Desarrollo de una Herramienta de Clasificación para la Detección de la Enfermedad de Alzheimer

Universitat Oberta de Catalunya

Marina Miguel García

MU Bioinf. y Bioest.
Apredizaje automático

Nombre Tutor/a de TF
Antonio Jesus Adsuar Gomez
Profesor/a responsable de la asignatura
Carles Ventura Royo

Fecha Entrega
06/2024

FICHA DEL TRABAJO FINAL

Título del trabajo:	Desarrollo de una Herramienta de Clasificación para el Análisis de Datos
Nombre del autor:	Marina Miguel García
Nombre del director/a:	Antonio Jesus Adsuar Gomez
Nombre del PRA:	Carles Ventura Royo
Fecha de entrega (mm/aaaa):	06/2024
Titulación o programa:	Bioinformática y Bioestadística
Área del Trabajo Final:	Aprendizaje Automático
Idioma del trabajo:	Castellano
Palabras clave	Sofware, Machine Learning, clasificación, Alzheimer, CNN

Resumen del Trabajo

La enfermedad de Alzheimer (EA) es una patología neurodegenerativa progresiva que requiere detección temprana para mejorar la calidad de vida de los afectados. Este Trabajo de Fin de Máster (TFM) desarrolló una herramienta de clasificación para predecir la EA mediante el análisis de imágenes de resonancia magnética (MRI) y datos clínicos usando aprendizaje automático.

Utilizando imágenes del conjunto de datos OASIS-1, se normalizaron y preprocesaron las imágenes GFC y Masked GFC, aunque el redimensionamiento a 64x64x64 píxeles conllevó una pérdida de resolución. Los modelos de redes neuronales convolucionales (CNN) alcanzaron precisiones de 72.89% y 52.58% para las imágenes GFC y Masked GFC, respectivamente. Modelos adicionales de SVM, Random Forest y Gradient Boosting lograron una precisión del 71.43%, destacando la calidad de las características extraídas.

Se desarrolló una interfaz web en Django para la carga y clasificación de datos, pero se encontraron fallos en el proceso de clasificación que impidieron su evaluación completa.

Los resultados son prometedores, pero se requiere optimización adicional y resolución de problemas técnicos para mejorar la precisión y funcionalidad de la herramienta. Este trabajo proporciona una base sólida para futuras investigaciones en el diagnóstico precoz de la EA mediante análisis de imágenes y aprendizaje automático.

Abstract

Alzheimer's disease (AD) is a progressive neurodegenerative disorder that requires early detection to improve the quality of life for those affected. This Master's Thesis developed a classification tool to predict AD using magnetic resonance imaging (MRI) and clinical data through machine learning techniques.

Using images from the OASIS-1 dataset, GFC and Masked GFC images were normalized and preprocessed, though resizing to 64x64x64 pixels led to significant resolution loss. Convolutional neural network (CNN) models achieved accuracies of 72.89% and 52.58% for GFC and Masked GFC images, respectively. Additional models, including SVM, Random Forest, and Gradient Boosting, reached an accuracy of 71.43%, highlighting the quality of the extracted features.

A Django-based web interface was developed for data upload and classification, but classification process errors prevented a complete evaluation of the tool.

The initial results are promising, but further model optimization and technical issue resolution are needed to enhance the tool's accuracy and functionality. This work lays a solid foundation for future research in early AD diagnosis through image analysis and machine learning.

Índice

Agradecimientos

Los datos fueron proporcionados [en parte] por OASIS. Investigadores principales: D. Marcus, R. Buckner, J. Csernansky, J. Morris; P50 AG05681, P01 AG03991, P01 AG026276, R01 AG021910, P20 MH071616, U24 RR021382.

1. Introducción

La enfermedad de Alzheimer (EA) es una patología neurodegenerativa progresiva que afecta a millones de personas en todo el mundo y representa una carga significativa tanto para los pacientes como para sus familias y el sistema de salud. La detección temprana y precisa de la EA es crucial para implementar intervenciones oportunas que puedan ralentizar la progresión de la enfermedad y mejorar la calidad de vida de los afectados. En este contexto, el análisis de imágenes de resonancia magnética (MRI) ha emergido como una técnica prometedora para el diagnóstico precoz de la EA.

Recientes avances en el campo del aprendizaje automático y el procesamiento de imágenes han proporcionado nuevas herramientas para mejorar la precisión y eficiencia en el diagnóstico de la EA. Lahmiri (2023) demostró que la integración de redes neuronales convolucionales (CNN), el algoritmo k-Nearest Neighbors (kNN) y la optimización bayesiana puede lograr un diagnóstico eficiente de la EA a partir de imágenes MRI . Este enfoque multidisciplinario aprovecha las fortalezas de cada técnica para proporcionar un sistema de diagnóstico robusto.

Asimismo, Saratxaga et al. (2021) desarrollaron una solución basada en aprendizaje profundo que utiliza imágenes MRI para predecir la EA, mostrando resultados prometedores en términos de precisión y robustez . Este estudio resalta la capacidad de las redes neuronales profundas para aprender características complejas de las imágenes, superando los enfoques tradicionales en términos de rendimiento.

Por otro lado, Pinaya et al. (2021) emplearon modelos normativos para detectar la progresión de la EA en estudios multicohorte, ofreciendo una perspectiva novedosa sobre cómo modelar la variabilidad interindividual y detectar cambios sutiles asociados con la enfermedad . Este enfoque es particularmente útil para identificar etapas tempranas de la EA, donde las intervenciones pueden ser más efectivas.

Finalmente, Gupta et al. (2022) revisaron los métodos de diagnóstico asistido por computadora (CAD) supervisados para clasificar trastornos neurodegenerativos basados en la EA, destacando la importancia de los métodos supervisados para mejorar la precisión diagnóstica . Estos métodos se basan en la clasificación de patrones en los datos de imágenes, permitiendo una detección más precisa y rápida.

Inspirado por estos estudios, este Trabajo de Fin de Máster (TFM) tiene como objetivo desarrollar una herramienta de clasificación para predecir la EA mediante el análisis de imágenes MRI. Utilizando técnicas avanzadas de aprendizaje automático y reducción de dimensionalidad, este proyecto busca

mejorar la precisión y eficiencia del diagnóstico, facilitando la implementación de intervenciones tempranas y mejorando la gestión de la EA en el ámbito clínico.

El presente trabajo se estructura de la siguiente manera: en primer lugar, se describe la obtención y preprocesamiento de los datos; a continuación, se detalla la metodología empleada para la reducción de dimensionalidad y el desarrollo de modelos de aprendizaje automático; finalmente, se presentan los resultados obtenidos y se discuten las implicaciones y posibles futuras direcciones de investigación en este campo.

1.1. Contexto y justificación del Trabajo

El Alzheimer es una enfermedad neurodegenerativa progresiva que afecta a millones de personas en todo el mundo. La detección temprana y precisa de esta enfermedad es crucial para el manejo y el tratamiento de los pacientes. En este contexto, el análisis de imágenes de resonancia magnética (MRI) de cerebros ha emergido como una técnica prometedora para la detección precoz del Alzheimer.

Este trabajo de Fin de Máster se centra en el desarrollo de una herramienta de clasificación mediante técnicas de aprendizaje automático para predecir la enfermedad de Alzheimer a partir de imágenes de MRI. El conjunto de datos OASIS-1 (Open Access Series of Imaging Studies) proporciona las imágenes y los datos demográficos y clínicos necesarios para este estudio. La justificación de este proyecto radica en la necesidad de mejorar la precisión y la eficiencia de los métodos actuales de diagnóstico, proporcionando una herramienta automatizada que pueda integrarse en entornos clínicos para facilitar el diagnóstico precoz y mejorar los resultados de los pacientes.

1.2. Objetivos del Trabajo

1. Objetivos generales:
Desarrollar una herramienta de clasificación mediante Machine Learning para el análisis de datos biomédicos con el fin de predecir la enfermedad de Alzheimer.

2. Objetivos específicos:
   I. Investigar y seleccionar algoritmos de Machine Learning adecuados para la clasificación de datos biomédicos.
   II. Recopilar y preprocesar conjuntos de datos.
   III. Implementar la suite de software con los algoritmos seleccionados y una interfaz web.
   IV. Evaluar el rendimiento de la herramienta.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

Sostenibilidad

La sostenibilidad es un aspecto fundamental en el desarrollo de cualquier proyecto de investigación, y en el caso del presente Trabajo de Fin de Máster (TFM), se considera desde diversas perspectivas. En primer lugar, el análisis de imágenes de resonancia magnética (MRI) para la detección temprana del Alzheimer busca mejorar la eficiencia en el diagnóstico médico, lo que puede traducirse en un uso más eficiente de los recursos del sistema de salud. La implementación de modelos de aprendizaje automático permite procesar grandes volúmenes de datos de manera rápida y precisa, reduciendo la necesidad de pruebas invasivas adicionales y disminuyendo los costos asociados.

Asimismo, se promueve la sostenibilidad mediante el uso de herramientas y plataformas de código abierto como Django, Visual Studio Code, Jupyter Notebook, y GitHub. Estas herramientas no solo son accesibles y gratuitas, sino que también fomentan la colaboración y el intercambio de conocimiento, reduciendo la duplicación de esfuerzos y promoviendo prácticas de desarrollo sostenible en la comunidad investigadora.

Impacto Ético-Social

Desde una perspectiva ético-social, este TFM aborda cuestiones críticas relacionadas con la privacidad y la confidencialidad de los datos de los pacientes. El uso de datos sensibles, como los provenientes de imágenes de resonancia magnética y registros clínicos, exige estrictas medidas de protección de datos para garantizar que la información de los sujetos se maneje de manera ética y segura. En este sentido, se han seguido rigurosamente las directrices establecidas por el conjunto de datos OASIS-1 y las normativas internacionales sobre privacidad de datos, asegurando el anonimato y la protección de la identidad de los participantes.

Además, el desarrollo de herramientas de diagnóstico basadas en inteligencia artificial debe considerar el impacto en la toma de decisiones médicas. Es crucial que estas herramientas se utilicen como soporte a la decisión clínica y no como reemplazo del juicio profesional, promoviendo así un enfoque responsable y ético en el uso de tecnologías avanzadas en la medicina.

Diversidad

En términos de diversidad, este proyecto busca contribuir a una representación más inclusiva y equitativa en la investigación médica. La selección de datos demográficos variados del conjunto OASIS-1 permite analizar cómo diferentes factores socioeconómicos y demográficos, como la edad, el género y el nivel educativo, pueden influir en la progresión de la enfermedad de Alzheimer. Este enfoque inclusivo es fundamental para desarrollar modelos predictivos que

sean efectivos y aplicables a una población diversa, asegurando que los beneficios del proyecto se extiendan a todos los grupos sociales.

1.4. Enfoque y método seguido

El desarrollo de la herramienta de clasificación de datos biomédicos se ha llevado a cabo utilizando un enfoque metodológico basado en el análisis de imágenes MRI y datos clínicos mediante técnicas de aprendizaje automático. Se han utilizado diversas herramientas de software y técnicas de procesamiento de datos, incluyendo:

1. Carga y Preprocesamiento de Datos:

- Recopilación de datos demográficos y clínicos de 436 sujetos del conjunto de datos OASIS-1.
- Conversión de columnas categóricas a valores numéricos.
- Imputación de datos faltantes utilizando el algoritmo K-Nearest Neighbors (KNN).
- Estandarización de datos numéricos.
- Normalización de imágenes MRI.

2. Reducción y Balanceo de Muestras:

- Reducción de muestras con CDR=0 para equilibrar la distribución de clases.
- Oversampling para aumentar las muestras con CDR=1 y CDR=2.

3. Data Augmentation para Imágenes MRI:

- Generación de nuevas muestras mediante técnicas de augmentación.
- Aplicación de transformaciones para simular variaciones naturales en las imágenes.

4. Implementación del Modelo de Clasificación:

- Desarrollo de modelos de Redes Neuronales Convolucionales (CNN) para la extracción de características de las imágenes MRI.
- Uso de clasificadores tradicionales como Support Vector Machines (SVM), Random Forest y Gradient Boosting para la clasificación final.

5. Desarrollo del Software:

- Implementación de suite de software que integra algoritmos seleccionados y proporciona una interfaz amigable para el usuario.
- Permite la carga de nuevas imágenes MRI y su clasificación automática.

1.5. Planificación del Trabajo

1.5.1. Tareas:

I. Investigación y selección de algoritmos (27 días).
   - Revisión de literatura y selección de algoritmos de aprendizaje automático para clasificación adecuados.

II. Recopilación y preprocesamiento de datos (52 días):
   - Obtención y preprocesamiento de los conjuntos de datos.
   - Imputación y estandarización de datos clínicos.
   - Normalización y balanceo de datos.

III. Entrenamiento de modelos (69 días).
   - Comparación de resultados y ajuste de parámetros.

IV. Desarrollo e implementación de la suite de software y la interfaz web (61 días).
   - Implementación de la interfaz web utilizando Django.
   - Integración de los modelos en la herramienta web.

