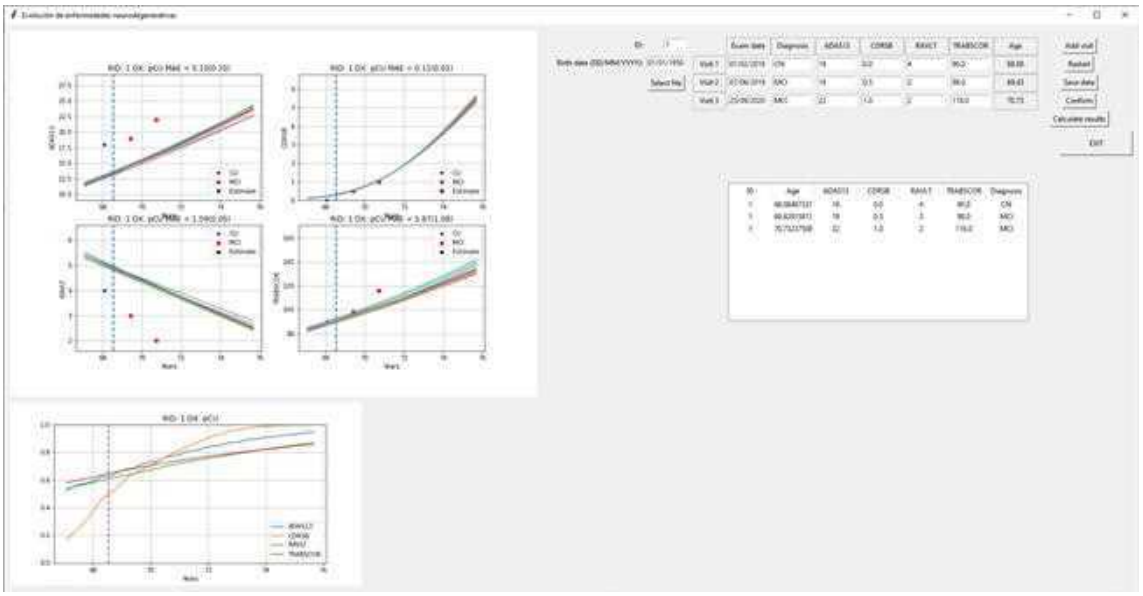

Figura 5.2: Apariencia final de la interfaz mostrando los resultados de un paciente.

## 5.2. Introducción de los datos

La sección más importante de la interfaz se trata de la destinada a la introducción de datos. Su fin consiste en obtener los datos de un paciente para tratarlos internamente y, posteriormente, enviarlos al algoritmo para obtener los resultados de sus predicciones. Sin embargo, una introducción de datos intuitiva y amigable para el usuario resulta fundamental para cumplir los objetivos de este proyecto, por lo que se han añadido todas las funcionalidades que se han considerado necesarias para lograr esto.

Tras ejecutar la aplicación, el estado inicial de la sección es el siguiente:


Figura 5.3: Estado inicial de la sección de introducción de datos.

Como se puede observar, aparecen dos entradas de texto a la izquierda llamadas 'ID' y 'Birth date (DD/MM/YYYY)', estas sirven para introducir el ID con el que se quiere identificar al sujeto y su fecha de nacimiento. La entrada de la fecha de nacimiento además indica el formato con el que debe ser introducida. Por otro lado, a su derecha se aprecian seis columnas con sus entradas correspondientes y una séptima columna no modificable. Como se muestra en el título de cada columna, los datos que deben ser introducidos en cada una son, de izquierda a derecha:

- La fecha de la visita del sujeto.

- Su diagnóstico clínico.
- La puntuación del biomarcador 'ADAS13' obtenida en esa visita.
- La puntuación del biomarcador 'CDRSB' obtenida en esa visita.
- La puntuación del biomarcador 'RAVLT' obtenida en esa visita.
- La puntuación del biomarcador 'TRABSCOR' obtenida en esa visita.

Además, la séptima columna 'Age' se actualizará automáticamente con la edad del sujeto en cada visita para facilitar la comprensión de los resultados al compararlos con los datos introducidos. Estas últimas, a diferencia de las entradas 'ID' y 'Birth date', se encuentran dispuestas de este modo porque varían para cada visita que haga el paciente.

Por otro lado, en el estado inicial se aprecian tres botones asociados a la tabla. El botón 'Add visit' se utiliza para agregar filas a la tabla, simbolizando estas una nueva visita del mismo paciente. Gráficamente, su efecto es el siguiente:


Figura 5.4: Funcionamiento del botón 'Add visit'.

Una de las razones para dejar tanto espacio libre debajo de la sección de introducción de datos es que no haya ningún problema a la hora de añadir visitas, ya que el usuario puede agregar las que considere sin límite. Si el usuario comete errores al ingresar los datos, puede utilizar el botón 'Restart' para comenzar de nuevo el proceso. Este botón devuelve la sección a su estado inicial, borrando cualquier valor que se encuentre en las entradas y eliminando todas las filas menos la primera.

El último botón que se observa, 'Select file', sirve para facilitar al usuario en gran medida la introducción de datos. Al interactuar con él, se abre una ventana que permite inspeccionar el equipo en busca de archivos Excel con un formato sencillo para completar la tabla con los datos que contenga. A continuación, se muestra un ejemplo del formato que deben tener los archivos Excel:

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | ID | Birth date | Exam date | Diagnosis | ADAS13 | CDRSB | RAVLT | TRABSCOR |
| 2 | 1 | 01/01/1950 | 01/02/2018 | CN | 18 | 0.0 | 4 | 90.0 |
| 3 | 1 | 01/01/1950 | 07/06/2019 | MCI | 19 | 0.5 | 3 | 98.0 |
| 4 | 1 | 01/01/1950 | 25/09/2020 | MCI | 22 | 1.0 | 2 | 116.0 |

Figura 5.5: Ejemplo de archivo Excel con la información de un sujeto.

Esta captura de pantalla corresponde a la información de un paciente introducida en un archivo Excel, se observa que este debe incluir una fila con los nombres de

cada columna seguida de una fila por cada visita del sujeto con los valores exactos que van a ser escritos en las entradas de la tabla. Este archivo puede ser obtenido completándolo de forma manual o mediante una funcionalidad que será presentada posteriormente. Se puede consultar este ejemplo en concreto en la dirección *demo/Anexo/paciente_ejemplo.xlsx*.

Una vez se termine de introducir la información del paciente, ya sea de forma manual o mediante un archivo Excel, el siguiente paso consiste en confirmar que se quieren utilizar los datos de la tabla para obtener las predicciones mediante el botón 'Confirm'. Sin embargo, para que este botón se muestre en la interfaz, los datos de la tabla deben cumplir con una serie de requisitos:

1. Las entradas 'ID' y 'Birth date' no pueden estar vacías y su formato debe ser correcto: el ID debe ser siempre numérico y la fecha de nacimiento debe seguir el formato 'Día/Mes/Año' solicitado.
2. Todas las entradas de la columna 'Exam date' (tantas como visitas indique el usuario), al igual que las dos anteriores, deben estar completas y con el mismo formato de fecha 'Día/Mes/Año'.
3. El resto de entradas pueden estar vacías ya que se considerarían como 'NaN' (*Not a Number*); sin embargo, si contienen información, esta debe tener el formato correcto: las entradas de la columna 'Diagnóstico' deben ser cadenas de texto y las demás, valores numéricos.

Una vez que los datos introducidos cumplen los requisitos anteriores, se muestran los botones 'Confirm' y 'Save data'. A continuación, se presenta un ejemplo de un conjunto de datos con un formato correcto que ocasiona la aparición de estos dos botones:

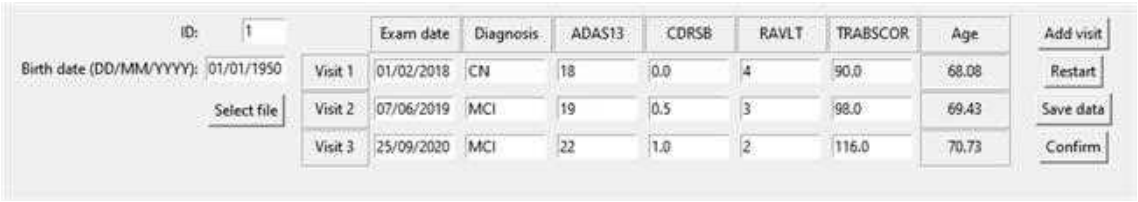


Figura 5.6: Ejemplo de un conjunto de datos con un formato correcto.

El nuevo botón 'Save data', similar al botón 'Select file', permite al usuario guardar un archivo Excel con los datos introducidos en la ubicación de su equipo que él seleccione. Este botón solo aparece cuando los datos tienen un formato correcto para asegurar que el Excel generado también lo tenga. Esto permite que, al utilizar los botones 'Select file' y 'Save data', el usuario pueda realizar tareas como cargar la información guardada de un paciente, editarla manualmente a través de la interfaz y guardarla de nuevo siempre y cuando el formato siga siendo correcto tras realizar los cambios a la información.

## 5.3. Presentación de los resultados

Una vez que los datos ingresados por el usuario tienen un formato correcto, aparece el botón 'Confirm'. Al hacer clic en este botón, se cargan internamente los

datos, listos para ser enviados al algoritmo, y se presentan en una tabla ubicada en la parte inferior derecha de la interfaz. A continuación, se presenta un ejemplo:

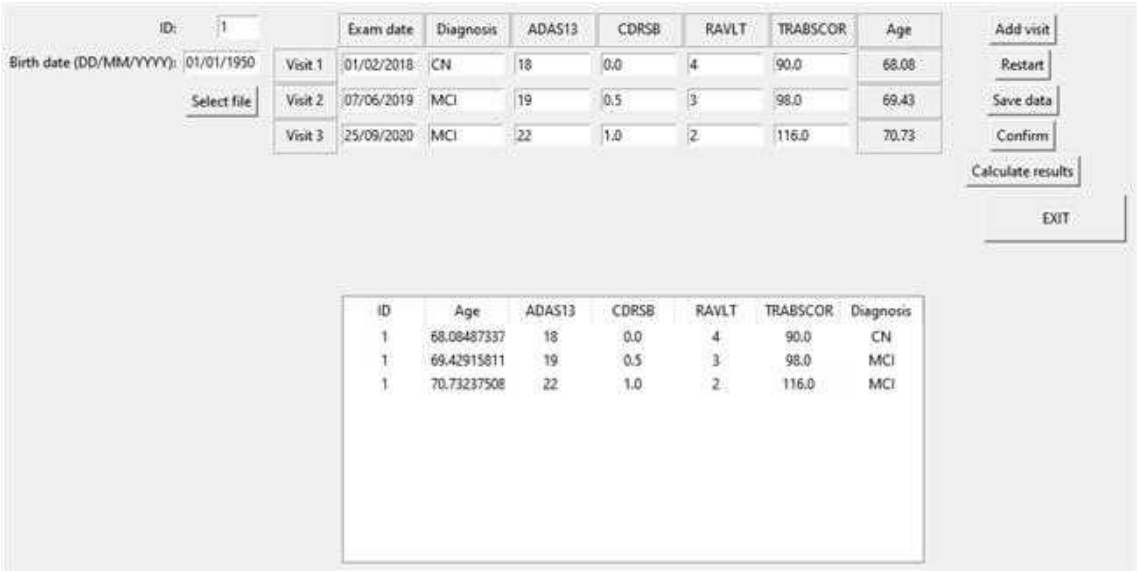


Figura 5.7: Presentación de los datos confirmados mediante una tabla.