V. Documentación y redacción de la memoria (78 días).

VI. Preparación de la presentación virtual (17 días).

1.5.2. Calendario:


GANTT project
2024
Nombre
Semana 1 4/3/24
Semana 2 11/3/24
Semana 3 18/3/24
Semana 4 25/3/24
Semana 5 1/4/24
Semana 6 8/4/24
Semana 7 15/4/24
Semana 8 22/4/24
Semana 9 29/4/24
Semana 10 6/5/24
Semana 11 13/5/24
Semana 12 20/5/24
Semana 13 27/5/24
Semana 14 3/6/24
Semana 15 10/6/24
Semana 16 17/6/24
Semana 17 24/6/24
Semana 18 1/7/24
5/3/24
Investigación y selección de algoritmos
[27 Día(s)]
Recopilación y preprocesamiento de datos
[52 Día(s)]
Entrenamiento de modelos
[69 Día(s)]
Desarrollo de la suite de software y la interfaz web
[61 Día(s)]
Documentación y redacción de la memoria
[78 Día(s)]
Preparación de la presentación
[17 Día(s)]

1.5.3. Hitos:

- Investigación inicial y selección de algoritmos adecuados.
- Preprocesamiento y limpieza de los datos del conjunto OASIS-1.
- Desarrollo y evaluación de los modelos de aprendizaje automático.
- Implementación de la herramienta de clasificación con una interfaz web.
- Documentación detallada del trabajo y preparación de la memoria final.
- Preparación y realización de la presentación virtual del TFM.

1.5.4. Análisis de riesgos

En la implementación de un proyecto de esta magnitud, es esencial anticipar y planificar posibles riesgos que puedan surgir a lo largo del desarrollo. A continuación se describen los principales riesgos identificados y las estrategias para mitigarlos:

Riesgos Técnicos

Calidad de los Datos:

Descripción: Los datos del conjunto OASIS-1 pueden tener inconsistencias, ruidos o valores atípicos que afecten la calidad del análisis.
Impacto: Alto. La calidad de los datos es fundamental para el rendimiento del modelo de aprendizaje automático.
Mitigación: Aplicar técnicas de preprocesamiento exhaustivas, incluyendo la normalización, eliminación de ruido y tratamiento de valores atípicos. Realizar una inspección manual de una muestra de los datos para asegurar su calidad.

Complejidad Computacional:

Descripción: La reducción de dimensionalidad y el entrenamiento de modelos de aprendizaje automático en grandes conjuntos de datos pueden requerir una capacidad computacional significativa.
Impacto: Alto. La complejidad computacional puede retrasar el desarrollo del proyecto.
Mitigación: Utilizar técnicas de reducción de dimensionalidad incremental para manejar grandes volúmenes de datos en lotes. Optimizar el código para mejorar la eficiencia computacional y considerar el uso de recursos en la nube si es necesario.

Desempeño de los Modelos:

Descripción: Los modelos de aprendizaje automático pueden no alcanzar la precisión esperada en la predicción de Alzheimer.
Impacto: Alto. Un bajo desempeño de los modelos comprometería el objetivo principal del proyecto.

Mitigación: Evaluar múltiples modelos y seleccionar aquellos con mejor desempeño mediante validación cruzada. Ajustar hiperparámetros y considerar técnicas avanzadas como el ensamblado de modelos (ensemble learning).
Riesgos de Gestión

Cumplimiento del Cronograma:

Descripción: El proyecto puede no adherirse al cronograma previsto debido a retrasos imprevistos.
Impacto: Medio. Los retrasos pueden afectar la entrega puntual del TFM.
Mitigación: Establecer hitos claros y realistas, con márgenes de tiempo para imprevistos. Realizar revisiones periódicas del progreso y ajustar el cronograma según sea necesario.

Disponibilidad de Recursos:

Descripción: La disponibilidad de recursos técnicos y humanos puede ser limitada.
Impacto: Medio. La falta de recursos puede ralentizar el progreso del proyecto.
Mitigación: Planificar con anticipación la necesidad de recursos y asegurarse de que estén disponibles cuando se necesiten. Considerar la posibilidad de colaborar con otros estudiantes o instituciones.

1.6. Breve sumario de productos obtenidos

- Memoria.
- Proyecto Django con la herramienta de clasificación compartida en repositorio GitHub.
- Presentación virtual

1.7. Breve descripción de los otros capítulos de la memoria

2. Estado del Arte

En este capítulo se revisa la literatura existente sobre las técnicas de diagnóstico de la enfermedad de Alzheimer, con un enfoque especial en el uso de imágenes de resonancia magnética (MRI) y aprendizaje automático. Se analizan estudios previos que han utilizado redes neuronales convolucionales (CNN) y otros algoritmos de clasificación, destacando los avances y desafíos en la predicción temprana de la EA. También se discuten las metodologías más efectivas y los conjuntos de datos utilizados en investigaciones recientes.

3. Materiales y Métodos

Este capítulo detalla los materiales y métodos empleados en el desarrollo de la herramienta de clasificación. Incluye la descripción del conjunto de datos OASIS-1 y las herramientas de software utilizadas, como Django, Visual Studio Code, Jupyter Notebook, GitHub e ITK-SNAP. Se describe el proceso de preprocesamiento de datos, que abarca la normalización y redimensionamiento de las imágenes MRI, la imputación de datos faltantes y la estandarización de

los datos clínicos. Además, se explica la metodología utilizada para entrenar y evaluar los modelos de aprendizaje automático.

4. Resultados
En este capítulo se presentan los resultados obtenidos a partir del entrenamiento y evaluación de los modelos de CNN y otros algoritmos de clasificación. Se incluyen las métricas de precisión para los modelos GFC y Masked GFC, así como los resultados de los modelos adicionales (SVM, Random Forest, Gradient Boosting). También se analizan los resultados del preprocesamiento de las imágenes y los datos clínicos, y se discuten las limitaciones y desafíos encontrados durante el desarrollo del proyecto.

5. Discusión y Conclusiones
Este capítulo ofrece una discusión detallada de los resultados obtenidos, comparándolos con estudios previos y analizando las posibles razones detrás de las diferencias observadas. Se examinan las implicaciones de los hallazgos para el diagnóstico temprano de la EA y se proponen mejoras futuras para los modelos y la herramienta de clasificación. También se reflexiona sobre las limitaciones del estudio y las oportunidades para investigaciones futuras.

En conclusiones se resumen los hallazgos más importantes del estudio, destacando la efectividad de las técnicas de aprendizaje automático en la predicción de la EA a partir de imágenes MRI. Se concluye con una reflexión sobre el impacto potencial de la herramienta desarrollada en el ámbito clínico y se subrayan las áreas clave que requieren más investigación y desarrollo.

7. Bibliografía
Este capítulo recopila todas las referencias bibliográficas utilizadas a lo largo del trabajo. Incluye estudios, artículos y libros que han sido citados en los capítulos anteriores, proporcionando una base sólida de fuentes académicas y científicas que respaldan la investigación realizada.

8. Apéndices
Incluye el código utilizado en el proyecto y el enlace al repositorio de GitHub donde se encuentra alojado. El apéndice proporciona un acceso detallado al código fuente y a las implementaciones específicas utilizadas para el preprocesamiento de datos, el entrenamiento de los modelos y el desarrollo de la interfaz web. Este material adicional complementa el contenido principal de la memoria y permite una mayor reproducibilidad y transparencia en la investigación.

2. Estado del arte

En esta sección se revisan las tecnologías y métodos actuales en el campo del análisis de datos biomédicos mediante técnicas de aprendizaje automático, con un enfoque especial en la predicción de la enfermedad de Alzheimer a partir de imágenes de resonancia magnética (MRI).

2.1 Revisión de Tecnologías y Métodos Actuales

El análisis de datos biomédicos ha avanzado significativamente gracias a la aplicación de técnicas de aprendizaje automático. Estas técnicas han permitido mejorar la precisión y la eficiencia en la clasificación de datos biológicos complejos. A continuación, se describen los métodos más relevantes en este ámbito:

- Máquinas de Vectores de Soporte (SVM): Las SVM son populares debido a su capacidad para manejar datos de alta dimensionalidad, como las imágenes de MRI. Este método busca encontrar el hiperplano que mejor separa las clases de datos en un espacio de características de alta dimensión. Las SVM se han utilizado ampliamente en estudios genómicos y proteómicos por su efectividad en la clasificación precisa de datos complejos (Hastie, Tibshirani, & Friedman, 2009).

- K-Nearest Neighbors (KNN): El algoritmo KNN es conocido por su simplicidad y eficacia en la clasificación de datos biológicos menos complejos. Funciona asignando una clase a un punto de datos en función de las clases de sus K vecinos más cercanos. Este método es particularmente útil en conjuntos de datos donde la estructura de los datos es simple y bien definida (Bishop, 2006).

- Árboles de Decisión y Random Forests: Los árboles de decisión son modelos de clasificación que segmentan el espacio de características en regiones homogéneas basándose en criterios de división. Los Random Forests, una extensión de los árboles de decisión, construyen múltiples árboles y combinan sus predicciones para mejorar la precisión y reducir el sobreajuste (Breiman, 2001).

- Redes Neuronales y Aprendizaje Profundo: Las redes neuronales, especialmente las redes neuronales profundas (DNN) y las redes neuronales convolucionales (CNN), han revolucionado el campo del análisis de imágenes. Estos modelos pueden aprender características complejas directamente de las imágenes sin necesidad de ingeniería de características manual. Las CNN, en particular, han mostrado un rendimiento sobresaliente en la clasificación de imágenes de MRI para la detección de enfermedades como el Alzheimer (Goodfellow, Bengio, & Courville, 2016).

2.2. Identificación de Brechas y Necesidades no Satisfechas

A pesar de los avances significativos, existen limitaciones importantes en los métodos actuales:

- Precisión y Escalabilidad: Los modelos actuales pueden tener dificultades para manejar conjuntos de datos extremadamente grandes o heterogéneos, como los encontrados en estudios de metagenómica. La precisión y la eficiencia de estos modelos pueden verse comprometidas cuando se enfrentan a datos de alta dimensionalidad y complejidad (Murphy, 2012).

- Eficiencia en Tiempo de Ejecución: La rapidez en la toma de decisiones es crucial en entornos clínicos. Sin embargo, muchos algoritmos actuales no son lo suficientemente rápidos para ser utilizados en situaciones en las que se requiere una respuesta inmediata (Goodfellow et al., 2016).

2.3. Justificación de la Importancia del Trabajo

Este proyecto busca abordar las limitaciones mencionadas mediante el desarrollo de un algoritmo de aprendizaje profundo específicamente adaptado para datos biológicos. Se espera que este enfoque mejore tanto la precisión como la eficiencia en la clasificación de datos biomédicos, facilitando análisis más rápidos y precisos que puedan integrarse en aplicaciones clínicas y de investigación biomédica (Aggarwal, 2018).

2.4. Formulación de Hipótesis y Objetivos

Hipótesis: La implementación de redes neuronales profundas adaptadas específicamente para datos biológicos resultará en una mejora sustancial en la precisión y eficiencia en comparación con los métodos convencionales.

Objetivos Específicos:

- Desarrollar un modelo de red neuronal que se adapte eficazmente a las características únicas de los datos biológicos.

- Comparar la eficacia de este modelo con algoritmos tradicionales en términos de precisión y tiempo de ejecución.

- Evaluar la aplicabilidad del modelo en diferentes tipos de datos biológicos para asegurar su versatilidad y escalabilidad.

3. Materiales y métodos

En esta sección se describen los materiales y métodos utilizados para desarrollar la herramienta de clasificación de datos biomédicos, específicamente para la predicción de la enfermedad de Alzheimer mediante el análisis de imágenes de resonancia magnética (MRI). El desarrollo de este

proyecto se ha llevado a cabo utilizando diversas herramientas de software y técnicas de procesamiento de datos.

3.1. Materiales

Herramientas de Desarrollo:

Django: Framework web utilizado para desarrollar la interfaz de usuario de la herramienta.
Visual Studio Code: Entorno de desarrollo integrado (IDE) utilizado para la programación y el desarrollo del proyecto.
Jupyter Notebook: Herramienta utilizada para el análisis exploratorio de datos, desarrollo de modelos y visualización de resultados.
GitHub: Plataforma de control de versiones utilizada para gestionar el código fuente y la colaboración.
ITK-SNAP: Software utilizado para la segmentación y visualización de imágenes médicas.

Conjunto de Datos:

El conjunto de datos utilizado en este proyecto proviene de la serie de estudios de imágenes de acceso abierto OASIS-1 (Open Access Series of Imaging Studies). Se trata de un conjunto de datos clínicos y demográficos de 436 sujetos, incluyendo adultos jóvenes, adultos de mediana edad y adultos mayores, tanto sanos como diagnosticados con demencia (Marcus et al., 2007). Las variables incluidas en el análisis son las siguientes:

- ID: Identificador único del sujeto.
- M/F: Género del sujeto.
- Hand: Dominancia manual.
- Age: Edad del sujeto.
- Educ: Años de educación.
- SES: Estado socioeconómico.
- MMSE: Puntuación en el Mini-Mental State Examination.
- CDR: Clinical Dementia Rating.
- eTIV: Volumen intracraneal estimado.
- nWBV: Volumen cerebral normalizado.
- ASF: Factor de escala ajustado.
- Delay: Retraso en la obtención de la imagen.

Tipos de Imágenes Usados

En este estudio, se han utilizado dos tipos específicos de imágenes MRI provenientes del conjunto de datos OASIS-1:
- Imagen registrada al atlas corregida por campo de ganancia (t88_gfc):

Estas imágenes están registradas a un atlas estándar y han sido corregidas por el campo de ganancia, lo que mejora la uniformidad de las intensidades en la

imagen. Este tipo de corrección es crucial para asegurar que las variaciones en las intensidades de los píxeles reflejen verdaderas diferencias anatómicas o patológicas y no artefactos técnicos.

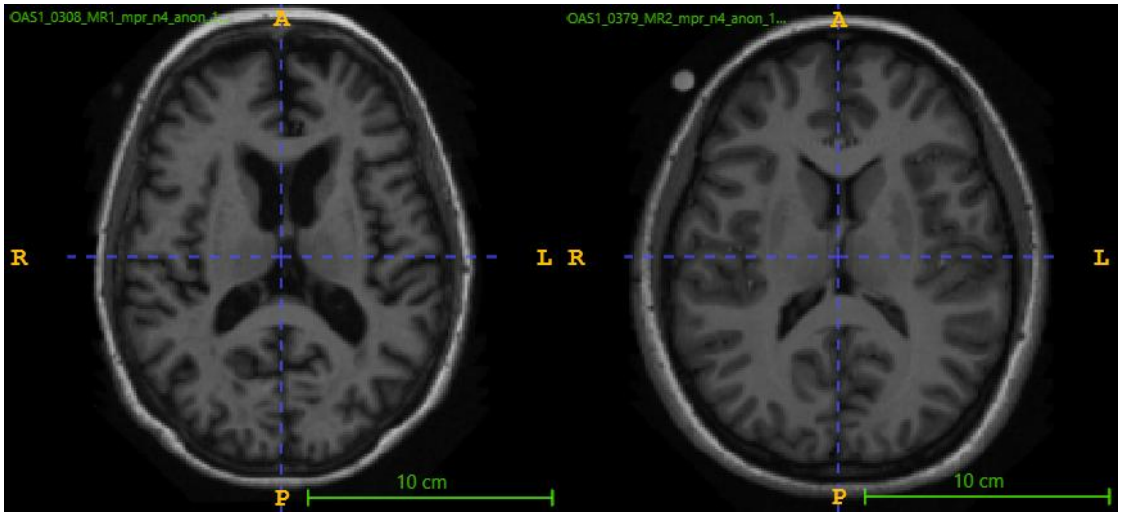

Imagen 1. Imagen registrada al atlas corregida por campo de ganancia. Imagen de sujeto con demencia (izquierda) e imagen de sujeto sin demencia (derecha) (Open Access Series of Imaging Studies).

- Versión enmascarada del cerebro de la imagen registrada al atlas (t88_masked_gfc_fseg):

Estas imágenes son versiones enmascaradas del cerebro de las imágenes t88_gfc registradas al atlas. El enmascaramiento del cerebro implica la eliminación de todas las estructuras no cerebrales de la imagen, lo que permite un análisis más enfocado en el tejido cerebral. La segmentación de estas imágenes asegura que las características anatómicas del cerebro sean las principales contribuyentes a las variaciones en las intensidades de los píxeles, eliminando posibles confusiones causadas por otras estructuras.

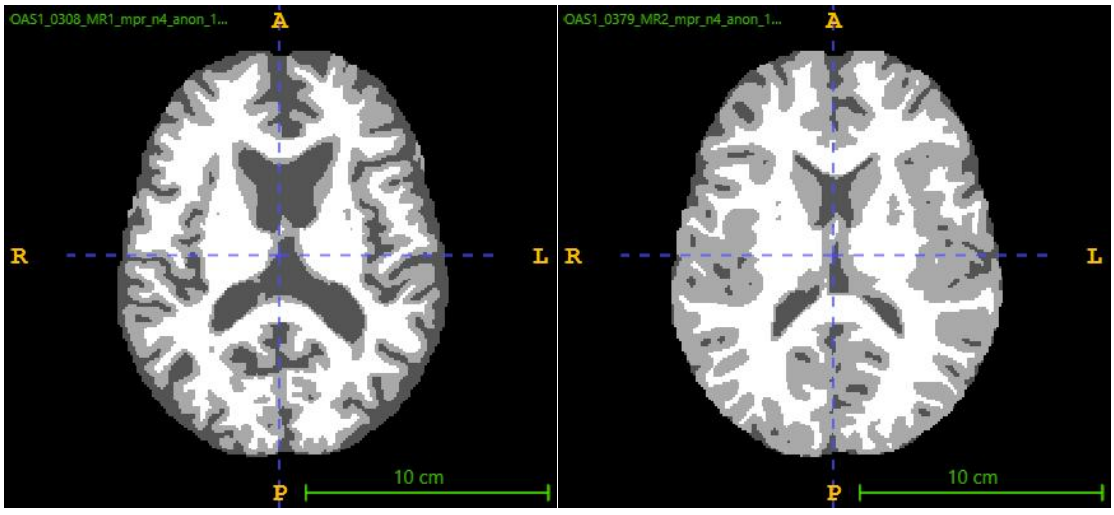

Imagen 2. Versión enmascarada del cerebro de la imagen registrada al atlas. Imagen de sujeto con demencia (izquierda) e imagen de sujeto sin demencia (derecha) (Open Access Series of Imaging Studies).

3.2. Métodos

Tratamiento de Datos

El proceso de tratamiento de datos comienza con la carga de los mismos desde un archivo Excel utilizando la biblioteca pandas (McKinney, 2010). Este archivo contiene información demográfica y clínica de los sujetos involucrados en el estudio, abarcando variables críticas para el análisis, tales como 'Educ' (años de educación), 'SES' (estatus socioeconómico) y 'MMSE' (Mini-Mental State Examination).

Conversión de Columnas Categóricas

Una vez cargados los datos, se procede a convertir las columnas categóricas ('M/F' para el género y 'Hand' para la dominancia manual) a valores numéricos. Esta conversión es necesaria para que las técnicas de imputación y estandarización puedan procesar correctamente estas columnas. Se utiliza `LabelEncoder` de la biblioteca `sklearn` para transformar las categorías textuales en valores numéricos (Pedregosa et al., 2011).

Imputación de Datos Faltantes

La imputación de datos faltantes es una etapa crucial para asegurar la integridad del conjunto de datos. En este estudio, se emplea el algoritmo K-Nearest Neighbors (KNN) para imputar valores faltantes en las columnas 'Educ', 'SES' y 'MMSE' (Troyanskaya et al., 2001).

- Preparación de los Datos para la Imputación: Se seleccionan y combinan datos completos de sujetos con puntuación CDR=0 y sujetos de control, formando un conjunto de datos de referencia robusto para la imputación.

- Aplicación del Imputador: Se configura el `KNNImputer` con k=4 y k=5, según los datos a imputar, calculando los valores faltantes utilizando la media de los valores de los vecinos más cercanos.

- Creación de un DataFrame con los Datos Imputados: Los datos imputados se convierten nuevamente en un DataFrame para facilitar la manipulación posterior.

- Actualización de los Datos Originales: Los valores imputados se integran de nuevo en el conjunto de datos original.

Estandarización de Datos

La estandarización de los datos es esencial para garantizar que todas las variables tengan una escala comparable.

- Extracción de las Columnas No Numéricas: Se extraen las columnas categóricas y el identificador único (ID) del conjunto de datos.

- Aplicación del Estandarizador: Se utiliza `StandardScaler` de `sklearn` para estandarizar las columnas numéricas restantes, ajustando los datos para que tengan una media de 0 y una desviación estándar de 1 (Pedregosa et al., 2011).

- Reintegración de Columnas No Numéricas: Las columnas categóricas y el identificador se añaden nuevamente al conjunto de datos. Se utiliza codificación one-hot para las variables categóricas como 'CDR'.

Normalización de Datos de Imágenes

El proceso de normalización de las imágenes de resonancia magnética (MRI) es un paso crucial en la preparación de los datos para su análisis mediante técnicas de aprendizaje automático.

La función `load_and_preprocess_image` es responsable de cargar una imagen MRI desde el archivo utilizando `nibabel` (Brett et al., 2020) y luego normalizar los datos. La normalización se realiza transformando las intensidades de los píxeles de la imagen a un rango entre 0 y 1. Esto se consigue restando el valor mínimo de los datos y dividiendo por el rango de valores. Este paso es crucial para garantizar que todas las imágenes tengan una escala uniforme.

La función `save_preprocessed_image` guarda las imágenes normalizadas en un formato específico (.nii) utilizando nuevamente `nibabel` (Brett et al., 2020). Se especifica una matriz de identidad como el afine para asegurar que las coordenadas espaciales de las imágenes se mantengan consistentes.

Preprocesamiento de Directorios Completo

La función preprocess_images itera sobre todos los archivos en los directorios crudos, aplica la normalización y guarda las imágenes procesadas en los directorios de destino. Este procedimiento asegura que todas las imágenes MRI sean normalizadas de manera uniforme.

La normalización de datos es un paso crítico en el procesamiento de imágenes de resonancia magnética por varias razones:

- Consistencia en la Escala de Intensidades: Las imágenes MRI pueden variar significativamente en sus intensidades. Normalizar estas intensidades a un rango estándar asegura que todas las imágenes sean comparables entre sí.

- Mejora del Rendimiento del Modelo: Los algoritmos de aprendizaje automático son sensibles a las escalas de las características de entrada. La normalización ayuda a mejorar la convergencia y la estabilidad del entrenamiento de los modelos.

- Reducción de Sesgos: La normalización ayuda a reducir los sesgos introducidos por las variaciones en las intensidades de las imágenes, permitiendo que los modelos aprendan patrones representativos de las condiciones clínicas.

- Facilitación de la Interpretación Visual: La normalización facilita la interpretación visual de las imágenes, ya que las variaciones en las intensidades reflejan diferencias anatómicas o patológicas en lugar de artefactos técnicos.

Reducción y Balanceo de Muestras

En el análisis de datos clínicos y de resonancia magnética (MRI) para el diagnóstico de la enfermedad de Alzheimer, es común encontrar una desproporción en la representación de diferentes clases de datos. En este estudio, se observó una sobrerepresentación de muestras con una puntuación de CDR=0, indicando la ausencia de demencia. Para abordar esta desproporción, se implementó una técnica de reducción de muestras para equilibrar la distribución de clases.

Primero, se cargaron los datos demográficos y clínicos desde un archivo Excel (McKinney, 2010). Luego, se identificaron las muestras con CDR=0 y se seleccionaron aleatoriamente 80 de estas muestras para su eliminación. Este procedimiento se llevó a cabo utilizando las capacidades de filtrado y muestreo aleatorio de la biblioteca pandas. Las muestras seleccionadas se eliminaron del DataFrame original, y el conjunto de datos resultante se guardó en un nuevo archivo Excel. Este paso asegura que las clases de datos estén más equilibradas, mejorando la precisión y la robustez de los modelos de aprendizaje automático que se entrenarán posteriormente.

Además, se reorganizaron las imágenes correspondientes a las muestras restantes. Utilizando patrones de búsqueda basados en expresiones regulares, las imágenes relacionadas con los sujetos seleccionados se copiaron a directorios específicos para un manejo más eficiente y organizado.

Oversampling de Datos para Aumentar Muestras con CDR=1 y CDR=2

Para complementar la reducción de muestras con CDR=0, se implementó una técnica de oversampling para aumentar la cantidad de muestras con puntuaciones CDR=1 y CDR=2, indicando niveles más avanzados de demencia. Esta técnica es crucial para asegurar que los modelos de aprendizaje automático tengan suficiente información para aprender a identificar estas clases menos representadas.

Se cargaron los datos filtrados desde un archivo Excel y se identificaron los sujetos con CDR=2. Para estos sujetos, se generaron nuevas muestras mediante un proceso de aumentación, que consiste en replicar las muestras existentes y asignarles nuevos identificadores únicos. Similarmente, se generaron nuevas muestras para sujetos con CDR=1.

Este proceso de aumentación fue realizado mediante una función personalizada que replica las filas seleccionadas del DataFrame original, asegurando que los nuevos identificadores sean únicos para cada muestra aumentada. Las muestras aumentadas se combinaron con el conjunto de datos original, resultando en un conjunto de datos balanceado que se guardó en un nuevo archivo Excel.

Estas técnicas de reducción y balanceo de datos son fundamentales para mitigar el sesgo introducido por la desproporción de clases, mejorando así la capacidad del modelo de aprendizaje automático para generalizar y proporcionar diagnósticos precisos en el contexto de la enfermedad de Alzheimer.

Data Augmentation para Imágenes de Resonancia Magnética (MRI)

El Data Augmentation es una técnica utilizada para aumentar el tamaño del conjunto de datos de entrenamiento mediante la creación de nuevas muestras a partir de las existentes. En este estudio, se empleó Data Augmentation para generar imágenes adicionales de resonancia magnética (MRI) correspondientes a sujetos con puntuaciones de CDR=1 y CDR=2, con el objetivo de equilibrar las clases y mejorar la capacidad del modelo para generalizar.

Se configura el entorno y se carga el conjunto de datos desde un archivo Excel, filtrando las muestras con puntuaciones CDR=1 y CDR=2. Estas muestras representan diferentes grados de demencia y son cruciales para el entrenamiento de un modelo robusto.

Se define una serie de transformaciones utilizando la biblioteca MONAI, que es especializada en el procesamiento de imágenes médicas (Cardoso et al., 2022). Las transformaciones incluyen la escalación de intensidades, volteos aleatorios, rotaciones, traslaciones y cortes espaciales. Estas transformaciones son aplicadas para simular variaciones naturales que podrían encontrarse en diferentes adquisiciones de imágenes.