En la captura de pantalla se observa cómo se presenta la tabla mientras sigue siendo posible realizar cualquier gestión con los datos ingresados. Sin embargo, aunque se realicen ediciones en la sección de introducción de datos, la única manera de actualizar la tabla y, por ende, los datos preparados para ser enviados al algoritmo, es haciendo clic nuevamente en el botón 'Confirm'. De esta forma, se minimizan las posibilidades de que el usuario pueda encontrarse con errores complejos como los relacionados con el algoritmo, ya que el programa se asegura internamente de que el formato de los datos es adecuado antes de permitirle al usuario utilizar el algoritmo.

En la imagen anterior, también se puede notar que al hacer clic en 'Confirm', aparece un nuevo botón llamado 'Calculate results'. Como su nombre indica, este botón se encarga de llamar a todas las funciones relacionadas con la interacción con el algoritmo para enviar los datos cargados que se muestran en la tabla. A continuación, se muestra un ejemplo de cómo se presentan los resultados del algoritmo:

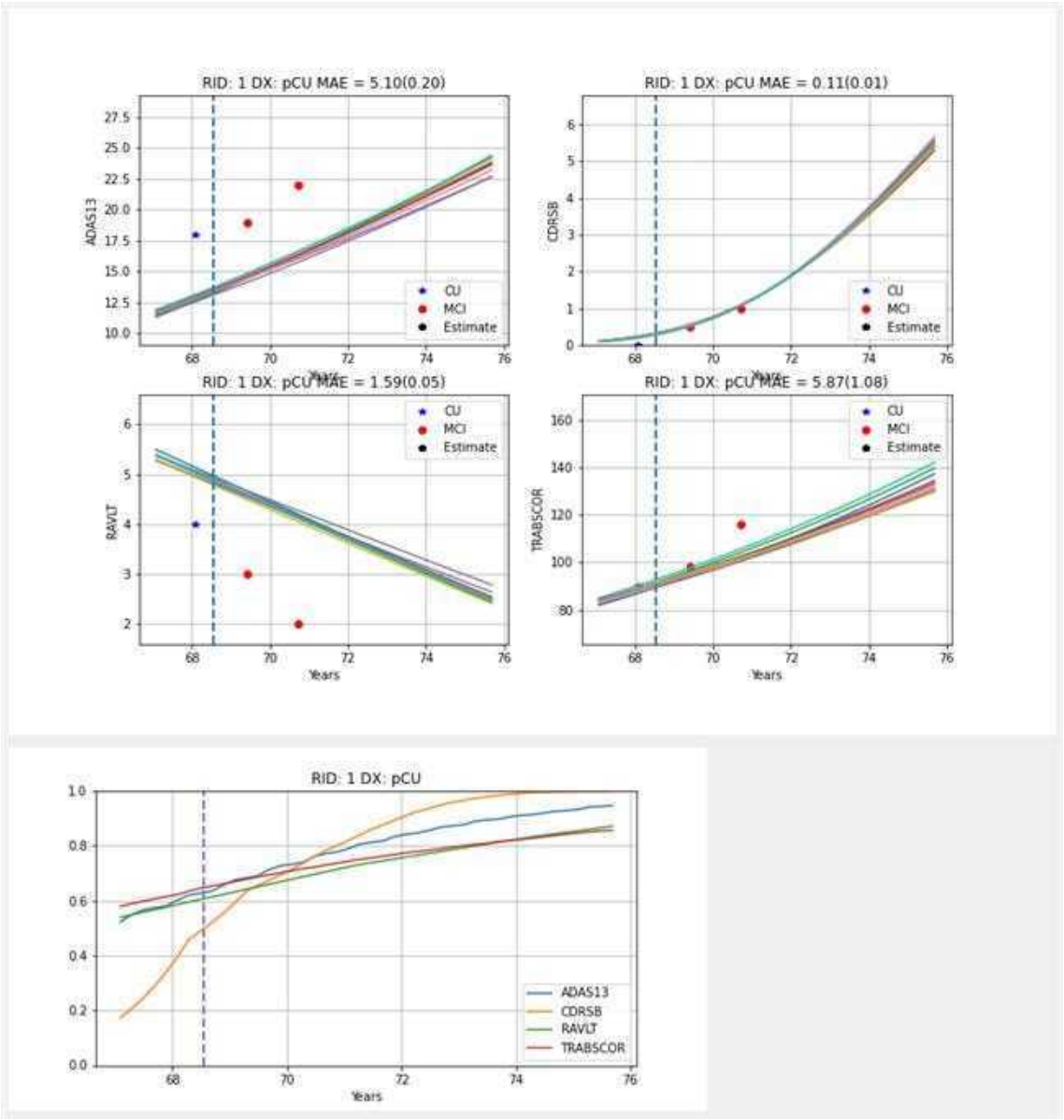


Figura 5.8: Presentación de los resultados del algoritmo.

Para mostrar estos resultados, se hace uso del espacio vacío que había inicialmente en la parte izquierda de la interfaz. En la sección *Visualización de las trayectorias* del capítulo de métodos se explica detalladamente la programación detrás de la obtención de estas trayectorias. A continuación, se explica cada componente de los gráficos:

En primer lugar, se presentan dos figuras. La figura superior está compuesta por cuatro gráficos correspondientes a los cuatro biomarcadores empleados en este estudio: 'ADAS13', 'CDRSB', 'RAVLT' y 'TRABSCOR'. Cada gráfico muestra el diagnóstico calculado por el biomarcador y el 'MAE' (Mean absolute error) obtenido. Por otro lado, en el gráfico se pueden apreciar las puntuaciones que el sujeto obtuvo en cada visita en función de su edad en ese momento, junto con las trayectorias estimadas que se espera que sigan a lo largo del tiempo.

En cuanto a la figura inferior, reúne las trayectorias de los cuatro biomarcadores mostrados en los gráficos de la figura superior y los presenta de forma normalizada para proporcionar una perspectiva general de las predicciones al usuario, utilizando todos los biomarcadores del estudio en conjunto. Además, se muestra el diagnóstico calculado con los resultados de los cuatro marcadores.

## 5.4. Ejecutables e instaladores

En el capítulo anterior se explica el procedimiento realizado para obtener los ejecutables e instaladores necesarios para la distribución de las aplicaciones. Sin embargo, aunque se han seguido todos los pasos aprendidos durante la documentación del proyecto de forma detallada, los resultados de esta parte del trabajo no han sido satisfactorios: mientras que los ejecutables e instaladores de la variante de MATLAB funcionan a la perfección, los correspondientes a las aplicaciones basadas en Python presentan problemas.

Aunque los instaladores y ejecutables se crean satisfactoriamente al seguir los pasos para todos los programas, tras ejecutarlos aparecen errores desconocidos que no se presentan al ejecutar los programas desde los entornos de desarrollo.

A pesar de estos inconvenientes, los resultados se pueden consultar igualmente en la carpeta *demo/Instaladores*, y en el siguiente capítulo se señalan estos errores y se discuten distintas soluciones posibles. Como los resultados para la versión de MATLAB sí han sido satisfactorios, se ha incluido una guía de instalación completa en el capítulo *Anexo*, tanto para Windows como para Linux.

# Capítulo 6

# Discusiones

## 6.1. Introducción

En este último capítulo, se lleva a cabo una reflexión sobre los resultados expuestos durante el capítulo anterior. Para ello, se analiza si los resultados satisfacen los objetivos establecidos inicialmente y se comparan con las alternativas más relevantes del mercado. Además, se dedica una sección a señalar posibles mejoras del producto y líneas de investigación o desarrollo futuras.

Para cumplir con los objetivos establecidos, se ha desarrollado principalmente una aplicación en Python. Como se señala a lo largo del capítulo anterior, todas las decisiones de diseño se han tomado con el objetivo de cumplir estos criterios: proporcionar una interfaz simple, un manejo intuitivo para usuarios sin experiencia en informática y un control de errores robusto, lo que facilita la experiencia del usuario y minimiza la preocupación por aspectos técnicos. Además del diseño, los objetivos de hacer que la aplicación sea multiplataforma y que el motor esté poco acoplado al resto del programa han sido fundamentales en todo momento.

Para demostrar y probar esto, se han desarrollado las dos versiones alternativas del programa que se detallan en los capítulos anteriores. Durante este capítulo, se discuten las ventajas e inconvenientes de los distintos entornos en los que se ha trabajado y de los diferentes motores que se han empleado para la obtención de las predicciones.

Por estas razones, se considera que el proyecto cumple satisfactoriamente con los objetivos iniciales. No obstante, a continuación se lleva a cabo una reflexión sobre las ventajas e inconvenientes que ofrece cada alternativa de diseño tratada durante la realización del proyecto. Además, más adelante en el capítulo se analizan las ventajas reales que ofrece este producto frente a los más relevantes del mercado y se hace una crítica sobre los puntos más débiles de las aplicaciones finales, proponiendo posibles líneas de investigación futuras en torno a estos aspectos.

## 6.2. Diferencias entre Python y MATLAB

En cuanto al lenguaje de programación utilizado, se ha invertido mucho tiempo tanto en Python como en MATLAB. Además de adquirir experiencia trabajando

estrechamente con ambos lenguajes, se ha realizado una investigación para poder discutir con mayor precisión qué entorno resulta más adecuado para proyectos de este ámbito. Ambos lenguajes son muy interesantes y ampliamente utilizados para el desarrollo de GUIs.

En el caso de Python, como se explica en el artículo de Özgür et al. (2017) [32], se trata de un lenguaje más compacto y mucho más fácil de leer que MATLAB, lo que supone una gran ventaja a la hora de escalar proyectos debido a la facilidad de depuración. Además, es un software gratuito y de código abierto, lo que favorece su accesibilidad para futuras líneas de desarrollo. Python destaca por su amplio catálogo de herramientas y paquetes, siendo superior a MATLAB en cuanto a opciones gráficas, lo cual resulta especialmente interesante para este proyecto.