Para cada sujeto, se cargan las imágenes MRI desde el disco. Se aplican las transformaciones definidas para generar un número específico de imágenes aumentadas (20 para sujetos con CDR=2 y 1 para sujetos con CDR=1). Este proceso asegura que las nuevas imágenes mantengan la estructura y características originales pero con variaciones adicionales que enriquecen el conjunto de datos.

Las imágenes aumentadas se guardan en directorios de salida específicos. Se mantiene la relación con los sujetos originales mediante la inclusión de identificadores únicos en los nombres de los archivos aumentados, facilitando así su trazabilidad.

El Data Augmentation es fundamental por varias razones:

- Aumento del Tamaño del Conjunto de Datos: Al generar nuevas muestras a partir de las existentes, se incrementa el tamaño del conjunto de datos de entrenamiento, lo que puede mejorar significativamente el rendimiento del modelo al proporcionar más ejemplos de entrenamiento.

- Equilibrio de Clases: El Data Augmentation ayuda a mitigar el problema de clases subrepresentadas (en este caso, CDR=1 y CDR=2), lo que conduce a un modelo más equilibrado y menos sesgado.

- Mejora de la Generalización: Las transformaciones aplicadas durante el Data Augmentation introducen variaciones en las imágenes que simulan las condiciones reales de adquisición de datos. Esto ayuda al modelo a reconocer la enfermedad bajo diversas condiciones.

- Reducción del Sobreajuste: Al proporcionar más variabilidad en el conjunto de datos de entrenamiento, el Data Augmentation puede ayudar a reducir el riesgo de sobreajuste, donde el modelo se adapta demasiado a las características específicas del conjunto de entrenamiento y falla al generalizar a nuevos datos.

En resumen, la técnica de Data Augmentation aplicada en este estudio no solo equilibra las clases de datos sino que también mejora la robustez y la capacidad de generalización del modelo de aprendizaje automático, facilitando un diagnóstico más preciso y fiable de la enfermedad de Alzheimer.

Implementación del Modelo de Clasificación utilizando Redes Neuronales Convolucionales (CNN) y Clasificadores Tradicionales

El objetivo de este proyecto es desarrollar un modelo de clasificación para predecir la enfermedad de Alzheimer utilizando datos de imágenes de resonancia magnética (MRI) procesadas y enmascaradas, junto con datos clínicos. Se empleó un enfoque híbrido que combina redes neuronales convolucionales (CNN) para la extracción de características con clasificadores tradicionales como Support Vector Machines (SVM), Random Forest y Gradient Boosting para la clasificación final.

Configuración y Carga de Datos Clínicos

El proceso comienza con la configuración del entorno de trabajo y la carga de los datos clínicos y demográficos desde un archivo Excel. Estos datos incluyen información crucial como la puntuación CDR (Clinical Dementia Rating), que es fundamental para la clasificación (Marcus et al., 2007).

Preprocesamiento de Imágenes MRI

Las imágenes MRI se cargaron desde directorios específicos y se preprocesaron para normalizar sus intensidades y redimensionarlas a una forma uniforme de 64x64x64 píxeles. La normalización se realizó transformando las intensidades de los píxeles a un rango entre 0 y 1, y luego se redimensionaron las imágenes para asegurar que todas tuvieran la misma forma. Este preprocesamiento es crucial para asegurar la uniformidad de los datos de entrada al modelo (Brett et al., 2020).

Construcción del Modelo de Red Neuronal Convolucional (CNN)

Se construyeron dos modelos CNN, uno para las imágenes GFC procesadas y otro para las imágenes enmascaradas. La arquitectura de la CNN utilizada incluye varias capas de convolución 3D, max-pooling y capas densas totalmente conectadas. Se utilizaron capas de dropout para prevenir el sobreajuste durante el entrenamiento (LeCun et al., 2015).

Entrenamiento del Modelo

Los datos se dividieron en conjuntos de entrenamiento y prueba, utilizando un 20% de los datos para la evaluación del modelo. Se entrenaron los modelos CNN con los datos de entrenamiento, utilizando una pequeña parte del conjunto de entrenamiento para la validación. Durante el entrenamiento, se aplicaron técnicas de regularización como dropout para mejorar la generalización del modelo (Srivastava et al., 2014).

Evaluación del Modelo

El modelo se evaluó utilizando el conjunto de prueba. La precisión obtenida en este conjunto proporciona una medida de la capacidad del modelo para generalizar a nuevos datos. Esta evaluación es crucial para validar el rendimiento del modelo en condiciones no vistas durante el entrenamiento.

Clasificadores Utilizados

Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales (CNN) son un tipo de red neuronal profunda especialmente diseñada para procesar datos que tienen una estructura de cuadrícula, como las imágenes. Las CNN son particularmente eficaces en el reconocimiento de patrones y características espaciales, lo que las hace ideales para tareas de análisis de imágenes (LeCun et al., 2015).

Fortalezas de las CNN	Debilidades de las CNN
Capacidad para capturar patrones espaciales complejos	Requieren grandes cantidades de datos para entrenarse
Alto rendimiento en tareas de clasificación de imágenes	Elevada demanda computacional

Automatización en la extracción de características	Difícil de interpretar comparado con modelos más simples
Robustez ante transformaciones y distorsiones	Susceptibles a sobreajuste si no se regularizan adecuadamente

Support Vector Machines (SVM)

Las Support Vector Machines (SVM) son algoritmos de aprendizaje supervisado utilizados tanto para clasificación como para regresión. Las SVM buscan encontrar el hiperplano óptimo que maximiza el margen entre las clases en un espacio de alta dimensión. Este algoritmo es especialmente útil en problemas de clasificación binaria y multiclase (Cortes y Vapnik, 1995).

Fortalezas del SVM	Debilidades del SVM
Eficaz en espacios de alta dimensionalidad	Rendimiento pobre en conjuntos de datos muy grandes
Funciona bien con un margen claro de separación	Sensible al ruido en los datos
Utiliza una función de kernel para manejar datos no lineales	Requiere ajuste cuidadoso de parámetros y selección del kernel

Random Forest

El Random Forest es un algoritmo de aprendizaje supervisado que se basa en la construcción de múltiples árboles de decisión durante el entrenamiento y la salida de la clase que es el modo de las clases de salida de los árboles individuales. Es conocido por su capacidad para manejar grandes conjuntos de datos con alta dimensionalidad y su robustez frente al sobreajuste (Breiman, 2001).

Fortalezas Random Forest	Debilidades Random Forest
Maneja bien la varianza y el sobreajuste	Puede ser computacionalmente intensivo
Robusto a outliers y ruido	Menos interpretable que un único árbol de decisión
Capaz de manejar grandes conjuntos de datos con muchas variables	Requiere un mayor tiempo de entrenamiento y predicción

Gradient Boosting

El Gradient Boosting es un algoritmo de aprendizaje supervisado que se utiliza tanto para clasificación como para regresión. Este algoritmo construye modelos

de forma secuencial, y cada nuevo modelo intenta corregir los errores cometidos por los modelos anteriores. Utiliza una técnica de boosting, donde los modelos se combinan para mejorar la precisión predictiva (Friedman, 2001).

Fortalezas del Gradient Boosting	Debilidades del Gradient Boosting
Alta precisión en una variedad de problemas	Muy sensible a los valores atípicos
Maneja bien datos no lineales y relaciones complejas	Puede sobreajustar si no se configura correctamente
Flexible con varios tipos de funciones de pérdida y potenciadores	Requiere más tiempo de entrenamiento y ajuste de hiperparámetros

Configuración del Entorno

Para la implementación de la herramienta, se creó un entorno virtual de Python para gestionar las dependencias del proyecto de manera aislada. Se utilizó Python 3.11 y Django 5.0.6 como base del framework web (Django Software Foundation, 2023). Las bibliotecas adicionales incluyeron pandas para la manipulación de datos (McKinney, 2010), numpy para operaciones numéricas (Harris et al., 2020), scikit-learn y scikit-image para el preprocesamiento de datos y manipulación de imágenes (Pedregosa et al., 2011; van der Walt et al., 2014), nibabel para la carga de imágenes MRI en formato NIfTI (Brett et al., 2020), y tensorflow para el uso de modelos de aprendizaje profundo (Abadi et al., 2016).

Desarrollo del Backend

El backend de la aplicación se construyó sobre Django. Se definieron varias vistas para manejar la carga y procesamiento de archivos, así como la lógica de predicción. En particular, se implementó una vista principal classify_image que permite a los usuarios subir una imagen MRI y un archivo Excel con datos clínicos. Los datos clínicos deben incluir las siguientes columnas: ID, M/F, Hand, Age, Educ, SES, MMSE, eTIV, nWBV, ASF, y Delay.

Se crearon formularios personalizados (UploadForm) para facilitar la carga de archivos a través de la interfaz web. Los modelos de aprendizaje automático, previamente entrenados y guardados en formato .h5 y .pkl, se cargan dinámicamente según la selección del usuario. Se diseñó un modelo de datos que permite preprocesar tanto las imágenes MRI como los datos clínicos antes de realizar la predicción.

Preprocesamiento de Imágenes y Datos Clínicos

Las imágenes MRI se preprocesan utilizando la biblioteca `nibabel` para la carga y `scikit-image` para la normalización y redimensionamiento a un tamaño

estándar de 64x64x64 píxeles. Los datos clínicos se cargan desde un archivo Excel y se convierten a un formato adecuado mediante pandas. Se utilizaron codificadores (`LabelEncoder`) para transformar las variables categóricas (M/F y Hand) y `StandardScaler` para normalizar las variables continuas (Pedregosa et al., 2011).

Integración de Modelos de Aprendizaje Automático

Los modelos de aprendizaje automático, se cargan en la aplicación y se utilizan para realizar predicciones. Estos modelos incluyen varias arquitecturas de redes neuronales convolucionales (CNN) entrenadas para detectar patrones indicativos de Alzheimer en imágenes MRI (LeCun et al., 2015). El proceso de predicción combina los datos preprocesados de la imagen y los datos clínicos, y devuelve una predicción basada en el modelo seleccionado.

Desarrollo del Frontend

El frontend de la aplicación se desarrolló utilizando las plantillas de Django, proporcionando una interfaz de usuario sencilla y fácil de usar. Las plantillas HTML (upload.html y result.html) permiten a los usuarios cargar archivos y ver los resultados de las predicciones. Se emplearon formularios HTML estándar con soporte para la carga de archivos y selección de opciones (Django Software Foundation, 2023).

4. Resultados

La visualización de datos previa al estudio indica que la variabilidad en las puntuaciones de CDR y MMSE aumenta con la edad, especialmente en los grupos de 61-80 y 81-100 años, reflejando una mayor prevalencia de demencia y deterioro cognitivo en estos grupos. Este análisis resalta la importancia de la detección temprana y el monitoreo continuo de la salud cognitiva en las personas mayores para la gestión efectiva de la enfermedad de Alzheimer y otras formas de demencia.