Por otro lado, MATLAB presenta una ventaja en cuanto al desempeño de la aplicación, ya que su optimizado motor de computación numérica resulta muy eficiente para tareas como la ejecución de algoritmos utilizados en las aplicaciones de este proyecto [40]. Aunque ser un software de pago puede suponer desventajas, incluye herramientas que facilitan significativamente las tareas del proyecto, como MATLAB App Designer o MATLAB App Installer, que permiten crear instaladores de manera más sencilla que las opciones disponibles en Python. Además, la cantidad de información y documentación disponible sobre cualquier función o paquete en MATLAB es abundante y útil para el desarrollo.

En conclusión, aunque MATLAB ofrece una serie de ventajas muy interesantes, para este proyecto se considera que Python es un lenguaje con mejores fortalezas a la hora de cumplir con los objetivos establecidos.

## 6.3. Diferencias entre Windows y Linux

Para cumplir con el objetivo de que la aplicación sea multiplataforma, ha sido necesario experimentar con distintos sistemas operativos y asegurarse de que las herramientas utilizadas faciliten la compatibilidad entre varias plataformas. Aunque existen otros entornos muy utilizados hoy en día, Windows sigue siendo el principal dominante en cuanto a sistemas operativos para dispositivos de escritorio [42], por lo que se eligió como la plataforma principal para esta aplicación. Por otro lado, los servidores de Curie, utilizados para alojar este proyecto, están basados en Linux, lo que hizo que se priorizara este sistema operativo frente a otros más utilizados en la actualidad como macOS. A continuación, se destacan las ventajas de desarrollar una aplicación en cada uno de estos entornos:

Windows ofrece una amplia compatibilidad con software y hardware, facilitando un entorno de desarrollo estable para Python y MATLAB. Herramientas como Spyder y MATLAB están optimizadas para Windows, proporcionando un flujo de trabajo eficiente. La facilidad de distribución de aplicaciones es otra ventaja, con herramientas como Inno Setup y Windows Installer que permiten empaquetar y distribuir aplicaciones de manera profesional. Además, la amplia base de usuarios de Windows asegura que las aplicaciones sean accesibles para un mayor número de personas sin configuraciones adicionales.

Por otro lado, Linux ofrece un entorno altamente flexible y personalizable, ideal para optimizar el desarrollo de aplicaciones en Python y MATLAB. Su estabilidad y eficiencia en la gestión de recursos son beneficiosas para ejecutar algoritmos complejos. La naturaleza de código abierto de Linux proporciona acceso a numerosas bibliotecas y herramientas gratuitas, facilitando soluciones personalizadas e integraciones de terceros. Además, la activa comunidad de desarrolladores de Linux ofrece soporte y documentación abundantes, lo cual es invaluable para resolver problemas técnicos y mejorar el desarrollo de la aplicación.

El desarrollo de las aplicaciones de este proyecto se ha realizado mayoritariamente en el entorno de Windows debido a la familiaridad con las herramientas de este sistema. Como se ha tratado de utilizar alternativas que favorezcan la portabilidad a otros entornos, realizar una versión para Linux de la aplicación principal fue sumamente sencillo. A pesar de ello, el algoritmo RPDPM portado de la versión de Windows de MATLAB al entorno de Linux no funciona como se esperaba, por lo que más adelante se propone una mejora de este aspecto como una posible línea de investigación futura.

## 6.4. Diferencias entre RPDPM y el algoritmo desarrollado en Leaspy

Desde un primer momento, se ha tratado de implementar el motor de la forma más desacoplada posible para facilitar su sustitución o actualización, lo que mejora la escalabilidad de la aplicación y sus desarrollos futuros. Para probar esto, se realizó una versión alternativa del programa principal que conserva en su totalidad la interfaz y todos los elementos ajenos al motor. Los dos motores elegidos para cubrir estas dos alternativas son RPDPM y un algoritmo basado en Leaspy, ya que se tratan de dos enfoques prominentes en el ámbito del modelado de la progresión de enfermedades. A continuación, se discuten las principales diferencias entre estos algoritmos en términos de su metodología, aplicación y eficacia.

En cuanto a la metodología, Leaspy se basa en un enfoque probabilístico que utiliza el modelo de Markov y técnicas de optimización estocástica para estimar la progresión de la enfermedad. Modela trayectorias individuales de pacientes utilizando una combinación de efectos fijos y aleatorios para representar tanto la progresión promedio de la enfermedad en la población como las variaciones individuales. Los parámetros individuales incluyen factores de aceleración y desplazamiento temporal, que permiten ajustar las trayectorias individuales en relación con la trayectoria promedio de la enfermedad.

Por otro lado, RPDPM emplea una función logística parametrizada y estimadores robustos para modelar la progresión de la enfermedad. Este algoritmo se enfoca en minimizar el error de modelado al tiempo que se mantiene robusto frente a los valores atípicos. Utiliza una técnica de optimización iterativa para ajustar los parámetros del modelo de forma que se alinee con los datos observados de biomarcadores.

En Leaspy, la estimación de parámetros se realiza mediante el algoritmo de Stochastic Approximation Expectation Maximization (SAEM), que es una variante estocástica del algoritmo de Expectation-Maximization (EM). Este enfoque permite manejar la incerteza y variabilidad inherentes a los datos longitudinales al iterar entre la simulación de variables latentes y la maximización de la función de verosimilitud.

RPDPM, en cambio, utiliza un enfoque de estimación iterativa que alterna entre la estimación de parámetros biomarcadores y parámetros específicos del sujeto. Esta técnica, conocida como optimización alternada, facilita la convergencia hacia valores óptimos al iterar entre ajustes parciales de los parámetros hasta alcanzar la convergencia global.

Leaspy es robusto en términos de manejar datos longitudinales complejos y heterogéneos, proporcionando una representación detallada de las trayectorias de la enfermedad tanto a nivel poblacional como individual. Su capacidad para personalizar las trayectorias basándose en parámetros específicos del sujeto lo hace flexible y adaptable a diferentes contextos clínicos.

Por su parte, RPDPM se destaca por su robustez frente a valores atípicos, una característica crucial cuando se trabaja con datos biomarcadores que pueden presentar variaciones significativas. El uso de funciones logísticas parametrizadas permite una modelización precisa de la progresión de la enfermedad, y su enfoque en la minimización de errores mediante M-estimadores asegura una alta estabilidad y precisión en las estimaciones.

Respecto a los resultados obtenidos por cada algoritmo, Leaspy ha sido aplicado exitosamente en el modelado de la progresión de la enfermedad de Alzheimer, demostrando su capacidad para capturar variaciones individuales y proporcionar predicciones precisas a largo plazo. Su enfoque en la calibración y personalización de parámetros lo hace particularmente útil en estudios longitudinales donde la variabilidad intersujetos es significativa.

Por otro lado, RPDPM también ha mostrado eficacia en el modelado de la progresión de la enfermedad de Alzheimer, especialmente en el análisis de datos de biomarcadores. Su capacidad para manejar datos ruidosos y valores atípicos lo hace ideal para aplicaciones donde la precisión y la robustez son críticas. Los estudios han demostrado que RPDPM puede proporcionar modelos de progresión fiables y útiles para la toma de decisiones clínicas.

En resumen, tanto Leaspy como RPDPM son algoritmos potentes para el modelado de la progresión de enfermedades, cada uno con sus propias fortalezas y aplicaciones ideales. Leaspy sobresale en la personalización y adaptación a datos longitudinales, mientras que RPDPM se distingue por su robustez frente a valores atípicos y su precisión en la estimación de parámetros biomarcadores. La elección entre estos algoritmos dependerá del contexto específico del estudio y de los objetivos de modelado. Debido a que ambas alternativas se consideran muy competitivas en la actualidad y que, al mismo tiempo, presentan diferencias que las hacen destacar por

distintas razones, se decidió utilizar estas dos opciones para demostrar la facilidad con la que se puede implementar un nuevo motor en el programa.

## 6.5. Comparación con las principales alternativas del mercado

Para evaluar la efectividad de la aplicación desarrollada, durante esta sección se compara con las principales alternativas disponibles en el mercado y las funcionalidades que ofrecen. A continuación, se explica brevemente en qué consisten las herramientas seleccionadas y, posteriormente, se discuten las diferencias que muestran respecto a la aplicación desarrollada durante este proyecto.

### 6.5.1. Cognivue [14]

Cognivue es una herramienta comercial líder en evaluación cognitiva que se destaca por su enfoque innovador y su capacidad para detectar y monitorear cambios en la función cognitiva de los pacientes. Esta herramienta se utiliza principalmente en entornos clínicos y de atención médica para evaluar la función cerebral y detectar posibles signos tempranos de deterioro cognitivo, como los asociados con enfermedades neurodegenerativas como el Alzheimer.

El funcionamiento de Cognivue se basa en la realización de pruebas cognitivas automatizadas, diseñadas para evaluar una amplia gama de funciones cognitivas, incluida la memoria, la atención, la velocidad de procesamiento y la función ejecutiva. Estas pruebas se presentan de manera interactiva a los pacientes a través de una interfaz intuitiva y amigable, que permite una fácil comprensión y participación.

Una de las principales utilidades de Cognivue radica en su capacidad para proporcionar mediciones objetivas y cuantificables de la función cognitiva de un individuo, lo que ayuda a los profesionales de la salud a realizar evaluaciones más precisas y a monitorear la progresión de condiciones cognitivas a lo largo del tiempo. Esto puede ser especialmente útil en la detección temprana de enfermedades neurodegenerativas, permitiendo intervenciones más tempranas y efectivas.

Además, Cognivue ofrece características avanzadas de análisis de datos y generación de informes, lo que facilita a los profesionales de la salud la interpretación de los resultados y la comunicación efectiva con los pacientes y otros miembros del equipo de atención médica. Esto contribuye a una atención más integral y personalizada para aquellos que pueden estar experimentando cambios en su función cognitiva.

Aunque Cognivue es una herramienta muy completa, se centra principalmente en la detección temprana de enfermedades neurodegenerativas. A diferencia de esta, la aplicación desarrollada durante este proyecto tiene como objetivo obtener la progresión de la enfermedad a lo largo de una gran cantidad de tiempo, sin limitarse a realizar detecciones tempranas. Por otro lado, la segunda diferencia significativa consiste en el tipo de pruebas realizadas: mientras que Cognivue funciona con pruebas automatizadas, esta aplicación utiliza biomarcadores de forma general. Aunque

la opción de Cognivue puede resultar más cómoda para el usuario, el uso de biomarcadores brinda a la aplicación una gran capacidad de actualización para adaptarse a una industria que avanza a gran velocidad.


Figura 6.1: Herramienta de Cognivue utilizada para hacer sus pruebas.

### 6.5.2. Cogstate

Cogstate es otra herramienta prominente en el ámbito de la evaluación cognitiva, utilizada tanto en investigaciones clínicas como en el cuidado de la salud. Al igual que Cognivue, Cogstate se especializa en detectar y monitorear cambios en la función cognitiva, proporcionando pruebas rápidas y automatizadas que evalúan diversas capacidades cognitivas, como la memoria, la atención, la velocidad de procesamiento y la función ejecutiva [15].

Cogstate emplea una batería de pruebas cognitivas informatizadas que son presentadas a través de una interfaz sencilla y accesible. Estas pruebas están diseñadas para ser administradas fácilmente y para proporcionar resultados inmediatos y objetivos. Una de las principales ventajas de Cogstate es su validación científica y su uso extendido en estudios de investigación, lo que respalda su fiabilidad y precisión en la evaluación cognitiva.

A diferencia de la aplicación desarrollada en este proyecto, que utiliza biomarcadores para evaluar la progresión de la enfermedad a lo largo del tiempo, Cogstate se enfoca en proporcionar una evaluación cognitiva rápida y precisa. Esta diferencia en el enfoque permite que Cogstate sea especialmente útil en situaciones donde se requiere una evaluación cognitiva inmediata y repetida, como en estudios clínicos o en evaluaciones periódicas de pacientes.

Otra diferencia clave radica en el ámbito de uso. Mientras que Cogstate se utiliza principalmente en entornos clínicos y de investigación, la aplicación desarrollada durante este proyecto está diseñada para un uso más generalizado, permitiendo el seguimiento a largo plazo de la progresión de la enfermedad en diversos contextos. Además, la utilización de biomarcadores en la aplicación del proyecto ofrece una perspectiva más completa y adaptable a los avances tecnológicos y científicos en el campo de la evaluación cognitiva.

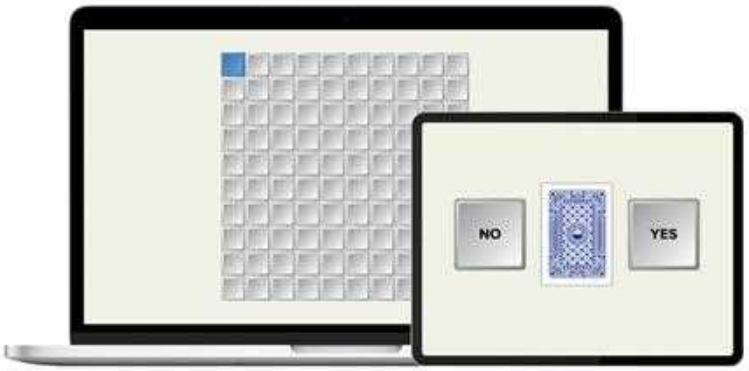


Figura 6.2: Ejemplo de pruebas de Cogstate.

### 6.5.3. Cambridge Cognition (CANTAB)

Cambridge Cognition, con su batería de pruebas cognitivas CANTAB (Cambridge Neuropsychological Test Automated Battery), es una herramienta de evaluación cognitiva ampliamente reconocida en entornos clínicos y de investigación. CANTAB se distingue por su uso de pruebas neuropsicológicas informatizadas que evalúan funciones cognitivas específicas mediante tareas diseñadas con base en investigaciones neurocientíficas [10].

Las pruebas de CANTAB son altamente estandarizadas y han sido validadas a través de numerosos estudios científicos, lo que las hace extremadamente fiables para la evaluación de la función cognitiva. Estas pruebas abarcan diversas áreas cognitivas, incluyendo la memoria, la atención, la velocidad de procesamiento, y la función ejecutiva. Una característica notable de CANTAB es su capacidad para identificar sutiles cambios cognitivos que pueden no ser detectados por otras herra-

mientas.

En comparación con la aplicación desarrollada en este proyecto, CANTAB se centra en la precisión y la validación científica de sus pruebas, lo que la hace ideal para estudios clínicos y de investigación. Sin embargo, la aplicación del proyecto ofrece una ventaja significativa en su capacidad para integrar biomarcadores, proporcionando una visión más holística de la progresión de la enfermedad. Además, mientras que CANTAB requiere un entorno clínico para su administración, la aplicación desarrollada está diseñada para ser más accesible y adaptable a diferentes contextos, facilitando el seguimiento a largo plazo de los pacientes.

Otra diferencia importante es el enfoque de uso. CANTAB está específicamente orientada hacia la investigación y la evaluación clínica detallada, mientras que la aplicación del proyecto busca combinar la evaluación cognitiva con la integración de datos biomarcadores, adaptándose a los avances tecnológicos y proporcionando una herramienta versátil y actualizada para el seguimiento de enfermedades neurodegenerativas.

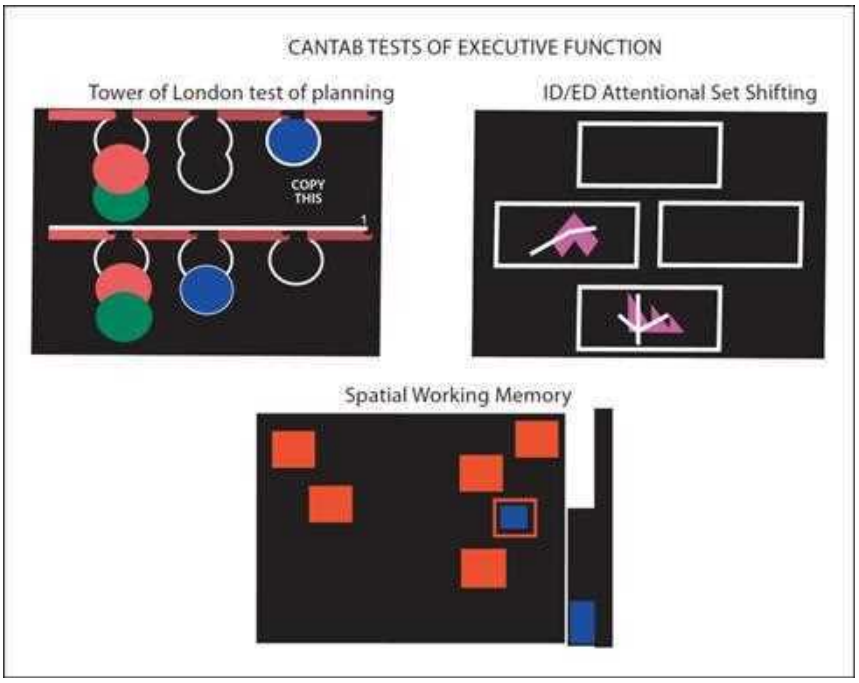


Figura 6.3: Ejemplo de tres pruebas de CANTAB. Fuente: [26]

### 6.5.4. Comparativa General

En resumen, Cognivue, Cogstate y Cambridge Cognition (CANTAB) representan algunas de las principales herramientas en el mercado para la evaluación cognitiva. Cada una de estas herramientas tiene sus propias fortalezas y áreas de especialización. Cognivue destaca por su enfoque en la detección temprana de enfermedades neurodegenerativas a través de pruebas automatizadas. Cogstate es valorada por su rapidez y fiabilidad en entornos de investigación y clínicas. CANTAB, por su parte,

se distingue por su precisión científica y su capacidad para detectar sutiles cambios cognitivos.

La aplicación desarrollada en este proyecto se diferencia de estas alternativas al enfocarse en la evaluación a largo plazo y en el uso de biomarcadores. Esta aproximación permite no solo la detección temprana sino también el seguimiento detallado de la progresión de la enfermedad, adaptándose a los avances científicos y tecnológicos en el campo. Además, la accesibilidad y adaptabilidad de la aplicación permiten su uso en una variedad de contextos, proporcionando una herramienta integral y avanzada para la evaluación cognitiva.

## 6.6. Desarrollos futuros

En esta sección se exploran diversas líneas de investigación y desarrollo destinadas a mejorar y expandir el proyecto. Se proponen iniciativas para depurar y optimizar las aplicaciones actuales, así como la creación de nuevas variantes que amplíen el alcance de la aplicación a otras plataformas y entornos. Estas propuestas incluyen tanto la resolución de errores identificados durante el desarrollo como la implementación de nuevas versiones que aprovechen herramientas y tecnologías emergentes para mejorar la funcionalidad y accesibilidad de la aplicación. La discusión se centra en cómo abordar estos desafíos y oportunidades para avanzar en la robustez y versatilidad del software.

### 6.6.1. Resolución de errores

Aunque se ha logrado cumplir con los objetivos y las aplicaciones demuestran un buen rendimiento en cuanto a su funcionamiento, existen varios aspectos con un amplio margen de mejora y se han identificado algunos errores.

En primer lugar, se ha comprobado que el algoritmo RPDPM presenta muchos problemas al ser portado a otros entornos. Aunque se ha logrado compilar este algoritmo desarrollado en MATLAB tanto en Win64 (.exe) como en Linux (.sh) sin mayores inconvenientes, las predicciones obtenidas por la versión de Linux varían significativamente de las generadas por la versión de Windows, a pesar de que el código es el mismo. Por lo tanto, se propone una futura investigación para mejorar este aspecto.

Por otro lado, similar al inconveniente anterior, el algoritmo desarrollado en Leaspy no parece estar funcionando correctamente al obtener los resultados normalizados. Este problema se diferencia del anterior porque, en este caso, el motor que obtiene las predicciones es totalmente distinto al RPDPM que se usa como referencia. Por lo tanto, se cree que el error está más relacionado con el motor que con el entorno informático. Además, este motor devuelve resultados muy satisfactorios cuando se consideran los valores sin normalizar, como se muestra en la primera figura de los resultados. Se propone un trabajo de depuración como desarrollo futuro para estudiar esta disparidad entre los resultados antes y después de su normalización. A continuación, se muestra un ejemplo del error mencionado, destacando cómo la primera figura muestra resultados realistas mientras que la segunda no logra alcanzar

la misma precisión:

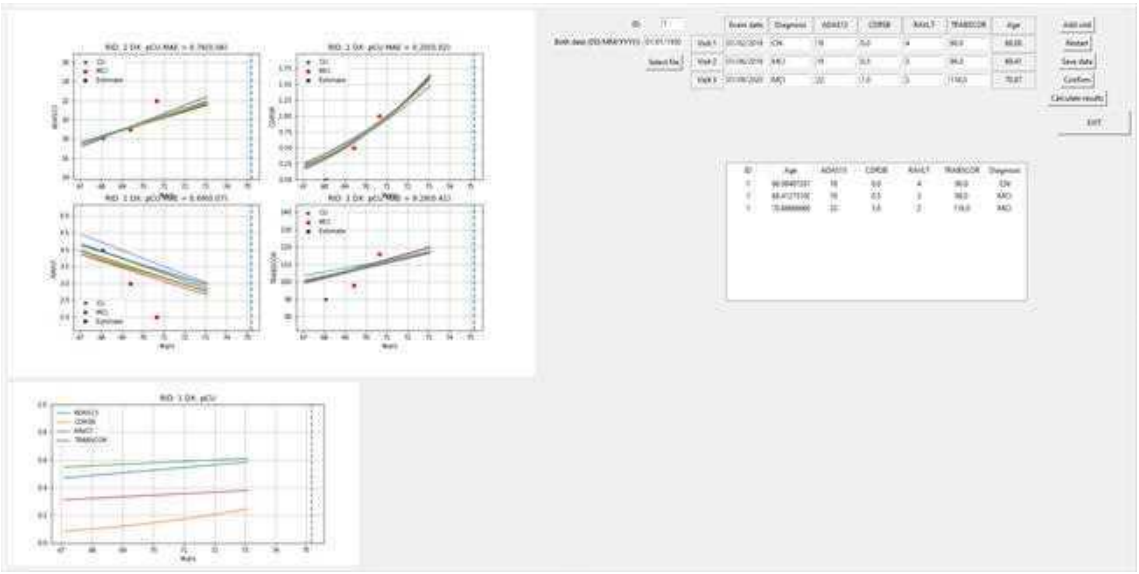

Figura 6.4: Ejemplo del error producido por el algoritmo de Leaspy en la normalización de los resultados.