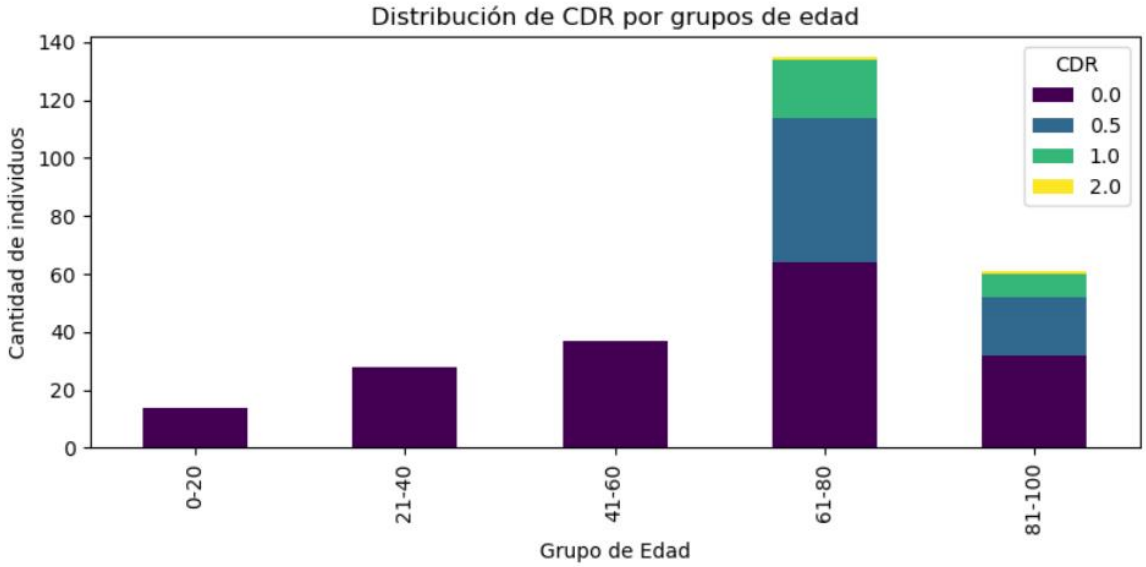

Ilustración 1. Distribución de CDR por grupos de edad

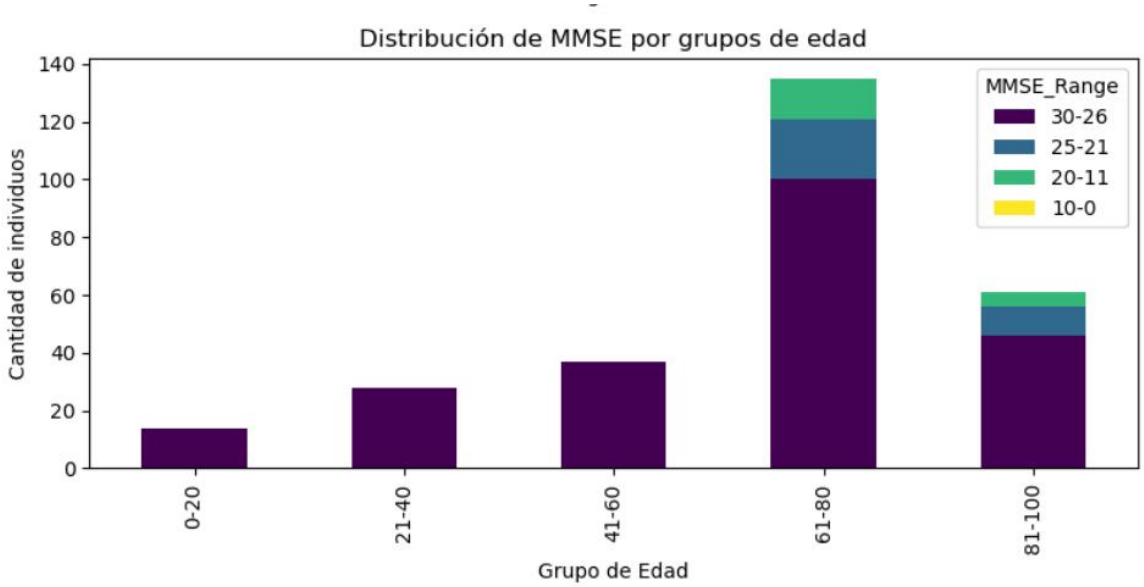

Ilustración 2. Distribución de MMSE por grupos de edad

La figura presenta los resultados del primer procesamiento de datos, mostrando la distribución de las puntuaciones de la Clinical Dementia Rating (CDR) y del Mini-Mental State Examination (MMSE) entre diferentes grupos de edad. Estas visualizaciones son fundamentales para entender cómo se distribuye la demencia y el deterioro cognitivo en la población estudiada.

Distribución de CDR por Grupos de Edad
0-20 años y 21-40 años: Todos los individuos tienen una puntuación CDR de 0.0, indicando ausencia de demencia.

41-60 años: Aumenta el número de individuos, pero todos tienen una puntuación CDR de 0.0.

61-80 años: Se observa mayor variabilidad con individuos presentando puntuaciones CDR de 0.0, 0.5, 1.0, y 2.0.

81-100 años: Similar al grupo anterior, con una notable variabilidad en las puntuaciones CDR.

Distribución de MMSE por Grupos de Edad

0-20 años y 21-40 años: Todos los individuos puntúan en el rango de 30-26, indicando un estado cognitivo normal.

41-60 años: La mayoría en el rango de 30-26.

61-80 años: Mayor variabilidad con puntuaciones en los rangos de 30-26, 25-21, 20-11, y algunos en 10-0.

81-100 años: Similar al grupo de 61-80 años, con una distribución variada en las puntuaciones MMSE.

Los resultados de la normalización de las imágenes GFC y Masked GFC son los siguientes:

- GFC:

Valor mínimo global: 0.0
Valor máximo global: 1.0
Valor medio global: 0.13191698404663557
Desviación estándar global: 0.018513963093076374
Valores mínimos: media = 0.0, std = 0.0
Valores máximos: media = 1.0, std = 0.0
Valores medios: media= 0.13191698404663557, std = 0.016928266921471254
Desviación estándar de los valores: media = 0.14932176140550263, std = 0.018513963093076374

- Masked GFC:

Valor mínimo global: 0.0
Valor medio global: 0.18198737610880666
Desviación estándar global: 0.0072490979730801585
Valores mínimos: media = 0.0, std = 0.0
Valores máximos: media = 1.0, std = 0.0
Valores medios: media = 0.18198737610880666,std = 0.006170785241675499
Desviación estándar de los valores: media = 0.33081092870212336, std = 0.0072490979730801585

Los resultados de la normalización muestran que las imágenes GFC y Masked GFC fueron exitosamente normalizadas para tener intensidades de píxeles en un rango entre 0 y 1. Esto es crucial para la consistencia en el análisis posterior mediante técnicas de aprendizaje automático.

- Imágenes GFC:

Uniformidad: Las imágenes GFC tienen un valor medio global de 0.1319 y una desviación estándar global de 0.0185, indicando una distribución relativamente uniforme de las intensidades de los píxeles después de la normalización.

Valores extremos: La media de los valores mínimos es 0.0 y la media de los valores máximos es 1.0, confirmando que las intensidades fueron correctamente escaladas.

- Imágenes Masked GFC:

Variabilidad: Las imágenes Masked GFC tienen un valor medio global más alto de 0.1820 y una desviación estándar global de 0.0072, sugiriendo una mayor variabilidad en las intensidades de los píxeles.

Valores extremos: Similar a las imágenes GFC, las imágenes Masked GFC también tienen una media de los valores mínimos de 0.0 y una media de los valores máximos de 1.0, asegurando la correcta normalización.

La matriz de correlación entre las variables del conjunto de datos proporciona información crucial sobre cómo estas variables están relacionadas entre sí. A continuación, se comentan las relaciones más destacadas:

- CDR y MMSE (r = -0.759): Hay una fuerte correlación negativa entre la puntuación CDR y la puntuación MMSE. Esto indica que a medida que la puntuación CDR, que mide el nivel de demencia, aumenta, la puntuación MMSE, que mide el estado cognitivo, tiende a disminuir. Esta relación es esperada, ya que un mayor nivel de demencia está asociado con un peor rendimiento cognitivo.

- CDR y nWBV (r = -0.524): Existe una correlación negativa moderada entre CDR y el volumen cerebral normalizado (nWBV). Esto sugiere que a medida que la demencia progresa (mayor CDR), el volumen cerebral tiende a disminuir, lo cual es consistente con los efectos neurodegenerativos del Alzheimer.

- MMSE y nWBV (r = 0.489): Existe una correlación positiva moderada entre MMSE y el volumen cerebral normalizado (nWBV). Esto sugiere que un mayor volumen cerebral está asociado con mejores puntuaciones en el MMSE, indicando mejor estado cognitivo.

- CDR y Age (r = 0.347): Hay una correlación positiva entre la edad y la puntuación CDR, indicando que los individuos mayores tienden a tener

puntuaciones de CDR más altas, reflejando una mayor prevalencia de demencia en edades avanzadas.

El proceso de entrenamiento de los modelos CNN mostró una mejora constante en la precisión de entrenamiento a lo largo de las épocas. Para el modelo GFC, la precisión final alcanzó un notable 72.89%, mientras que para el modelo Masked GFC, la precisión se situó en un 52.58%. Esta diferencia en los resultados podría atribuirse a las características intrínsecas de las imágenes enmascaradas, que podrían no ser tan representativas o informativas como las imágenes GFC para la tarea de clasificación. La precisión de validación, aunque prometedora, fue más variable, oscilando entre el 50% y el 60% para el modelo GFC y manteniéndose alrededor del 55% para el modelo Masked GFC. Esta variabilidad sugiere que los modelos podrían estar sobreajustándose a los datos de entrenamiento, lo que indica la necesidad de aplicar técnicas adicionales de regularización, como el uso de dropout más agresivo, la implementación de L2 regularization o la utilización de técnicas de aumento de datos (data augmentation) para mejorar la generalización y reducir el sobreajuste.

La precisión consistente del 71.43% obtenida por los modelos de clasificación adicionales, incluyendo SVM, Random Forest y Gradient Boosting, es un indicativo positivo de la robustez y la calidad de las características extraídas por los modelos CNN. Estos modelos adicionales demostraron ser capaces de aprovechar eficazmente las características generadas, logrando una precisión de clasificación elevada y uniforme. El modelo SVM, con su enfoque en la maximización del margen entre clases, demostró ser eficaz en este contexto. De igual manera, el modelo Random Forest, con su enfoque de ensemble de múltiples árboles de decisión, ofreció una sólida resistencia a la variabilidad de los datos. Por último, el modelo Gradient Boosting, conocido por su capacidad para corregir iterativamente los errores de modelos anteriores, también mostró un rendimiento robusto.

A pesar de los resultados positivos, queda claro que hay margen para mejoras adicionales. El ajuste de hiperparámetros, mediante técnicas como GridSearchCV o RandomizedSearchCV, podría optimizar aún más el rendimiento de los modelos de clasificación. Además, implementar técnicas avanzadas de aumento de datos podría no solo incrementar la cantidad de datos disponibles para el entrenamiento, sino también mejorar la capacidad de generalización de los modelos al exponerlos a una mayor diversidad de ejemplos. La regularización adicional, tanto a nivel de las capas convolucionales como de las densas, podría ayudar a mitigar el sobreajuste observado, asegurando que los modelos aprendan patrones verdaderamente representativos de los datos subyacentes.

Desarrollo del Software e Implementación de la Interfaz

Se ha conseguido configurar exitosamente el software utilizando el framework Django, proporcionando una interfaz de carga para imágenes MRI y datos tabulares. La interfaz permite a los usuarios seleccionar y cargar diferentes modelos de clasificación para los datos ingresados. Esta implementación

incluye formularios personalizados que facilitan la carga de archivos y la selección de modelos.

Sin embargo, durante las pruebas del sistema, se identificó un fallo crítico en el proceso de clasificación. A pesar de que los datos se cargan correctamente y los modelos pueden ser seleccionados, el proceso de clasificación no se completa, resultando en la incapacidad de clasificar los datos y, por ende, de evaluar la herramienta en un entorno práctico.

Problemas Identificados en el Preprocesamiento de Imágenes

Durante el preprocesamiento de las imágenes MRI, se realizó un redimensionamiento a un tamaño estándar de 64x64x64 píxeles. Este paso fue necesario debido a las limitaciones de capacidad computacional del equipo utilizado. Sin embargo, esta redimensión resultó en una pérdida significativa de resolución en las imágenes, lo que podría haber afectado negativamente el rendimiento de los modelos de clasificación. La pérdida de detalles importantes en las imágenes podría haber contribuido a la menor precisión observada en el modelo Masked GFC y a la variabilidad en la precisión de validación de ambos modelos.

Áreas de Mejora

Para asegurar el funcionamiento completo y eficiente de la herramienta, se deben abordar varias áreas de mejora:

1. Resolución de Fallos en la Clasificación:

- Identificar y corregir los errores en el proceso de clasificación que impiden la finalización del análisis.
- Implementar pruebas unitarias y de integración más exhaustivas para asegurar la robustez del sistema.

2. Optimización de Modelos:

- Realizar un ajuste fino de los hiperparámetros utilizando técnicas como GridSearchCV o RandomizedSearchCV para mejorar el rendimiento de los modelos de clasificación.
- Experimentar con diferentes arquitecturas de CNN y técnicas de regularización para mitigar el sobreajuste observado.

3. Mejora del Aumento de Datos (Data Augmentation):

- Implementar técnicas avanzadas de aumento de datos para generar un conjunto de datos más diversificado y robusto.
- Evaluar el impacto de diferentes transformaciones en la capacidad de generalización del modelo.

4. Desarrollo de la Interfaz de Usuario:

- Mejorar la usabilidad de la interfaz web para facilitar una experiencia de usuario más intuitiva y eficiente.
- Asegurar que la interfaz proporciona retroalimentación clara y útil durante el proceso de carga y clasificación.

Conclusiones

El desarrollo de este proyecto ha permitido alcanzar avances significativos en la construcción de una herramienta de clasificación para la predicción de la enfermedad de Alzheimer mediante el análisis de imágenes MRI. A pesar de los problemas identificados en el proceso de clasificación y la pérdida de resolución de las imágenes debido a redimensionar las imágenes, los resultados obtenidos durante el entrenamiento de los modelos y la implementación del software proporcionan una base sólida sobre la cual construir. Con las mejoras sugeridas, se espera que la herramienta pueda funcionar de manera efectiva, proporcionando diagnósticos más precisos y facilitando la intervención temprana en la gestión de la enfermedad de Alzheimer.

5. Glosario

Términos

- **Enfermedad de Alzheimer (EA):** Una enfermedad neurodegenerativa progresiva que causa deterioro cognitivo y pérdida de memoria.
- **Red Neuronal Convolucional (CNN):** Un tipo de red neuronal profunda que es particularmente eficaz para el procesamiento de datos con estructura de cuadrícula, como las imágenes.
- **Imágenes de Resonancia Magnética (MRI):** Técnica de imagen médica que utiliza campos magnéticos y ondas de radio para crear imágenes detalladas de los órganos y tejidos del cuerpo.
- **Normalización:** Proceso de ajuste de valores de datos a una escala común, generalmente en un rango de 0 a 1, para mejorar la consistencia en el análisis.
- **Preprocesamiento de Datos:** Conjunto de técnicas aplicadas a los datos crudos para prepararlos para el análisis, incluyendo la limpieza, normalización y transformación de los datos.
- **Imputación de Datos:** Técnica utilizada para reemplazar los valores faltantes en un conjunto de datos con estimaciones razonables.
- **Redimensionamiento:** Proceso de cambiar las dimensiones de una imagen o conjunto de datos a un tamaño específico.
- **Regularización:** Técnica utilizada en el aprendizaje automático para prevenir el sobreajuste del modelo a los datos de entrenamiento.
- **Sobreajuste:** Situación en la que un modelo de aprendizaje automático se ajusta demasiado a los datos de entrenamiento, resultando en un rendimiento deficiente en datos no vistos.