Otro problema que ha resultado muy complicado de solventar tiene que ver con la elaboración de los ejecutables e instaladores. Aunque las tres aplicaciones funcionan sin inconvenientes antes de ser compiladas, al ejecutarlas mediante un archivo ejecutable o tras ser instaladas a través de un instalador, aparecen errores que no se presentaban antes y que han resultado muy difíciles de resolver. Este problema ocurre específicamente con las dos aplicaciones basadas en Python, ya que las herramientas proporcionadas por MATLAB para compilar programas de este lenguaje funcionan de manera excelente.

Además de proponerse futuras líneas de investigación para solucionar estos errores, desde el principio se tuvo en mente la idea de portar la aplicación principal de este trabajo al entorno de los dispositivos móviles. Esta tarea supondría un gran esfuerzo, por lo que se propone llevarla a cabo mediante un nuevo proyecto de características similares. Varias propiedades de estas aplicaciones, como su portabilidad, su compatibilidad con múltiples plataformas, y la elección de lenguajes de programación sencillos y fáciles de depurar, deberían favorecer esta ambiciosa tarea. Próximamente, se desarrolla con más detalle esta idea.

### 6.6.2. Elaboración de una nueva variante de la aplicación principal

Debido a la sorprendente conveniencia de desarrollar la variante de MATLAB utilizando la herramienta MATLAB App Designer, se propone la creación de una nueva versión del programa principal usando una herramienta similar pero en el lenguaje Python. Esto permitiría evitar las principales desventajas de un entorno de código cerrado y de pago como MATLAB, manteniendo las ventajas de utilizar una herramienta que permita diseñar GUIs de manera visual, como App Designer. Además, se espera que la creación de los ejecutables e instaladores sea más sencilla para esta versión. Con este objetivo, se ha realizado un estudio en busca de las

mejores herramientas que cumplan estos requisitos. A continuación, se presentan las mejores alternativas encontradas:

**Qt Designer [36]**

Qt Designer, parte de las bibliotecas PyQt5 y PySide2, es una herramienta gráfica que facilita el diseño de interfaces de usuario mediante una metodología de arrastrar y soltar. Los diseños se guardan en archivos `.ui`, que posteriormente se pueden convertir en código Python utilizando bindings específicos. Qt Designer destaca por su flexibilidad y capacidad para soportar una amplia variedad de widgets y layouts, lo que permite crear interfaces sofisticadas y personalizadas. Sin embargo, uno de sus inconvenientes es la necesidad de convertir los archivos `.ui` a código Python, lo que añade un paso adicional en el flujo de trabajo. Además, puede tener una curva de aprendizaje más pronunciada para quienes no están familiarizados con Qt.

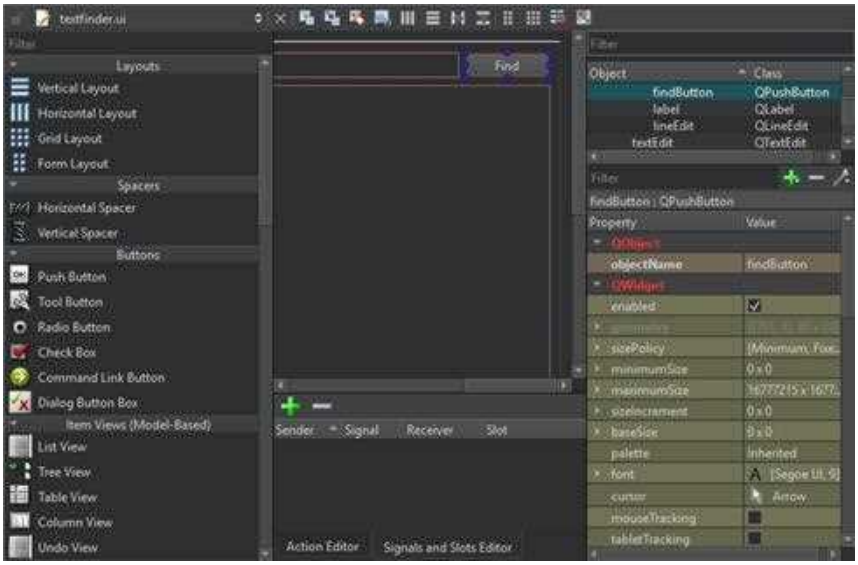


Figura 6.5: Interfaz de Qt Designer.

**PAGE [33]**

PAGE, o Python Automatic GUI Generator, es una herramienta que permite diseñar GUIs para Tkinter de manera visual. Tkinter, siendo la biblioteca estándar de Python para crear interfaces gráficas, se beneficia de PAGE al ofrecer una interfaz intuitiva donde los usuarios pueden arrastrar y soltar componentes para construir la interfaz. Una ventaja significativa de PAGE es su integración directa con Tkinter, lo que facilita el desarrollo de interfaces sin necesidad de escribir código manualmente. No obstante, sus capacidades pueden estar limitadas en comparación con otras herramientas más avanzadas, y la estética de las interfaces creadas con Tkinter puede ser más básica.

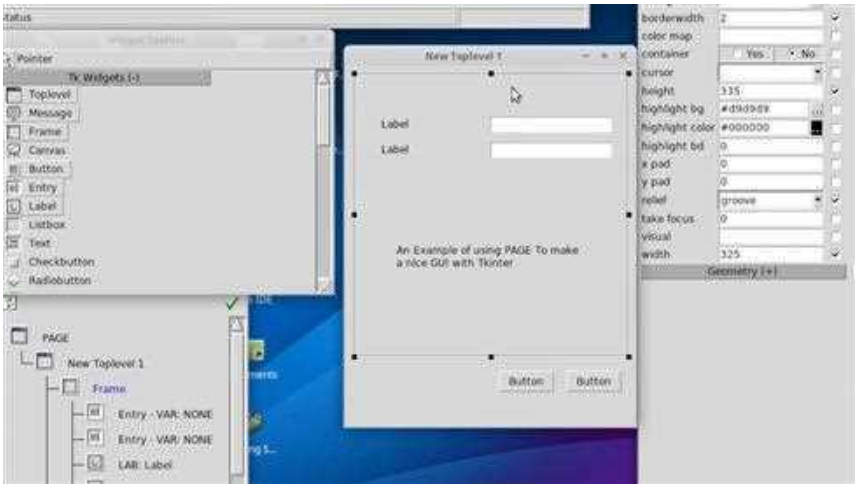


Figura 6.6: Ejemplo de una GUI básica de Tkinter desarrollada con PAGE.

**Kivy Designer [27]**

Kivy Designer es otra herramienta gráfica para el diseño de GUIs, esta vez enfocada en Kivy, una biblioteca de Python orientada al desarrollo de aplicaciones multitáctiles y modernas. Kivy Designer permite crear aplicaciones de forma visual y generar el código correspondiente de manera eficiente. Entre sus ventajas se encuentra el soporte completo para interfaces multitáctiles y la facilidad para crear aplicaciones modulares y escalables. Sin embargo, Kivy puede requerir la instalación de dependencias adicionales y puede ser menos intuitivo para aquellos que no están familiarizados con el entorno de desarrollo de Kivy.

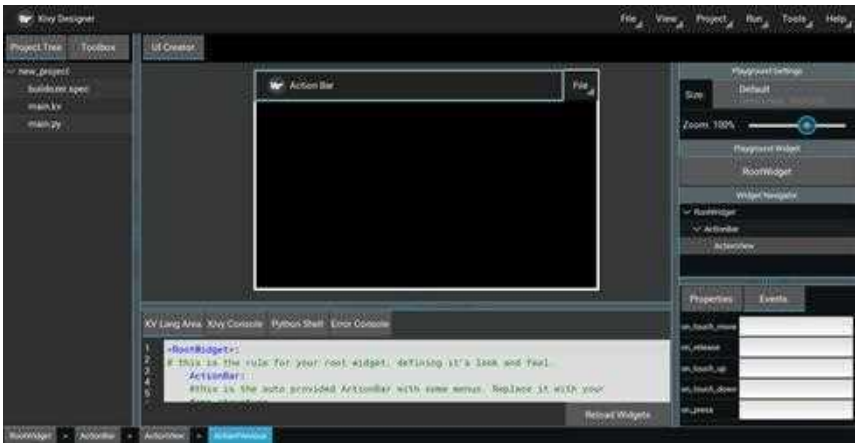


Figura 6.7: Interfaz de Kivy Designer.

**WxFormBuilder [49]**

WxFormBuilder es una herramienta visual para diseñar interfaces de usuario en wxPython. Esta herramienta permite arrastrar y soltar componentes para construir la interfaz y generar el código Python necesario. Una de las principales ventajas de WxFormBuilder es su flexibilidad y potencia, ya que soporta una amplia variedad

de widgets y controles, permitiendo la creación de interfaces complejas. A pesar de sus capacidades avanzadas, wxPython y WxFormBuilder pueden tener una curva de aprendizaje más empinada, especialmente para desarrolladores que no están familiarizados con el marco wxPython.

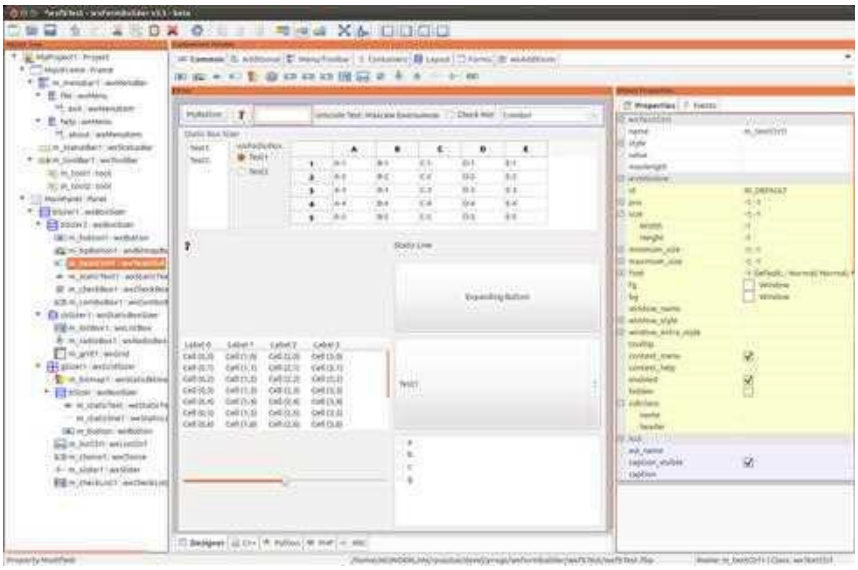

Figura 6.8: Interfaz de WxFormBuilder.

### 6.6.3. Creación de nuevas versiones que permitan dar el salto a otras plataformas o entornos

El desarrollo de versiones futuras de la aplicación contempla la expansión hacia diferentes plataformas y arquitecturas, incluyendo una app para Android tipo cliente/servidor, una aplicación web tipo cliente/servidor y una adaptación de la aplicación principal para macOS. Estas proyecciones tienen como objetivo aumentar la accesibilidad y funcionalidad de la aplicación, permitiendo su uso en una gama más amplia de dispositivos y sistemas operativos.

#### App para Android tipo cliente/servidor [38]

El desarrollo de una aplicación para Android tipo cliente/servidor implica la creación de una aplicación móvil que interactúe con un servidor remoto para la gestión y procesamiento de datos. Esta arquitectura permite que la mayor parte del procesamiento y almacenamiento de datos se realice en el servidor, mientras que la aplicación cliente en el dispositivo Android se utiliza principalmente para la interfaz de usuario y la comunicación con el servidor.

Entre las ventajas de esta aproximación se encuentran la capacidad de manejar grandes volúmenes de datos y la posibilidad de realizar actualizaciones y mantenimientos en el servidor sin necesidad de modificar la aplicación cliente. Sin embargo, se deben considerar aspectos como la gestión de la conectividad de red y la seguridad de la comunicación entre el cliente y el servidor. Herramientas como Android Studio

para el desarrollo de la aplicación móvil y frameworks como Flask o Django para el backend pueden ser de gran utilidad en este proceso.

#### Aplicación web tipo cliente/servidor [39]

Una aplicación web tipo cliente/servidor se basa en la interacción entre un navegador web (cliente) y un servidor web. Este modelo permite que la aplicación sea accesible desde cualquier dispositivo con un navegador web, eliminando la necesidad de instalar software adicional en el dispositivo del usuario. El desarrollo de esta versión de la aplicación implica la creación de una interfaz web interactiva y la implementación de un backend robusto para manejar las solicitudes de los usuarios.

Las principales ventajas de una aplicación web incluyen su accesibilidad y facilidad de actualización, ya que los cambios realizados en el servidor se reflejan inmediatamente en todos los clientes. Además, este enfoque facilita la integración con otros servicios web y bases de datos. Tecnologías como HTML, CSS y JavaScript (para el frontend) y Node.js, Flask o Django (para el backend) son fundamentales en el desarrollo de aplicaciones web. No obstante, se deben considerar aspectos como la optimización del rendimiento y la seguridad de la aplicación web.

#### Adaptación de la aplicación principal para macOS [41]

Siguiendo la línea de los objetivos de este proyecto, se propone ampliar la compatibilidad de la aplicación al segundo sistema operativo más utilizado en sistemas de escritorio en la actualidad: macOS [42]. La adaptación de la aplicación principal para macOS implica la modificación del software existente para que funcione de manera nativa en el sistema operativo macOS. Esto puede incluir la reescritura de partes del código, la utilización de bibliotecas y frameworks específicos de macOS, y la adaptación de la interfaz de usuario para alinearse con las convenciones de diseño de Apple.

Entre las ventajas de desarrollar una aplicación nativa para macOS se encuentran el mejor rendimiento y la integración profunda con las características del sistema operativo, como las notificaciones, el almacenamiento en iCloud y los servicios de seguridad de macOS. Sin embargo, se requiere un conocimiento específico del entorno de desarrollo de Apple y de herramientas como Xcode y Swift. Además, la adaptación puede implicar un esfuerzo considerable en la reconfiguración del código existente y en la realización de pruebas exhaustivas para garantizar la compatibilidad y la estabilidad de la aplicación en macOS.

## 6.7. Conclusiones

En este capítulo se ha realizado una reflexión exhaustiva sobre los resultados obtenidos a lo largo del proyecto, analizando su cumplimiento con los objetivos iniciales y comparando las distintas alternativas tecnológicas empleadas. Se ha destacado la importancia de haber desarrollado una aplicación en Python, subrayando su capacidad para ofrecer una interfaz simple, intuitiva y multiplataforma, con un motor desacoplado que favorece la flexibilidad y escalabilidad futura del sistema.

El análisis comparativo de los sistemas operativos Windows y Linux ha destacado las fortalezas y desafíos de cada plataforma. Windows ha proporcionado un entorno de desarrollo estable y familiar, mientras que Linux ha ofrecido eficiencia y personalización. No obstante, problemas como la portabilidad del algoritmo RPDPM destacan la necesidad de futuras mejoras en la compatibilidad entre plataformas.

La comparación entre los motores RPDPM y Leaspy ha resaltado la importancia de trabajar con un motor desacoplado y adaptable, demostrando que ambos enfoques pueden ser efectivos, aunque cada uno presenta desafíos únicos. La flexibilidad de la aplicación para integrar diferentes motores demuestra su potencial para adaptarse a diversas necesidades y contextos clínicos.

En cuanto a las alternativas del mercado, herramientas como Cognivue, Cogstate y Cambridge Cognition (CANTAB) ofrecen enfoques robustos para la evaluación cognitiva. Sin embargo, la aplicación desarrollada en este proyecto se distingue por su enfoque en la evaluación a largo plazo utilizando biomarcadores, proporcionando una perspectiva más completa y adaptable a los avances tecnológicos.

Finalmente, las propuestas de desarrollos futuros han identificado áreas clave para la mejora de la aplicación, incluyendo la resolución de errores, la creación de nuevas variantes de la aplicación principal, y la expansión hacia plataformas móviles y otros entornos operativos. Estas propuestas aseguran que la aplicación puede evolucionar para satisfacer las necesidades cambiantes de los usuarios y mantenerse relevante en un campo en rápida evolución.

En resumen, este proyecto ha demostrado la viabilidad y eficacia de una aplicación de evaluación cognitiva desarrollada en Python, con un enfoque en la flexibilidad, escalabilidad y adaptabilidad a diferentes entornos y necesidades. Las reflexiones y comparaciones presentadas en este capítulo no solo validan el trabajo realizado, sino que también establecen una hoja de ruta clara para futuros desarrollos e investigaciones.

# Capítulo 7

# Anexo

En este capítulo se explican métodos que se han llevado a cabo durante el desarrollo del proyecto que no están directamente relacionados con el tema principal.

## A.1. Guía de instalación de la variante de MATLAB

En esta sección, se explica detalladamente cómo instalar la versión de la aplicación principal desarrollada en MATLAB tanto para Windows como para sistemas operativos basados en Linux. Los instaladores para ambos sistemas operativos se pueden obtener de la carpeta *demo/Instaladores/GUI_MATLAB*.

Dentro de esta carpeta, existe una para el instalador de Windows y otra para el de Linux. En cada una de estas, se encuentran el archivo del proyecto de MATLAB Compiler `GUI_MATLAB.prj` y la carpeta 'GUI_MATLAB'. El contenido de este último directorio sigue siendo el mismo para ambos instaladores y se explica a continuación:

- **for_redistribution**: Esta carpeta contiene todos los archivos necesarios para redistribuir la aplicación: en este caso el instalador completo, que ofrece la posibilidad de instalar la versión de MATLAB Runtime necesaria para ejecutar la aplicación.
- **for_redistribution_files_only**: Esta carpeta contiene solo los archivos mínimos necesarios para la redistribución de la aplicación: el ejecutable. Tener acceso a este ejecutable es suficiente para poder acceder a todo lo que te ofrece la aplicación, pero este necesita de una instalación válida de la versión adecuada de MATLAB Runtime en el equipo
- **for_testing**: Esta carpeta incluye archivos y scripts utilizados para probar la aplicación antes de su distribución final.

### A.1.1. Instalación en Windows

Para instalar la aplicación en un sistema operativo Windows, se deben seguir los siguientes pasos:

1. **Descarga del instalador**
   - Navegar a la carpeta *demo/Instaladores/GUI_MATLAB/Windows/GUI_MATLAB/for_redistribution*.

- Descargar el archivo `MyAppInstaller_mcr.exe` contenido en la carpeta.

2. **Ejecución del instalador**
   - Hacer doble clic en el archivo `MyAppInstaller_web.exe`.
3. **Proceso de instalación**
   - Seguir las instrucciones del asistente de instalación.
   - Seleccionar el directorio de instalación deseado.
   - Aceptar los términos y condiciones de la licencia de MATLAB Runtime.
   - Hacer clic en 'Next' para continuar con la instalación.
4. **Incluir MATLAB Runtime**
   - Durante el proceso de instalación, se preguntará si se desea incluir MATLAB Runtime.
   - Seleccionar 'Yes' para incluir MATLAB Runtime y hacer clic en 'Next'.
   - Si ya se tiene MATLAB Runtime instalado, se puede seleccionar 'No' y especificar la ruta de instalación existente.
5. **Finalización**
   - Una vez completada la instalación, hacer clic en 'Finish' para cerrar el asistente de instalación.
   - La aplicación estará lista para ser utilizada. Para ejecutarla, solamente es necesario hacer doble clic en el ejecutable generado, que se encuentra en el directorio seleccionado para la instalación y en el escritorio si se marcó la opción correspondiente.

### A.1.2. Instalación en Linux

Para instalar la aplicación en un sistema operativo basado en Linux, se deben seguir los siguientes pasos:

1. **Descarga del instalador**
   - Navegar a la carpeta *demo/Instaladores/GUI_MATLAB/Linux/GUI_MATLAB/for_redistribution*.
   - Descargar el archivo `MyAppInstaller_mcr.install` contenido en la carpeta.
2. **Ejecución del instalador**
   - Abrir una terminal y navegar al directorio donde se encuentra el archivo descargado.
   - Ejecutar el siguiente comando para iniciar el instalador:

     ```
     sudo ./MyAppInstaller_mcr.install
     ```
3. **Proceso de instalación**
   - Seguir las instrucciones del asistente de instalación.

- Seleccionar el directorio de instalación deseado.
- Aceptar los términos y condiciones de la licencia de MATLAB Runtime.
- Hacer clic en 'Next' para continuar con la instalación.