Acrónimos

- **CDR (Clinical Dementia Rating):** Escala utilizada para cuantificar la severidad de los síntomas de la demencia.
- **MMSE (Mini-Mental State Examination):** Prueba de 30 puntos utilizada para medir el deterioro cognitivo.
- **GFC (Gain Field Corrected):** Imágenes de resonancia magnética corregidas por el campo de ganancia.
- **Masked GFC:** Versión enmascarada del cerebro de las imágenes GFC registradas al atlas.
- **KNN (K-Nearest Neighbors):** Algoritmo de aprendizaje supervisado utilizado para clasificación e imputación de datos.
- **SVM (Support Vector Machine):** Algoritmo de aprendizaje supervisado utilizado para clasificación y regresión.
- **Random Forest (RF):** Algoritmo de aprendizaje supervisado que construye múltiples árboles de decisión y los combina para mejorar la precisión.
- **Gradient Boosting (GB):** Algoritmo de aprendizaje supervisado que construye modelos de forma secuencial, cada uno corrigiendo los errores de los modelos anteriores.

- **Django:** Framework web de alto nivel utilizado para el desarrollo de aplicaciones web rápidas y seguras.
- **OASIS (Open Access Series of Imaging Studies):** Conjunto de datos que proporciona acceso abierto a datos de imágenes de resonancia magnética para estudios de neurociencia.
- **ROI (Region of Interest):** Región específica de una imagen seleccionada para un análisis detallado.
- **L2 Regularization:** Técnica de regularización que añade una penalización proporcional al cuadrado de los coeficientes del modelo para prevenir el sobreajuste.
- **StandardScaler:** Técnica de estandarización de datos que ajusta las características para que tengan una media de 0 y una desviación estándar de 1.
- **ITK-SNAP:** Software utilizado para la segmentación y visualización de imágenes médicas.
- **Jupyter Notebook:** Aplicación web que permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo.
- **GitHub:** Plataforma de control de versiones y colaboración que permite a los desarrolladores gestionar su código fuente y realizar trabajo colaborativo.
- **NIfTI (Neuroimaging Informatics Technology Initiative):** Formato de archivo estándar para el almacenamiento de datos de imágenes de neurociencia.

6. Bibliografía

Abadi, M., Barham, P., Chen, J., et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 265-283.

Brett, M., Markiewicz, C. J., Hanke, M., et al. (2020). nipy/nibabel: 3.2.1. Zenodo. https://doi.org/10.5281/zenodo.4295521

Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

Cardoso, M. J., Li, W., Brown, R., et al. (2022). MONAI: An open-source framework for deep learning in healthcare. Nature Biomedical Engineering, 6(2), 154-169.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297.

Django Software Foundation. (2023). Django Documentation. https://docs.djangoproject.com/en/stable/

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics, 29(5), 1189-1232.

Gupta, S., Saravanan, V., Choudhury, A., Alqahtani, A., Abonazel, M. R., & Babu, K. S. (2022). Supervised Computer-Aided Diagnosis (CAD) Methods for Classifying Alzheimer's Disease-Based Neurodegenerative Disorders. Comput. Math. Methods Med., 2022. https://doi.org/10.1155/2022/9092289.

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.

Lahmiri, S. (2023). Integrating convolutional neural networks, kNN, and Bayesian optimization for efficient diagnosis of Alzheimer's disease in magnetic resonance images. Biomed. Signal Process. Control, 80, 104375. https://doi.org/10.1016/J.BSPC.2022.104375.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

Marcus, D. S., Fotenos, A. F., Csernansky, J. G., Morris, J. C., & Buckner, R. L. (2007). Open Access Series of Imaging Studies: Longitudinal MRI Data in Nondemented and Demented Older Adults. Journal of Cognitive Neuroscience, 22(12), 2677-2684.

Marcus, D. S., Wang, T. H., Parker, J., Csernansky, J. G., Morris, J. C., & Buckner, R. L. (2007). Open Access Series of Imaging Studies (OASIS): Cross-sectional MRI Data in Young, Middle Aged, Nondemented, and Demented Older Adults. Journal of Cognitive Neuroscience, 19(9), 1498-1507. https://doi.org/10.1162/jocn.2007.19.9.1498.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 51-56.

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Pinaya, W. H. L., et al. (2021). Using normative modelling to detect disease progression in mild cognitive impairment and Alzheimer's disease in a cross-sectional multi-cohort study. Sci. Rep., 11(1), 1-13. https://doi.org/10.1038/s41598-021-95098-0.

Saratxaga, C. L., et al. (2021). MRI deep learning-based solution for Alzheimer's disease prediction. J. Pers. Med., 11(9). https://doi.org/10.3390/jpm11090902.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929-1958.

Troyanskaya, O., Cantor, M., Sherlock, G., et al. (2001). Missing value estimation methods for DNA microarrays. Bioinformatics, 17(6), 520-525.

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., et al. (2014). scikit-image: Image processing in Python. PeerJ, 2, e453.

GitHub Repositories Using CNN with OASIS-1:

GMattheisen, Predicting Alzheimer's from MRI. GitHub Repository. Retrieved from GitHub

diegoperac, Alzheimer's Disease Prediction. GitHub Repository. Retrieved from GitHub

IzzyHuang, IEEE-Alzheimer-CNN-Classifier. GitHub Repository. Retrieved from GitHub

himanshub1007, Alzheimer's Disease Prediction Using Deep Learning. GitHub Repository. Retrieved from GitHub

Kaggle Dataset Analysis:

Alzheimer's Disease Detection on OASIS. Retrieved from Kaggle

7. Anexos

https://github.com/mmiguelgar/ALZDET

In [ ]:

```python
#----- Descripción, escalado y codificación de los datos tabulares ------------------
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import KNNImputer
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns

file_path = 'demographic_clinical_data.xlsx'
data = pd.read_excel(file_path)

# Cambiar los nombres de las columnas a strings
data.columns = data.columns.astype(str)

print(data.info())
print(data.columns)
print(data.head())
print(data.describe())

# Asignar 0 a los valores faltantes en 'Delay'
data['Delay'].fillna(0, inplace=True)

print(data.isnull().sum())

# Convertir las columnas categóricas a numéricas
label_encoder_gender = LabelEncoder()
label_encoder_hand = LabelEncoder()

# ATransformar los datos
data.loc[:, 'M/F'] = label_encoder_gender.fit_transform(data['M/F'])
data.loc[:, 'Hand'] = label_encoder_hand.fit_transform(data['Hand'])

print(data.head())

# Identificar sujetos sin demencia control mediante MR2 y homólogos MR1
control_subjects_mr2 = data[(data['ID'].str.contains('_MR2')) & (data['CDR'].isnull())].copy()
control_subjects_mr1 = data[(data['ID'].str.contains('_MR1')) & (data['ID'].str[:-3]
                                                                 .isin(control_subjects_mr2['ID'].str[:-3]))].copy()

# Asignar CDR=0 a sujetos control sin demencia
control_subjects_mr1.loc[:, 'CDR'] = 0
control_subjects_mr2.loc[:, 'CDR'] = 0

print(f"Sujetos control MR1: {len(control_subjects_mr1)}")
print(control_subjects_mr1.head())

print(f"Sujetos control MR2: {len(control_subjects_mr2)}")
print(control_subjects_mr2.head())

# Actualizar el DataFrame original con los cambios
data.loc[control_subjects_mr1.index, 'CDR'] = 0
data.loc[control_subjects_mr2.index, 'CDR'] = 0

# Separar sujetos CDR=0 con datos completos (para entrenamiento de KNNImputer)
complete_data_subjects_CDR0 = data[data['CDR'] == 0].dropna(subset=['Educ', 'SES', 'MMSE'])

# Configurar KNNImputer
imputer = KNNImputer(n_neighbors=4)

# Seleccionar columnas relevantes para la imputación
columns_to_impute = ['M/F', 'Hand', 'Age', 'Educ', 'SES', 'MMSE', 'eTIV', 'nWBV', 'ASF']
control_subjects = pd.concat([control_subjects_mr1, control_subjects_mr2])

# Extraer solo las columnas a imputar y las demás relevantes
complete_data_for_imputation = complete_data_subjects_CDR0[columns_to_impute]
control_data_for_imputation = control_subjects[columns_to_impute]

# Combinar los datos completos con los sujetos de control para la imputación
combined_data_for_imputation = pd.concat([complete_data_for_imputation, control_data_for_imputation], axis=0)

# Imputar los valores faltantes
imputed_data = imputer.fit_transform(combined_data_for_imputation)

# Crear un DataFrame con los datos imputados
imputed_df = pd.DataFrame(imputed_data, columns=columns_to_impute)

# Separar los datos imputados correspondientes a los sujetos de control
imputed_control_data = imputed_df.iloc[-len(control_subjects):]

# Actualizar los sujetos de control con los datos imputados
control_subjects_imputed = control_subjects.copy()
control_subjects_imputed[columns_to_impute] = imputed_control_data[columns_to_impute].values

# Verificar los resultados de la imputación
print(control_subjects_imputed.head())

# Actualizar el DataFrame original con los valores imputados
data.update(control_subjects_imputed)

print(f"Sujetos CDR=0 con datos completos: {len(complete_data_subjects_CDR0)}")
print(f"Sujetos control imputados: {len(control_subjects_imputed)}")
```

```python
# Separar sujetos con datos faltantes
missing_data_subjects = data[data.isnull().any(axis=1)]
print(missing_data_subjects.isnull().sum())

# Excluir 'SES'
columns_to_check = data.columns.difference(['SES'])

# Separar
missing_SES_only = data.dropna(subset=columns_to_check)

print(f"Total de sujetos: {len(data)}")
print(f"Sujetos con solo SES faltantes: {len(missing_SES_only)}")
print(missing_SES_only.isnull().sum())

#Imputar valores faltantes en SES

# Configurar KNNImputer
imputer = KNNImputer(n_neighbors=5)

# Seleccionar columnas relevantes para la imputación
columns_to_impute = ['M/F', 'Hand', 'Age', 'Educ', 'MMSE', 'CDR', 'eTIV', 'nWBV', 'ASF', 'SES']

# Aplicar la imputación solo en las filas con datos faltantes en 'SES'
complete_data_subjects_imputed = missing_SES_only.copy()
complete_data_subjects_imputed[columns_to_impute] = imputer.fit_transform(missing_SES_only[columns_to_impute])

# Verificar los resultados de la imputación
print(complete_data_subjects_imputed[['ID', 'SES']].head(20))

# Actualizar el DataFrame original con los valores imputados
data.update(complete_data_subjects_imputed)
```

```python
final_complete_subjects = data.dropna()
final_missing_subjects = data[data.isnull().any(axis=1)]
print(f"Sujetos con datos completos: {len(final_complete_subjects)}")
print(f"Sujetos con datos faltantes: {len(final_missing_subjects)}")

print(data['CDR'].value_counts())

# Separar muestras MR2 por sobrerepresentación de individuos sin demencia
mr2_subjects = final_complete_subjects[final_complete_subjects['ID'].str.contains('_MR2')]
model_subjects = final_complete_subjects[~final_complete_subjects['ID'].str.contains('_MR2')]

print(f"Sujetos MR2: {len(mr2_subjects)}")
print(f"Sujetos con datos completos (sin MR2): {len(model_subjects)}")

final_complete_subjects = data.dropna()
final_missing_subjects = data[data.isnull().any(axis=1)]
print(f"Sujetos con datos completos: {len(final_complete_subjects)}")
print(f"Sujetos con datos faltantes: {len(final_missing_subjects)}")

print(data['CDR'].value_counts())

# Separar muestras MR2 por sobrerepresentación de individuos sin demencia
mr2_subjects = final_complete_subjects[final_complete_subjects['ID'].str.contains('_MR2')]
model_subjects = final_complete_subjects[~final_complete_subjects['ID'].str.contains('_MR2')]

print(f"Sujetos MR2: {len(mr2_subjects)}")
print(f"Sujetos con datos completos (sin MR2): {len(model_subjects)}")

# Guardar los DataFrames modificados en el archivo .xlsx
new_file_path = 'demographic_clinical_data_imputed.xlsx'
with pd.ExcelWriter(new_file_path) as writer:
    data.to_excel(writer, sheet_name='Updated_Data', index=False)
    model_subjects.to_excel(writer, sheet_name='Model_Subjects', index=False)
    final_complete_subjects.to_excel(writer, sheet_name='Complete_Subjects', index=False)
    control_subjects_imputed.to_excel(writer,sheet_name='Control_Subjects_Imputed', index=False)
    final_missing_subjects.to_excel(writer,sheet_name='Subjects_Missing_Data', index=False)

new_file = 'demographic_clinical_data_imputed.xlsx'
sheet_name = 'Model_Subjects'
new_data = pd.read_excel(new_file, sheet_name=sheet_name)

matrix_corr = new_data.drop(columns=['ID', 'Hand']).corr()
print(new_data.describe())
print(matrix_corr)

# Aplicar One-Hot Encoding a 'CDR'
cdr_one_hot = pd.get_dummies(new_data['CDR'], prefix='CDR', dtype=float)

# Extraer columnas categóricas antes de escalar
ids = new_data['ID']
gender = new_data['M/F']
hand = new_data['Hand']
cdr = new_data['CDR']

scaler = StandardScaler()

columns_scaled = new_data.drop(columns=['ID', 'M/F', 'Hand', 'CDR'])
scaled_features = scaler.fit_transform(columns_scaled)

# Creación del DataFrame escalado
scaled_data = pd.DataFrame(scaled_features, columns=columns_scaled.columns)


# Añadir las columnas 'ID', 'Hand', 'M/F' y la columna one-hot de 'CDR'al DataFrame escalado
```

```
scaled_data.insert(0, 'Hand', hand.values)
scaled_data.insert(0, 'M/F', gender.values)
scaled_data = pd.concat([scaled_data, pd.get_dummies(cdr, prefix='CDR')], axis=1)
scaled_data.insert(0, 'ID', ids.values)


# Guardar el DataFrame escalado en un archivo Excel
output_file_path = 'scaled_demographic_data.xlsx'
with pd.ExcelWriter(output_file_path) as writer:
    scaled_data.to_excel(writer, sheet_name='Complete_Subjects_Scaled', index=False)

print(f"El DataFrame escalado se ha guardado como '{output_file_path}'")

#--------------------------- Normalización imágenes ------------------------------------------------------------
import nibabel as nib
import numpy as np
import os
import logging

# Configurar el logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# Directorios para los datos de imágenes
RAW_GFC_DIR = 'D:/data/images/raw/gfc'
RAW_MASKED_DIR = 'D:/data/images/raw/masked'
PROCESSED_GFC_DIR = 'D:/data/images/processed/gfc_processed'
PROCESSED_MASKED_DIR = 'D:/data/images/processed/masked_gfc_fseg_processed'

def load_and_preprocess_image(file_path):
    logging.info(f'Cargando imagen: {file_path}')
    img = nib.load(file_path)
    data = img.get_fdata()
    data_normalized = (data - np.min(data)) / (np.max(data) - np.min(data))
    return data_normalized.astype(np.float32)

def save_preprocessed_image(data, output_path):
    logging.info(f'Guardando imagen preprocesada: {output_path}')
    img = nib.Nifti1Image(data, np.eye(4))
    nib.save(img, output_path)

def preprocess_images(raw_dir, processed_dir):
    if not os.path.exists(processed_dir):
        os.makedirs(processed_dir)
    for file_name in os.listdir(raw_dir):
        if file_name.endswith('.img'):
            file_path = os.path.join(raw_dir, file_name)
            data = load_and_preprocess_image(file_path)
            output_path = os.path.join(processed_dir, file_name.replace('.img', '_processed.nii'))
            save_preprocessed_image(data, output_path)
    logging.info(f'Preprocesamiento completo para {raw_dir}.')

if __name__ == "__main__":
    preprocess_images(RAW_GFC_DIR, PROCESSED_GFC_DIR)
    preprocess_images(RAW_MASKED_DIR, PROCESSED_MASKED_DIR)

#----------------------- Undersampling-------------------------------------------------
import pandas as pd
import shutil
import os
import re


# Cargar el archivo
file_path = r"D:\data\scaled_demographic_data.xlsx"
df = pd.read_excel(file_path)

# Filtrar las muestras con CDR=0
cdr_0_samples = df[df['CDR_0.0'] == 1]

# Seleccionar aleatoriamente 80 muestras para eliminar
samples_to_remove = cdr_0_samples.sample(n=80, random_state=42)

# Eliminar las muestras seleccionadas del DataFrame original
df_filtered = df.drop(samples_to_remove.index)

# Guardar el nuevo conjunto de datos en un archivo
output_file_path = r"D:\data\demographic_clinical_data_filtered.xlsx"
df_filtered.to_excel(output_file_path, index=False)

print(f"Conjunto de datos filtrado guardado en {output_file_path}")

# Separar las imágenes correspodientes en otro directorio

# Rutas de archivos y directorios
excel_file_path = r"D:\data\demographic_clinical_data_filtered.xlsx"
gfc_dir = r"D:\data\images\gfc_complete"
masked_gfc_dir = r"D:\data\images\masked_gfc_complete"
output_gfc_dir = r"D:\data\images\filtered_gfc_complete"
output_masked_gfc_dir = r"D:\data\images\filtered_masked_gfc_complete"

# Crear los directorios de salida si no existen
os.makedirs(output_gfc_dir, exist_ok=True)
os.makedirs(output_masked_gfc_dir, exist_ok=True)

# Cargar archivo Excel
df_filtered = pd.read_excel(excel_file_path)
```

```python
# Obtener los IDs
subject_ids = df_filtered['ID'].tolist()

def find_file_with_pattern(directory, pattern):
    for filename in os.listdir(directory):
        if re.match(pattern, filename):
            return filename
    return None

# Copiar las imágenes correspondientes a los IDs
for subject_id in subject_ids:
    gfc_pattern = re.compile(f"{subject_id}_mpr_n\d+_anon_111_t88_gfc_processed.nii")
    masked_gfc_pattern = re.compile(f"{subject_id}_mpr_n\d+_anon_111_t88_masked_gfc_fseg_processed.nii")

    gfc_image_filename = find_file_with_pattern(gfc_dir, gfc_pattern)
    masked_gfc_image_filename = find_file_with_pattern(masked_gfc_dir, masked_gfc_pattern)

    if gfc_image_filename:
        gfc_image_path = os.path.join(gfc_dir, gfc_image_filename)
        shutil.copy(gfc_image_path, output_gfc_dir)
    else:
        print(f"Imagen no encontrada para el patrón: {gfc_pattern.pattern}")

    if masked_gfc_image_filename:
        masked_gfc_image_path = os.path.join(masked_gfc_dir, masked_gfc_image_filename)
        shutil.copy(masked_gfc_image_path, output_masked_gfc_dir)
    else:
        print(f"Imagen no encontrada para el patrón: {masked_gfc_pattern.pattern}")

print("Imágenes copiadas exitosamente.")
```

```python
#---------------- Oversampling ----------------------------

# Cargar archivo
file_path = r"D:\data\demographic_clinical_data_filtered.xlsx"
df = pd.read_excel(file_path)

# Identificadores de sujetos específicos para CDR=2
cdr_2_subjects = ['OAS1_0308_MR1', 'OAS1_0351_MR1']

# Número de nuevas muestras para crear
num_new_samples_cdr_2 = 10
num_new_samples_cdr_1 = 3

# Función para generar nuevas muestras
def generate_augmented_samples(samples, num_samples, prefix):
    augmented_samples = samples.copy()
    for i in range(num_samples):
        aug_samples = samples.copy()
        aug_samples['ID'] = aug_samples['ID'].apply(lambda x: f"{x}_aug_{i}")
        augmented_samples = pd.concat([augmented_samples, aug_samples])
    return augmented_samples.tail(num_samples)

# Crear nuevas muestras para CDR=2
cdr_2_samples = df[df['CDR_2.0'] == 1]
new_samples_cdr_2 = []
for index, sample in cdr_2_samples.iterrows():
    sample_df2 = sample.to_frame().T
    new_samples_cdr_2.append(generate_augmented_samples(sample_df2, num_new_samples_cdr_2, sample['ID']))

# Crear nuevas muestras para CDR=1
cdr_1_samples = df[df['CDR_1.0'] == 1]
new_samples_cdr_1 = []
for index, sample in cdr_1_samples.iterrows():
    sample_df = sample.to_frame().T
    new_samples_cdr_1.append(generate_augmented_samples(sample_df, num_new_samples_cdr_1, sample['ID']))

# Combinar las muestras originales con las nuevas muestras creadas
df_balanced = pd.concat([df] + new_samples_cdr_2 + new_samples_cdr_1)

# Guardar el nuevo conjunto de datos en un archivo Excel
output_file_path = r"D:\data\demographic_clinical_data_balanced.xlsx"
df_balanced.to_excel(output_file_path, index=False)

print(f"Conjunto de datos balanceado guardado en {output_file_path}")

# ------------- Data Augmentation images -------------------------------
import os
import numpy as np
import nibabel as nib
import pandas as pd
import torch
from monai.transforms import Compose, ScaleIntensity, RandFlip, RandRotate90, RandAffine, RandSpatialCrop
from monai.utils import set_determinism

# Configurar la semilla para la reproducibilidad
set_determinism(seed=42)

# Función para cargar las imagenes
def load_image(file_path):
    if not os.path.exists(file_path):
        raise FileNotFoundError(f"No such file or no access: '{file_path}'")
    img = nib.load(file_path)
    data = img.get_fdata()
    return data, img.affine
```

```python
# Función para guardar las imágenes
def save_image(data, affine, file_path):
    nib.save(nib.Nifti1Image(data, affine), file_path)

# Función para realizar cortes y Data Augmentation
def augment_image(data, affine, num_augmentations=10):
    augmented_images = []
    transforms = Compose([
        ScaleIntensity(),
        RandFlip(prob=0.5, spatial_axis=[0, 1, 2]),
        RandRotate90(prob=0.5, max_k=3, spatial_axes=[0, 1]),
        RandAffine(prob=0.5, translate_range=(5, 5, 5)),
        RandSpatialCrop(roi_size=(88, 104, 88), random_size=False)
    ])

    for _ in range(num_augmentations):
        # Aplicar transformaciones
        augmented_data = transforms(data)
        # Convertir de tensor a numpy array
        if isinstance(augmented_data, torch.Tensor):
            augmented_data = augmented_data.numpy()
        # Restaurar la dimensión del canal
        if augmented_data.shape[-1] != 1:
            augmented_data = np.expand_dims(augmented_data, axis=-1)
        augmented_images.append((augmented_data, affine))

    return augmented_images

# Directorios de imágenes y nuevas imágenes
input_dir_gfc = r"D:\data\images\filtered_gfc_complete"
input_dir_masked_gfc = r"D:\data\images\filtered_masked_gfc_complete"
output_dir_gfc = r"D:\data\images\augmented_gfc_complete"
output_dir_masked_gfc = r"D:\data\images\augmented_masked_gfc_complete"
os.makedirs(output_dir_gfc, exist_ok=True)