4. **Incluir MATLAB Runtime**
   - Durante el proceso de instalación, se preguntará si se desea incluir MATLAB Runtime.
   - Seleccionar 'Yes' para incluir MATLAB Runtime y hacer clic en 'Next'.
   - Si ya se tiene MATLAB Runtime instalado, se puede seleccionar 'No' y especificar la ruta de instalación existente.
5. **Finalización**
   - Una vez completada la instalación, la aplicación estará lista para ser utilizada.
   - Para ejecutar la aplicación, se deben seguir los siguientes pasos:
     a) Abrir una terminal y navegar al directorio seleccionado para la instalación.
     b) Ejecutar el siguiente comando para darle permisos de ejecución al ejecutable llamado `run_GUI_MATLAB.sh`:
     ```
     chmod +x run_GUI_MATLAB.sh
     ```
     c) Ejecutar `run_GUI_MATLAB.sh` indicando el directorio de instalación de MATLAB Runtime mediante el siguiente comando:
     ```
     ./run_GUI_MATLAB.sh <Directorio de MATLAB Runtime>
     ```

## A.2. Depuración de una función de MATLAB que se ejecuta desde un script de Python

Para la elaboración del programa, se ha utilizado en su mayoría código de Python: la interfaz, la obtención y tratado de datos de entrada o el cálculo de resultados como trayectorias son algunas de las funcionalidades que están programadas en este lenguaje. Sin embargo, el algoritmo utilizado para el cálculo de estos resultados se encuentra en MATLAB, por lo que, una vez decidida la forma de realizar esta comunicación entre los dos lenguajes de programación, hubo que pensar en cómo depurar las funciones de MATLAB que se están ejecutando de forma externa.

Para ello se valoraron distintas formas de conseguirlo, principalmente una que ofrece la biblioteca utilizada para la comunicación entre ambos lenguajes de programación mediante funciones de la misma. También se intentó conectar con el hilo que está utilizando Python para ejecutar la función externa desde el propio cliente de MATLAB, pero ninguno de estos métodos terminaba de funcionar suficientemente bien. Finalmente, se encontró una forma de conseguir esto que resultó ser bastante simple y conveniente: colocar puntos de interrupción mediante líneas de código.

Previamente se comprobó que los puntos de interrupción colocados de forma manual eran completamente ignorados al ejecutarse esta función, ya que estos solo se tienen

en cuenta cuando la función de MATLAB se ejecuta desde el mismo cliente desde el que se han colocado los puntos, mientras que en este caso el código se había editado desde el cliente convencional de MATLAB, para después ser ejecutado por un motor externo creado por el script de Python. Sin embargo, al utilizar líneas de código para crear los puntos de interrupción, este se coloca únicamente al ejecutarse dicha línea.

Existen varias formas de crear puntos de interrupción de esta forma, como 'dbstop if condition', 'dbstop in file if expression', 'dbstop in file at location if expression' o 'dbstop in file at location', pero se optó por esta última estructura por comodidad. El resultado final de depurar mediante este método consiste en una interrupción de la ejecución del script de Python (independientemente de si este se estaba depurando o no) y la aparición de un editor convencional de MATLAB, donde se puede ejecutar línea a línea la función si se desea, comprobar los valores de las variables, con especial interés en aquellas que se han obtenido como input de Python o realizar una depuración más en profundidad de otras funciones que pueda estar utilizando MATLAB.

## A.3. Llamada a una función de Python desde MATLAB

Aunque finalmente no ha sido utilizada para el proyecto, tras investigar en profundidad la API matlab.engine [30], se llegó a la conclusión de que esta herramienta puede tener una utilidad enorme para proyectos como este. La capacidad de integración entre MATLAB y Python que provee esta API permite aprovechar las ventajas de ambos entornos de programación, facilitando el desarrollo de aplicaciones híbridas que pueden beneficiarse de las bibliotecas y herramientas disponibles en cada uno de estos lenguajes. En esta sección, se describe el procedimiento para invocar una función de Python desde un programa de MATLAB.

Para realizar una llamada a una función de Python desde MATLAB, se deben seguir los siguientes pasos:

1. **Instalación de matlab.engine**: Antes de poder utilizar matlab.engine, es necesario asegurarse de que esta API está instalada y configurada correctamente. MATLAB proporciona una forma sencilla de instalarla ejecutando el siguiente comando en la línea de comandos de MATLAB:

   ```
   >> cd (fullfile(matlabroot,'extern','engines','python'))
   >> system('python setup.py install')
   ```

   Este comando instalará la API de MATLAB para Python en el entorno de Python seleccionado.

2. **Iniciar el motor de MATLAB desde Python**: Para llamar a una función de Python desde MATLAB, primero es necesario iniciar el motor de MATLAB en el entorno de Python. A continuación se muestra un ejemplo básico de cómo hacerlo:

   ```
   import matlab.engine
   eng = matlab.engine.start_matlab()
   ```

3. **Definir la función de Python**: Para este ejemplo, la siguiente función es la que se desea invocar desde MATLAB:

```
def mi_funcion_python(x, y):
return x + y
```

Se trata de una función simple para facilitar el entendimiento de este proceso. Se encuentra en un archivo llamado `mi_modulo.py` y se puede consultar en *demo/Anexo/MatlabEngine_MATLAB/mi_modulo.py*.

4. **Llamar a la función de Python desde MATLAB**: En el entorno de MATLAB, se utilizan las funciones de la API matlab.engine para invocar la función de Python. A continuación se muestra un ejemplo de cómo hacerlo:

```
import matlab.engine
eng = matlab.engine.start_matlab()

% Llamada a la función de Python
result = eng.mi_funcion_python(5, 3)
fprintf('El resultado es: %d\n', result)
```

En este ejemplo, se inicia el motor de MATLAB, se llama a la función `mi_funcion_python` con los argumentos `5` y `3`, y se imprime el resultado.

5. **Cerrar el motor de MATLAB**: Finalmente, es importante cerrar el motor de MATLAB una vez que se ha terminado de utilizar para liberar recursos. Esto se puede hacer con el siguiente comando:

```
eng.quit()
```

Este procedimiento permite a los usuarios de MATLAB aprovechar las capacidades de Python, integrando sin problemas funciones y bibliotecas de Python en sus scripts y aplicaciones de MATLAB. La combinación de MATLAB y Python puede ser especialmente poderosa en proyectos que requieren análisis numérico avanzado, visualización de datos y acceso a bibliotecas específicas de Python. Los archivos necesarios para realizar el ejemplo desarrollado durante esta sección se encuentran en el directorio *demo/Anexo/MatlabEngine_MATLAB*.

## A.4. Creación de una GUI multiplataforma básica con Tkinter

Tkinter es la biblioteca estándar de Python para la creación de GUI y es compatible con Windows, macOS y Linux. Esto la convierte en una opción excelente para el desarrollo de aplicaciones multiplataforma. A continuación, se detalla cómo crear una GUI básica usando Tkinter.

### A.4.1. Instalación de Tkinter

Tkinter viene preinstalado con la mayoría de las distribuciones de Python. Sin embargo, si se necesita instalar manualmente, se puede hacer mediante `pip` (en sistemas donde no está preinstalado):

```
pip install tk
```

### A.4.2. Creación de una aplicación básica

El siguiente pseudocódigo muestra cómo crear una ventana simple con un botón usando Tkinter:

**Algorithm 7** Creación de una aplicación básica en Tkinter

```
Importar módulos necesarios
import tkinter as tk
from tkinter import messagebox
Definir la función de callback para el botón
def on_click():
  messagebox.showinfo('Información', 'Botón presionado')
Definir la clase principal de la aplicación
class MyApp:
  def __init__(self, root):
    self.root = root
    self.root.title('Aplicación multiplataforma')
      btn = tk.Button(self.root, text='Presióname', command=on_click)
    btn.pack(pady=20)
Definir la función principal para iniciar la aplicación
def main():
  root = tk.Tk()
  app = MyApp(root)
  root.mainloop()
Ejecutar la función principal si el script es ejecutado directamente
if __name__ == '__main__':
  main()
```

El código anterior es completamente compatible con Windows, macOS y Linux. Tkinter abstrae las diferencias entre los sistemas operativos, permitiendo que la misma base de código se ejecute en cualquier plataforma sin modificaciones. Esto es posible gracias a que Tkinter utiliza la biblioteca Tcl/Tk, que es multiplataforma por naturaleza. El código desarrollado durante esta sección se encuentra en la carpeta *demo/Anexo/Ejemplo_Tkinter/app_tkinter.py*.

## A.5. Creación de una GUI multiplataforma básica con MATLAB App Designer

MATLAB App Designer es una herramienta potente para la creación de aplicaciones GUI que pueden ejecutarse en diferentes sistemas operativos. A continuación, se detalla cómo crear una aplicación básica usando MATLAB App Designer.

### A.5.1. Creación de una aplicación básica

El siguiente pseudocódigo describe los pasos para crear una aplicación básica en MATLAB App Designer:

**Algorithm 8** Creación de una aplicación básica en MATLAB App Designer

```
# Abrir MATLAB y seleccionar App Designer en la pestaña de Home.
# Crear una nueva aplicación.

# Arrastrar un botón desde la galería de componentes a la ventana de diseño.
# Configurar las propiedades del botón (por ejemplo, cambiar el texto a 'Presióname').

# Agregar una función de devolución de llamada para el evento 'ButtonPushed'.
function ButtonPushed(app, event)
   msgbox('Botón presionado', 'Información');
end
```

MATLAB App Designer genera aplicaciones que pueden ejecutarse en cualquier sistema operativo compatible con MATLAB, incluyendo Windows, macOS y Linux. Esto asegura que la aplicación sea accesible y funcional en múltiples plataformas sin necesidad de cambios adicionales.

En cuanto a la distribución de la aplicación, MATLAB permite empaquetar aplicaciones en instaladores o ejecutables utilizando MATLAB Compiler. Este enfoque asegura que todas las dependencias y archivos necesarios estén incluidos, permitiendo que la aplicación se ejecute independientemente de la instalación de MATLAB. El programa de ejemplo explicado durante esta sección se puede consultar en el directorio *demo/Anexo/Ejemplo_AppDesigner/app_mldesigner.mlapp*.

## A.6. Originalidad del documento

Con el objetivo de validar y demostrar la originalidad de esta memoria, se ha utilizado la herramienta Turnitin [44]. Turnitin es un software especializado en la detección de plagio y verificación de la originalidad de documentos. Esta herramienta compara el contenido del texto con una amplia base de datos que incluye artículos académicos, contenido web y trabajos estudiantiles previamente enviados, generando un informe de similitud que destaca las coincidencias y posibles plagios. Esta verificación asegura que el trabajo presentado es auténtico y cumple con los estándares de integridad académica.

A continuación, se adjunta un resumen del informe de similitud obtenido:

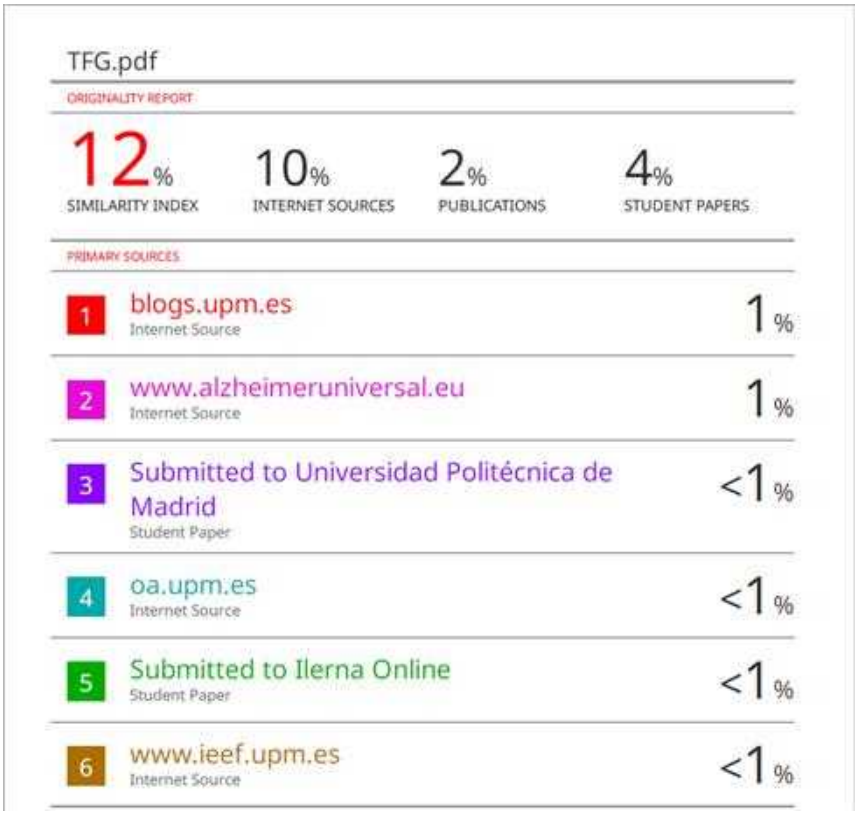


Figura 7.1: Resumen del informe sobre similitud de Turnitin.

Como se observa en la imagen, el índice total de similitud es del 12 %. Para ver más detalles sobre este informe, se puede acceder al documento completo en *demo/Anexo/Informe_Turnitin.pdf*.

# Bibliografía


[1] FileZilla. https://filezilla-project.org/. Consultado en marzo de 2024.

[2] Spyder IDE. https://www.spyder-ide.org/. Consultado en marzo de 2024.

[3] Xming. http://www.straightrunning.com/XmingNotes/. Consultado en marzo de 2024.

[4] ADNI. Documentación oficial de ADNI. https://adni.loni.usc.edu, sin fecha. Consultado en junio de 2024.

[5] Alzheimer's Disease International. Dementia facts and figures, 2023.

[6] Alzheimer's Disease International. From plan to impact v, 2023.

[7] Anaconda. Documentación oficial de Anaconda. https://www.anaconda.com, sin fecha. Consultado en junio de 2024.

[8] Alzheimer's Association. 2023 alzheimer's disease facts and figures. Alzheimer's & Dementia, 19(4), 2023.

[9] M. Bentivoglio and G.G. Zucconi. Cuando el cerebro envejece: mitos y certezas sobre un proceso universal (e inevitable). Neurociencia y psicología. Emse Edapp, S.L., 2018.

[10] Cambridge Cognition. Cantab - cambridge cognition. https://www.cambridgecognition.com/cantab. Consultado en mayo de 2024.

[11] CEAFA. Nota de prensa: El 80 % de las personas con alzheimer son cuidadas por sus familias, que asumen de media el 87 % del coste total que conlleva la enfermedad, 2023.

[12] G Kerchner CJ Lansdall, LM Butler. Alzheimer's disease assessment scale-cognitive subscale variants in mild cognitive impairment and mild alzheimer's disease: change over time and the effect of interventions. Alzheimer's Research & Therapy, 8(1):1–10, 2016.

[13] G Kerchner CJ Lansdall, LM Butler. Anchor- and distribution-based methods to establish clinically meaningful score changes on the clinical dementia rating scale-sum of boxes in patients with prodromal alzheimer's disease. Journal of Prevention of Alzheimer's Disease, 2019.

[14] Cognivue. Cognivue. https://cognivue.com/. Consultado en mayo de 2024.

[15] Cogstate. Cogstate. https://www.cogstate.com. Consultado en mayo de 2024.

[16] Confederación Española de Asociaciones de Familiares de personas con Alzheimer y otras demencias (CEAFA). Censo de personas con alzheimer y otras demencias en españa. resultados y conclusiones, 2022. Revisado por Ainhoa Etayo Zabalegui, Secretaría Técnica de CEAFA, Diseño y Maquetación por iLUNE Diseño, Impresión por RODONA, Depósito Legal: NA-2902-2022.

[17] Gad Marshall Catherine M. Roe Paul K. Crane Ron A. Thomas Michael C. Donohue Dorene M. Rentz, Reisa A. Sperling. Optimizing the preclinical alzheimer's cognitive composite with semantic processing: the pacc5. Alzheimer's & Dementia, 13(3):1175–1183, 2017.

[18] EpData. Las cifras del alzheimer en españa: Número de personas, mortalidad, muertes, gráficos, datos, 2023.

[19] Sarah Tomaszewski Farias, Karen Lau, Danielle Harvey, Katherine G Denny, Cheyanne Barba, and Anthony N Mefford. Early functional limitations in cognitively normal older adults predict diagnostic conversion to mild cognitive impairment. Journal of the American Geriatrics Society, 65(6):1152–1158, 2017.

[20] Mostafa Mehdipour Ghazi, Mads Nielsen, Akshay Pai, Marc Modat, M Jorge Cardoso, Sébastien Ourselin, and Lauge Sørensen. Robust parametric modeling of alzheimer's disease progression. NeuroImage, 225:117460, 2021.

[21] P.G. Gregorio. Neurodegeneración: Alzheimer, Parkinson y ELA. Neurociencia y psicología. Emse Edapp, S.L., 2018.

[22] Oskar Hansson, John Seibyl, Erik Stomrud, Henrik Zetterberg, John Q Trojanowski, Tobias Bittner, Valeria Lifke, Veronika Corradini, Udo Eichenlaub, Richard Batrla, et al. Csf biomarkers of alzheimer's disease concord with amyloid-$\beta$ pet and predict clinical progression: a study of fully automated immunoassays in biofinder and adni cohorts. Alzheimer's & dementia, 14(11):1470–1481, 2018.

[23] J García-Sancho J Navarro E Mudarra-Sánchez J Espinosa, S Martín-Arévalo. Validation and normative data of the spanish version of the rey auditory verbal learning test and associated long-term forgetting measures in middle-aged and elderly adults. Frontiers in Aging Neuroscience, 14:809019, 2022.

[24] Clifford R Jack Jr, David A Bennett, Kaj Blennow, Maria C Carrillo, Billy Dunn, Samantha Budd Haeberlein, David M Holtzman, William Jagust, Frank Jessen, Jason Karlawish, et al. Nia-aa research framework: toward a biological definition of alzheimer's disease. Alzheimer's & Dementia, 14(4):535–562, 2018.

[25] Mark A. Mintun S. Stroup Brian Gordon Stephanie Raedle Catherine M. Roe John C. Morris, Margaret M. Williams. Neuropsychological performance across cognitive impairment and alzheimer's disease. Journal of Alzheimer's Disease, 27(3):423–433, 2011.

[26] Angie A Kehagia, Roger A Barker, and Trevor W Robbins. Cognitive impairment in parkinson's disease: the dual syndrome hypothesis. Neurodegenerative diseases, 11(2):79–92, 2012.

[27] Kivy Project. Kivy designer. `https://github.com/kivy/kivy-designer`. Consultado en junio de 2024.

[28] Igor Koval. Learning multimodal digital models of disease progression from longitudinal data: methods & algorithms for the description, prediction and simulation of Alzheimer's disease progression. PhD thesis, Institut polytechnique de Paris, 2020.

[29] MathWorks. Matlab compiler sdk. `https://es.mathworks.com/products/matlab-compiler-sdk.html`, 2023. Consultado en junio de 2024.

[30] MathWorks. MATLAB Engine API, 2023.

[31] MathWorks. Documentación oficial de MATLAB APP Designer. `https://es.mathworks.com/products/matlab/app-designer.html`, sin fecha. Consultado en junio de 2024.

[32] Ceyhun Ozgur, Taylor Colliau, Grace Rogers, Zachariah Hughes, et al. Matlab vs. python vs. r. Journal of data Science, 15(3):355–371, 2017.

[33] PAGE - Python Automatic GUI Generator. Page - python automatic gui generator. `https://page.sourceforge.io/`. Consultado en junio de 2024.

[34] PyInstaller Development Team. Pyinstaller documentation. `https://pyinstaller.readthedocs.io/en/stable/`, 2023. Consultado en junio de 2024.

[35] Python. Python. `https://www.python.org/`. Consultado en marzo de 2024.

[36] Qt Project. Qt designer. `https://doc.qt.io/qt-5/qtdesigner-manual.html`. Consultado en junio de 2024.

[37] Real Python. The advantages of cross-platform development with python. `https://realpython.com/cross-platform-development/`, 2023. Consultado en junio de 2024.

[38] M. H. Reddy and B. S. Rekha. Effective development of android applications using android services. International Journal of Computer Applications, 2012.

[39] A. B. Sallow, H. I. Dino, Z. S. Ageed, and M. R. Mahmood. Client/server remote control administration system: design and implementation. International Journal of Multidisciplinary Research and Publications, 2020.

[40] Simplilearn. MATLAB vs. Python. `https://www.simplilearn.com/matlab-vs-python-article`. Consultado en marzo de 2024.

[41] A. Singh. Mac OS X Internals: A Systems Approach. Addison-Wesley Professional, 2006.

[42] Statcounter. Desktop Operating System Market Share Worldwide. `https://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-202304-202404`. Consultado en mayo de 2024.

[43] TeXstudio. Documentación oficial de TeXstudio. `https://www.texstudio.org`, sin fecha. Consultado en junio de 2024.

[44] Turnitin. Turnitin. `https://www.turnitin.com`, 2023. Consultado en junio de 2024.

[45] Wikipedia. LaTeX. https://es.wikipedia.org/wiki/LaTeX. Consultado en marzo de 2024.

[46] Wikipedia. MATLAB. https://es.wikipedia.org/wiki/MATLAB, sin fecha. Consultado en marzo de 2024.

[47] Wikipedia contributors. PuTTY. https://es.wikipedia.org/wiki/PuTTY, 2024. Consultado en marzo de 2024.

[48] World Health Organization. Dementia fact sheet, 2023.

[49] WxFormBuilder Team. Wxformbuilder. https://github.com/wxFormBuilder/wxFormBuilder. Consultado en junio de 2024.