os.makedirs(output_dir_masked_gfc, exist_ok=True)

# Cargar el archivo Excel
file_path = r"D:\data\demographic_clinical_data_balanced.xlsx"
df = pd.read_excel(file_path)

# Filtrar las muestras con CDR=2 y CDR=1
cdr_2_samples = df[df['CDR_2.0'] == 1]
cdr_1_samples = df[df['CDR_1.0'] == 1]

# Función para procesar y aumentar imágenes
def process_samples(samples, num_augmentations, output_dir_gfc, output_dir_masked_gfc):
    for idx, row in samples.iterrows():
        subject_id = row['ID']
        pattern_gfc = rf"{subject_id}_mpr_n\d+_anon_111_t88_gfc_processed\.nii"
        pattern_masked_gfc = rf"{subject_id}_mpr_n\d+_anon_111_t88_masked_gfc_fseg_processed\.nii"

        image_file_gfc = find_file(input_dir_gfc, pattern_gfc)
        image_file_masked_gfc = find_file(input_dir_masked_gfc, pattern_masked_gfc)

        # Procesar imágenes gfc
        try:
            data_gfc, affine_gfc = load_image(image_file_gfc)
            augmented_images_gfc = augment_image(data_gfc, affine_gfc, num_augmentations=num_augmentations)

            for i, (augmented_data, augmented_affine) in enumerate(augmented_images_gfc):
                output_file = os.path.join(output_dir_gfc, f"{subject_id}_aug_{i}_mpr_n4_anon_111_t88_gfc_processed.nii")
                save_image(augmented_data, augmented_affine, output_file)

        except FileNotFoundError as e:
            print(e)

        # Procesar imagenes masked_gfc
        try:
            data_masked_gfc, affine_masked_gfc = load_image(image_file_masked_gfc)
            augmented_images_masked_gfc = augment_image(data_masked_gfc, affine_masked_gfc,
                                                        num_augmentations=num_augmentations)

            for i, (augmented_data, augmented_affine) in enumerate(augmented_images_masked_gfc):
                output_file = os.path.join(output_dir_masked_gfc,
                                           f"{subject_id}_aug_{i}_mpr_n4_anon_111_t88_masked_gfc_fseg_processed.nii")
                save_image(augmented_data, augmented_affine, output_file)

        except FileNotFoundError as e:
            print(e)

# Aumentar las imágenes de CDR=2
process_samples(cdr_2_samples, num_augmentations=20, output_dir_gfc=output_dir_gfc,
                output_dir_masked_gfc=output_dir_masked_gfc)

# Aumentar las imágenes de CDR=1
output_dir_gfc_cdr1 = os.path.join(output_dir_gfc, 'cdr1')
output_dir_masked_gfc_cdr1 = os.path.join(output_dir_masked_gfc, 'cdr1')
os.makedirs(output_dir_gfc_cdr1, exist_ok=True)
os.makedirs(output_dir_masked_gfc_cdr1, exist_ok=True)

process_samples(cdr_1_samples, num_augmentations=1, output_dir_gfc=output_dir_gfc_cdr1,
                output_dir_masked_gfc=output_dir_masked_gfc_cdr1)

print(f"Imágenes augmentadas guardadas en {output_dir_gfc} y {output_dir_masked_gfc}")
print(f"Imágenes augmentadas de CDR=1 guardadas en {output_dir_gfc_cdr1} y {output_dir_masked_gfc_cdr1}")
```

```python
#------------------------------- Redimensionar imágenes originales normalizadas -------------------------------
import os
import re
import numpy as np
import nibabel as nib
import pandas as pd
import torch
from monai.transforms import Compose, Resize

# Función para cargar imagen
def load_image(file_path):
    if not os.path.exists(file_path):
        raise FileNotFoundError(f"No such file or no access: '{file_path}'")
    img = nib.load(file_path)
    data = img.get_fdata()
    return data, img.affine

# Función para guardar imágenes
def save_image(data, affine, file_path):
    nib.save(nib.Nifti1Image(data, affine), file_path)

# Función para redimensionar imágenes
def resize_image(data, target_size=(88, 104, 88)):
    transforms = Compose([
        Resize(spatial_size=target_size)
    ])
    resized_data = transforms(data)
    if isinstance(resized_data, torch.Tensor):
        resized_data = resized_data.numpy()
    if resized_data.shape[-1] != 1:
        resized_data = np.expand_dims(resized_data, axis=-1)
    return resized_data

# Función para encontrar archivos basados en un patrón
def find_file(directory, pattern):
    for filename in os.listdir(directory):
        if re.search(pattern, filename):
            return os.path.join(directory, filename)
    return None

# Directorios de imágenes y nuevas imágenes
input_dir_gfc = r"D:\data\images\filtered_gfc_complete"
input_dir_masked_gfc = r"D:\data\images\filtered_masked_gfc_complete"
output_dir_gfc = r"D:\data\images\resized_gfc_complete"
output_dir_masked_gfc = r"D:\data\images\resized_masked_gfc_complete"
os.makedirs(output_dir_gfc, exist_ok=True)
os.makedirs(output_dir_masked_gfc, exist_ok=True)

# Cargar el archivo
file_path = r"D:\data\demographic_clinical_data_filtered.xlsx"
df = pd.read_excel(file_path)

# Procesar y redimensionar imágenes
def process_and_resize_images(samples, output_dir_gfc, output_dir_masked_gfc):
    for idx, row in samples.iterrows():
        subject_id = row['ID']
        pattern_gfc = rf"{subject_id}_mpr_n\d+_anon_111_t88_gfc_processed\.nii"
        pattern_masked_gfc = rf"{subject_id}_mpr_n\d+_anon_111_t88_masked_gfc_fseg_processed\.nii"

        image_file_gfc = find_file(input_dir_gfc, pattern_gfc)
        image_file_masked_gfc = find_file(input_dir_masked_gfc, pattern_masked_gfc)

        # Redimensionar y guardar gfc
        if image_file_gfc:
            try:
                data_gfc, affine_gfc = load_image(image_file_gfc)
                resized_data_gfc = resize_image(data_gfc)
                output_file_gfc = os.path.join(output_dir_gfc,
                                               f"{subject_id}_resized_mpr_n_anon_111_t88_gfc_processed.nii")
                save_image(resized_data_gfc, affine_gfc, output_file_gfc)
            except FileNotFoundError as e:
                print(e)
        else:
            print(f"Archivo GFC no encontrado para el sujeto {subject_id}")

        # Redimensionar y guardar imágenes masked_gfc
        if image_file_masked_gfc:
            try:
                data_masked_gfc, affine_masked_gfc = load_image(image_file_masked_gfc)
                resized_data_masked_gfc = resize_image(data_masked_gfc)
                output_file_masked_gfc = os.path.join(output_dir_masked_gfc,
                                                      f"{subject_id}_resized_mpr_n_anon_111_t88_masked_gfc_fseg_processed.nii")
                save_image(resized_data_masked_gfc, affine_masked_gfc, output_file_masked_gfc)
            except FileNotFoundError as e:
                print(e)
        else:
            print(f"Archivo Masked GFC no encontrado para el sujeto {subject_id}")

# Procesar todas las imágenes en el conjunto de datos
process_and_resize_images(df, output_dir_gfc, output_dir_masked_gfc)

print(f"Imágenes redimensionadas guardadas en {output_dir_gfc} y {output_dir_masked_gfc}")


#------------------------------------ modelo CNN, SVM, RF, GB para ambos conjuntos de imágenes ----------------------
```

```python
import numpy as np
import os
import re
import nibabel as nib
import logging
import pandas as pd
from sklearn.model_selection import train_test_split
from skimage.transform import resize
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv3D, MaxPooling3D, Flatten, Dense, Dropout
import tensorflow as tf
from joblib import dump
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

# Configurar el logging
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')

# Cargar el archivo de datos clínicos actualizado
file_path = 'D:/data/demographic_clinical_data_balanced.xlsx'
complete_subjects = pd.read_excel(file_path)

# Directorios con las imágenes
GFC_PROCESSED_DIR = 'D:/data/images/gfc'
MASKED_GFC_PROCESSED_DIR = 'D:/data/images/masked'

# Reducir el tamaño de las imágenes
TARGET_SHAPE = (64, 64, 64)

def load_and_preprocess_image(file_path, target_shape=TARGET_SHAPE):
    try:
        img = nib.load(file_path)
        data = img.get_fdata()
        data_normalized = (data - np.min(data)) / (np.max(data) - np.min(data))
        if data_normalized.shape[:3] != target_shape:
            data_resized = resize(data_normalized, target_shape, mode='constant', anti_aliasing=True)
            logging.debug(f'Redimensionando imagen {file_path} de {data_normalized.shape[:3]} a {target_shape}')
        else:
            data_resized = data_normalized
        return data_resized.astype(np.float32)
    except Exception as e:
        logging.error(f"Error al cargar la imagen {file_path}: {e}")
        return None

def find_images(subject_id, directory, pattern_template):
    pattern = re.compile(pattern_template.format(subject_id=subject_id))
    matched_files = [f for f in os.listdir(directory) if pattern.match(f)]
    return matched_files

def load_data(subjects, gfc_directory, masked_directory):
    data_gfc = []
    data_masked = []
    labels = []
    valid_subjects = []

    for index, row in subjects.iterrows():
        if index % 10 == 0:
            logging.info(f'Procesando sujeto {index+1}/{len(subjects)}: {row["ID"]}')
        subject_id = row['ID']

        # Cargar imágenes GFC
        gfc_image_files = find_images(subject_id, gfc_directory, pattern_template="{subject_id}_111_t88_gfc.nii")
        gfc_subject_data = []
        for file_name in gfc_image_files:
            file_path = os.path.join(gfc_directory, file_name)
            image_data = load_and_preprocess_image(file_path)
            if image_data is not None:
                gfc_subject_data.append(image_data)
            else:
                logging.warning(f"Imagen gfc no cargada correctamente: {file_path}")

        # Cargar imágenes enmascaradas
        masked_image_files = find_images(subject_id, masked_directory,
                                         pattern_template="{subject_id}_111_t88_masked_gfc_fseg.nii")
        masked_subject_data = []
        for file_name in masked_image_files:
            file_path = os.path.join(masked_directory, file_name)
            image_data = load_and_preprocess_image(file_path)
            if image_data is not None:
                masked_subject_data.append(image_data)
            else:
                logging.warning(f"Imagen masked no cargada correctamente: {file_path}")

        if gfc_subject_data and masked_subject_data:
            averaged_gfc_data = np.mean(gfc_subject_data, axis=0)
            averaged_masked_data = np.mean(masked_subject_data, axis=0)
            data_gfc.append(averaged_gfc_data)
            data_masked.append(averaged_masked_data)

            # Obtener la etiqueta CDR
            cdr_value = np.argmax(row[['CDR_0.0', 'CDR_0.5', 'CDR_1.0', 'CDR_2.0']].values)
            labels.append(cdr_value)
            valid_subjects.append(subject_id)
            logging.debug(f'Sujeto {subject_id} procesado con éxito')
        else:
            logging.warning(f"No se encontraron imágenes válidas para el sujeto: {subject_id}")
```

```
    logging.info(f"Total de sujetos válidos: {len(valid_subjects)}")
    return np.array(data_gfc), np.array(data_masked), np.array(labels)

# Modelo CNN
def build_cnn_model(input_shape):
    model = Sequential([
        Conv3D(32, (3, 3, 3), activation='relu', input_shape=input_shape),
        MaxPooling3D((2, 2, 2)),
        Dropout(0.25),
        Conv3D(64, (3, 3, 3), activation='relu'),
        MaxPooling3D((2, 2, 2)),
        Dropout(0.25),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(4, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

# Entrenar y guardar modelos
def train_and_save_models(gfc_directory, masked_directory):
    logging.info('Cargando datos...')
    gfc_data, masked_data, labels = load_data(complete_subjects, gfc_directory, masked_directory)

    if gfc_data.size == 0 or masked_data.size == 0:
        logging.error("No se han cargado datos. Verifica los archivos de imagen y sus rutas.")
        return

    try:
        gfc_data = gfc_data.reshape((gfc_data.shape[0], TARGET_SHAPE[0], TARGET_SHAPE[1], TARGET_SHAPE[2], 1))
        masked_data = masked_data.reshape((masked_data.shape[0], TARGET_SHAPE[0], TARGET_SHAPE[1], TARGET_SHAPE[2], 1))
    except ValueError as e:
        logging.error(f"Error al redimensionar los datos: {e}")
        return

    # Dividir los datos en conjuntos de entrenamiento y prueba
    X_train_gfc, X_test_gfc, X_train_masked, X_test_masked, y_train, y_test = train_test_split(
        gfc_data, masked_data, labels, test_size=0.2, random_state=42)

    # Modelo CNN para GFC
    logging.info('Entrenando modelo CNN para GFC...')
    cnn_model_gfc = build_cnn_model((TARGET_SHAPE[0], TARGET_SHAPE[1], TARGET_SHAPE[2], 1))
    cnn_model_gfc.fit(X_train_gfc, y_train, epochs=10, validation_split=0.1, batch_size=2)
    cnn_model_gfc.save('cnn_model_gfc.h5')

    # Modelo CNN para Masked GFC
    logging.info('Entrenando modelo CNN para Masked GFC...')
    cnn_model_masked = build_cnn_model((TARGET_SHAPE[0], TARGET_SHAPE[1], TARGET_SHAPE[2], 1))
    cnn_model_masked.fit(X_train_masked, y_train, epochs=10, validation_split=0.1, batch_size=2)
    cnn_model_masked.save('cnn_model_masked_gfc.h5')

    # Extracción de características
    logging.info('Extrayendo características con los modelos CNN...')
    train_features_gfc = cnn_model_gfc.predict(X_train_gfc)
    validation_features_gfc = cnn_model_gfc.predict(X_test_gfc)
    train_features_masked = cnn_model_masked.predict(X_train_masked)
    validation_features_masked = cnn_model_masked.predict(X_test_masked)

    # Concatenar características
    train_features = np.concatenate((train_features_gfc, train_features_masked), axis=1)
    validation_features = np.concatenate((validation_features_gfc, validation_features_masked), axis=1)

    # Modelos de clasificación adicionales
    logging.info('Entrenando modelos de clasificación...')

    # SVM Model
    logging.info('Entrenando modelo SVM...')
    svm_model = SVC(kernel='linear')
    svm_model.fit(train_features, y_train)
    dump(svm_model, 'svm_model.pkl')

    # Random Forest Model
    logging.info('Entrenando modelo Random Forest...')
    rf_model = RandomForestClassifier(n_estimators=100)
    rf_model.fit(train_features, y_train)
    dump(rf_model, 'rf_model.pkl')

    # Gradient Boosting Model
    logging.info('Entrenando modelo Gradient Boosting...')
    gb_model = GradientBoostingClassifier(n_estimators=100)
    gb_model.fit(train_features, y_train)
    dump(gb_model, 'gb_model.pkl')

    # Evaluar los modelos en el conjunto de prueba
    logging.info('Evaluando modelos...')
    svm_acc = svm_model.score(validation_features, y_test)
    rf_acc = rf_model.score(validation_features, y_test)
    gb_acc = gb_model.score(validation_features, y_test)

    logging.info(f'Precisión del modelo SVM: {svm_acc}')
    logging.info(f'Precisión del modelo Random Forest: {rf_acc}')
    logging.info(f'Precisión del modelo Gradient Boosting: {gb_acc}')

    logging.info('Entrenamiento y guardado de modelos completado.')
```

```
if __name__ == "__main__":
    train_and_save_models(GFC_PROCESSED_DIR, MASKED_GFC_PROCESSED_DIR)


# --------------------------------Visualización de los modelos resultantes tipo .h5 ----------------------------------
import tensorflow as tf
from tensorflow.keras.models import load_model
import h5py

# Cargar el modelo
model_path = 'cnn_model_gfc.h5'  # archivo .h5 de ejemplo
model = load_model(model_path)

# Mostrar un resumen del modelo
model.summary()

# Acceder a los pesos del modelo
with h5py.File(model_path, 'r') as f:
    for layer_name in f['model_weights']:
        print(layer_name)
        for weight_name in f['model_weights'][layer_name]:
            print(f['model_weights'][layer_name][weight_name])

from tensorflow.keras.utils import plot_model

# Guardar un gráfico del modelo
plot_model(model, to_file='modelo_arquitectura.png', show_shapes=True, show_layer_names=True)

for layer in model.layers:
    weights = layer.get_weights()  # esto devuelve una lista de numpy arrays
    print(f"Pesos de la capa {layer.name}:")
    print(weights)

#------------------ Scripts de la configuración de Django en la carpeta del proyecto ALZDET en repositorio GitHub-------------------
```

In [ ]: