



UNIVERSIDADE DA CORUÑA

Universidade de Vigo

Trabajo Fin de Máster

Análisis estadístico y clasificación automática de datos de Tomografía de Emisión por Positrones en Enfermedad de Alzheimer

Juan Antonio Arias López

Máster en Técnicas Estadísticas

Curso 2022-2023

Propuesta de Trabajo Fin de Máster

Título en galego: Análise estatístico e clasificación automática de datos de Tomografía de Emisión por Positrons na Enfermidade de Alzheimer
Título en español: Análisis estadístico y clasificación automática de datos de Tomografía de Emisión por Positrones en Enfermedad de Alzheimer
English title: Statistical analysis and automatic classification of Positron Emission Tomography data in Alzheimer's disease
Modalidad: Modalidad A
Autor/a: Juan Antonio Arias López, Universidad de A Coruña
Director/a: Rubén Fernández Casal, Universidad de A Coruña; Manuel Oviedo de la Fuente, Universidad de A Coruña
Breve resumen del trabajo: <p>La Enfermedad de Alzheimer supone un gran impacto en la sociedad actual y, sin cura, el diagnóstico precoz es clave. Este proyecto usará datos de Tomografía de Emisión por Positrones y técnicas de Aprendizaje Estadístico para clasificar pacientes como sanos o enfermos, tratando de mejorar métodos existentes.</p>

Don/doña Rubén Fernández Casal, profesor contratado doctor de la Universidad de A Coruña y don/doña Manuel Oviedo de la Fuente, profesor contratado interino de sustitución de la Universidad de A Coruña informan que el Trabajo Fin de Máster titulado

Análisis estadístico y clasificación automática de datos de Tomografía de Emisión por Positrones en Enfermedad de Alzheimer

fue realizado bajo su dirección por don/doña Juan Antonio Arias López para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En A Coruña, a 5 de Junio de 2023.

El/la director/a:
Don/doña Rubén Fernández Casal

El/la director/a:
Don/doña Manuel Oviedo de la Fuente

El/la autor/a:
Don/doña Juan Antonio Arias López

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, [Disposición 2978 del BOE núm. 48 de 2022](#)), **el/la autor/a declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,... sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Agradecimientos

A mi abuela, por coser a mano la americana con la que defiende este TFM, una prenda que ni Amancio Ortega podría pagar. También a Carmen Cadarso, que me metió en este jaleo y no llegó a verme salir de él, pero nunca dudó de que lo haría. Por último, gracias a Dios en toda su insignificancia, por darle a estos grandes simios unos apetitos de bestias y una tecnología de dioses, una combinación que el tiempo dirá si fue la más acertada.

Índice general

Resumen	XI
1. Introducción	1
1.1. Antecedentes y motivación	1
1.2. La enfermedad de Alzheimer	2
1.3. Técnicas de aprendizaje estadístico	3
1.3.1. Árboles de decisión	3
1.3.2. Bosques aleatorios	5
1.3.3. Máquinas de Vectores de Soporte	6
1.3.4. Aprendizaje Profundo	8
1.4. Análisis de datos funcionales	9
1.5. Planteamiento del problema	11
1.6. Objetivos y preguntas de investigación	12
2. Datos de Neuroimagen	13
2.1. Descripción del conjunto de datos	13
2.2. Preprocesamiento de datos de neuroimagen	15
3. Resultados	17
3.1. Árboles de decisión	19
3.2. Bosques Aleatorios	20
3.3. Máquinas de vectores de soporte	22
3.4. Aprendizaje profundo	23
3.5. Aprendizaje profundo con datos funcionales	24
4. Discusión	29
4.1. Interpretación de resultados	29
4.2. Limitaciones	31
5. Conclusiones	33
5.1. Contribuciones al diagnóstico del Alzheimer y al Aprendizaje Estadístico	33
5.2. Recomendaciones para futuras investigaciones	33
A. Código R	35
Bibliografía	51
Índice de figuras	53
Índice de cuadros	55
Acrónimos	57

Resumen

Resumen en español

Las enfermedades neurodegenerativas como la enfermedad de Alzheimer se están estableciendo como algunas de las enfermedades de mayor impacto sobre la sociedad actual. Como condiciones altamente debilitantes y que afectan en especial a la tercera edad, se las empieza a considerar como la Gran Atadura para las sociedades occidentales, caracterizadas por una gran longevidad y progresivo envejecimiento de la población. Dado que a menudo no existe una cura para dichas enfermedades -solamente tratamientos paliativos- el diagnóstico precoz es la prioridad actual. En este proyecto utilizaremos datos de Tomografía de Emisión por Positrones (PET), correspondientes a pacientes sanos y enfermos, para diseñar un sistema de aprendizaje estadístico (*Statistical Learning*) que, a partir de técnicas de aprendizaje profundo (*Deep Learning*) e ingeniería de características basada en resultados de técnicas de datos funcionales, permita discernir con precisión entre pacientes sanos y enfermos a partir de sus imágenes PET cerebrales. Para ello, se implementará el modelo propuesto junto con otros modelos alternativos para comparar su eficiencia y comportamiento en una tarea de clasificación.

English abstract

Neurodegenerative diseases such as Alzheimer's disease are establishing themselves as some of the diseases with the greatest impact on today's society. As highly debilitating conditions that particularly affect the elderly, they are coming to be seen as the Great Bind for Western societies characterised by high longevity and a progressively ageing population. As there is often no cure for such diseases -only palliative treatments- early diagnosis is the current priority. In this project we will use Positron Emission Tomography (PET) data, corresponding to healthy and diseased patients, to design a Statistical Learning system that, based on Deep Learning techniques and feature engineering based on results from functional data analysis, allows to accurately discern between healthy and diseased patients from their brain PET images. To this end, the proposed model will be compared against other alternative models to compare their efficiency and behaviour in a classification task.

Capítulo 1

Introducción

1.1. Antecedentes y motivación

Las enfermedades neurodegenerativas, como la enfermedad de Alzheimer (*Alzheimer's Disease*; AD, por sus siglas en inglés), representan un creciente desafío para la sociedad actual, especialmente en las sociedades occidentales, caracterizadas por una creciente longevidad y un progresivo envejecimiento de la población. El AD causa un gran impacto en la calidad de vida de las personas afectadas, sus familias, y el sistema de salud. En este contexto, el diagnóstico temprano es fundamental para poder ofrecer tratamientos paliativos y mejorar la calidad de vida de los pacientes, ya que no existe una cura definitiva (Hort et al., 2012).

El diagnóstico de AD se ha llevado a cabo tradicionalmente utilizando software como Statistical Parametric Mapping (SPM), desarrollado en la década de los 90, usado sobre técnicas de neuroimagen como la Tomografía de Emisión por Positrones (*Positron Emission Tomography*; PET, por sus siglas en inglés) (Friston, 2007). A pesar de la gran cantidad de datos de neuroimagen disponibles en la actualidad, las técnicas de análisis de estos datos no han evolucionado al mismo ritmo, lo que puede estar lastrando la capacidad de diagnóstico y tratamiento tempranos.

En este escenario, las técnicas de aprendizaje estadístico (*Statistical Learning*; SL, por sus siglas en inglés) o *Machine Learning* representan un enfoque prometedor para mejorar el diagnóstico automático del AD. Estas técnicas pueden complementar y asistir al profesional clínico en la toma de decisiones, proporcionando información valiosa a partir de la gran cantidad de datos de neuroimagen disponibles. Estudios previos han demostrado la utilidad de estos enfoques en la mejora del diagnóstico, logrando resultados significativamente superiores a los métodos tradicionales (Castellazzi et al., 2020).

En investigaciones anteriores (Arias-Lopez et al., 2021, 2022), el autor ha desarrollado técnicas de análisis de datos funcionales (*Functional Data Analysis*; FDA, por sus siglas en inglés) aplicados a neuroimagen consistentes en la obtención de imágenes medias estimadas y en el cálculo de bandas de confianza simultáneas para la diferencia en actividad PET entre grupos de pacientes, tanto controles como patológicos. A través de estas imágenes, se pueden identificar las áreas del cerebro cuya actividad -promediada entre un gran número de pacientes- cae fuera de los intervalos de confianza estimados, lo cual indica variaciones en los valores de PET atribuibles a la pérdida neuronal por efecto de la AD. Esta técnica ha sido probada como superior a SPM a la hora de trabajar con grupos y comparar grandes volúmenes de datos (Arias-Lopez et al., 2021) y será utilizada en este estudio como referencia para conocer las áreas de mayor relevancia a la hora de realizar un diagnóstico diferencial, áreas que se utilizarán para generar nuevas variables (i.e., *ingeniería de características*) con las que se tratará de mejorar los resultados de un algoritmo de aprendizaje profundo en una tarea de clasificación de pacientes.

Por lo tanto, el objetivo primario (ver Sección 1.6) de este trabajo de fin de máster es desarrollar un algoritmo de aprendizaje profundo (*Deep Learning*) que incorpore variables adicionales generadas mediante ingeniería de características en base a las imágenes obtenidas por el autor mediante FDA

en estudios previos ([Arias-Lopez et al., 2021](#)). De esta manera se trata de proporcionar información adicional sobre las áreas más relevantes para el diagnóstico utilizando resultados de FDA como guía. La combinación de técnicas de SL, eficientes a la hora de clasificar individuos, con las investigaciones del autor en FDA, centradas en el trabajo con grandes grupos, tiene el potencial de mejorar significativamente los resultados obtenidos por el uso exclusivo de algoritmos de SL, mejorando la capacidad de diagnóstico diferencial en base a escáneres cerebrales PET.

1.2. La enfermedad de Alzheimer

Esta enfermedad fue descubierta por el médico alemán Alois Alzheimer en 1906, cuando estudió el caso de una paciente llamada Auguste Deter ([Hippius and Neundorfer, 2003](#)). En aquel entonces, la enfermedad fue confundida con otras afecciones psiquiátricas o neurológicas, lo que dificultó su identificación y estudio. La falta de tecnología y conocimientos en neurociencia en esa época también contribuyeron a que no fuese descubierta con anterioridad ([Zerr, 2015](#)).

La sintomatología de la AD se caracteriza principalmente por una pérdida progresiva de memoria, desorientación, deterioro cognitivo, y cambios de personalidad ([Lane et al., 2018](#)). Estos síntomas se deben al daño y a la muerte progresiva de neuronas en regiones cerebrales clave para el aprendizaje, la orientación, y la memoria como el hipocampo, la corteza entorrinal, y otras áreas de la corteza cerebral ([Wenk, 2006](#)).

En cuanto a la etiología, aún no se ha identificado una causa específica. Sin embargo, se han propuesto varias teorías no excluyentes entre las que se encuentran la acumulación de placas amiloides, la formación de ovillos neurofibrilares de proteína Tau, la inflamación, y el estrés oxidativo en el cerebro ([Breijyeh and Karaman, 2020](#)). También se han identificado factores genéticos y ambientales que pueden influir en el desarrollo de la enfermedad.

El pronóstico de la AD es ampliamente desfavorable, ya que actualmente no existe ni una cura ni un tratamiento eficaces. De hecho, el tratamiento se centra en aliviar los síntomas -no las causas- y trata de retrasar la progresión de la enfermedad lo más posible, aunque los resultados varían en gran medida entre los pacientes. La esperanza de vida tras el diagnóstico es también variable, pero en promedio es de aproximadamente 8 a 10 años ([Zhang et al., 2021](#)). El AD en sí mismo no mata directamente al paciente, pero la pérdida de la función cognitiva y motora eventualmente conduce a complicaciones como infecciones, neumonía, edema, y problemas para tragar, lo que acaba resultando fatal de una forma u otra ([Lane et al., 2018](#)).

A nivel global, se estima que en el 2020 unos 50 millones de personas padecían AD ([Breijyeh and Karaman, 2020](#)). Se trata de una enfermedad que afecta ligeramente más a mujeres que a hombres y es más común en personas mayores de 65 años, aunque existen variantes *early-onset* o de desarrollo temprano, a menudo debido a causas genéticas. Entre los factores de riesgo se incluyen la edad avanzada, antecedentes familiares, ciertos factores de estilo de vida como la falta de actividad física y mental, y trastornos depresivos. La adopción de hábitos saludables, como una dieta equilibrada, ejercicio regular, y actividades cognitivas, puede ayudar a prevenir la enfermedad ([Zhang et al., 2021](#)).

El diagnóstico de AD se basa en pruebas neuropsicológicas, análisis de líquido cefalorraquídeo, y técnicas de neuroimagen como la PET y la resonancia magnética ([Jalbert et al., 2008](#)). Sin embargo, a menudo se realiza un diagnóstico tardío debido a la gran plasticidad y reserva cognitiva del cerebro, que hace que los pacientes no acudan a un profesional médico hasta que los síntomas son severos y el daño neuronal está ya extendido. Dadas las tendencias actuales de envejecimiento de la población, se estima que el número de pacientes con AD podría alcanzar los 130 millones en 2050 ([Prince et al., 2016](#)). Esto subraya la importancia de la investigación y el desarrollo de nuevas terapias y estrategias de prevención para abordar esta creciente crisis de salud pública. No obstante, en la actualidad la única respuesta existente consiste en un diagnóstico y un tratamiento tempranos para ralentizar el avance de la enfermedad.

1.3. Técnicas de aprendizaje estadístico

Las técnicas de aprendizaje estadístico (*Statistical Learning*; SL por sus siglas en inglés) son un conjunto de métodos matemáticos y computacionales que se fundamentan en la teoría de la probabilidad y la estadística para realizar inferencias, predicciones, y decisiones a partir de datos. Estas técnicas pueden ser aplicadas en diversos problemas, tales como la regresión y clasificación, con el objetivo de modelar relaciones entre variables y realizar predicciones sobre datos no observados previamente (Has-tie et al., 2009). Ejemplos de aplicaciones reales incluyen el diagnóstico médico, la detección de fraude en transacciones financieras, el reconocimiento de voz, o la recomendación de productos en comercio electrónico.

Los primeros pasos en SL pueden ser rastreados a la segunda mitad del siglo XX, con científicos como Alan Turing y Frank Rosenblatt, quienes desarrollaron conceptos fundamentales en el campo. Turing propuso la idea de una máquina que pudiera aprender a partir de datos (Millican, 2021), mientras que Rosenblatt introdujo el perceptrón, un modelo lineal de clasificación binaria (Rosenblatt, 1960). Los primeros casos prácticos en este campo incluyen la identificación de caracteres ópticos y la clasificación de especies de plantas a partir de medidas morfológicas.

Entre los métodos más avanzados de aprendizaje estadístico, encontramos técnicas como los árboles de decisión (*Decision Trees*), los bosques aleatorios (*Random Forests*), las máquinas de soporte de vectores (*Support Vector Machines*; SVM, por sus siglas en inglés), o el aprendizaje profundo (*Deep Learning*). Estos métodos avanzados ofrecen ventajas sobre las técnicas iniciales, como una mayor capacidad de modelado y mejor desempeño en problemas no lineales.

A día de hoy, el SL -y en concreto, el aprendizaje profundo- se han consolidado como áreas fundamentales en la investigación y la industria. Las técnicas de SL se aplican en una amplia variedad de campos, como la medicina, la ingeniería, la física, la biología y las ciencias sociales, entre otros. Además, han surgido enfoques complementarios, como el aprendizaje por refuerzo, el aprendizaje no supervisado, y el aprendizaje semi-supervisado, que amplían las capacidades y aplicaciones de estas técnicas (Gocheva-Ilieva et al., 2021).

En cuanto a las perspectivas futuras, se espera que el SL siga evolucionando y expandiéndose a nuevos dominios y problemas. Se espera un crecimiento en áreas como la inteligencia artificial explicable, que busca desarrollar modelos de aprendizaje profundo más interpretables y transparentes. Además, la investigación en la eficiencia computacional es de creciente importancia, dada la necesidad de reducir el coste asociado a al entrenamiento de modelos de gran escala. Por último, la discusión ética y la responsabilidad en el desarrollo y aplicación de estas técnicas son temas críticos que seguirán influyendo en el campo, ya que el impacto de estas tecnologías en la sociedad se vuelve cada vez más evidente y significativo (Kelleher, 2019).

A continuación se procede a explicar con detalle las técnicas de SL que serán utilizadas en este estudio para comparar la eficiencia del modelo propuesto con otros ya establecidos en la literatura.

1.3.1. Árboles de decisión

Los árboles de decisión son técnicas de aprendizaje estadístico que se originaron en la década de 1970 como una alternativa flexible a los métodos estadísticos tradicionales, que a menudo se basaban en supuestos de linealidad y normalidad. Aunque su rendimiento predictivo suele ser limitado, los árboles de decisión forman la base de métodos más avanzados, como el *bagging* mediante el uso de bosques aleatorios que se explicará en la siguiente sección.

El concepto subyacente de los árboles de decisión es la segmentación del espacio de las variables predictoras en regiones simples, que se pueden representar mediante un árbol binario. Se comienza con un nodo inicial que representa todo el conjunto de datos de entrenamiento, del cual se derivan dos ramas que dividen el conjunto de datos en dos subconjuntos, cada uno representado por un nuevo nodo. Este proceso se repite hasta que se obtienen los nodos terminales u hojas del árbol, que se utilizan para realizar las predicciones. En cada nodo terminal, la predicción se realiza utilizando la media en un problema de regresión y la moda en un problema de clasificación.

Al final de este proceso, el espacio de las variables predictoras se ha dividido en regiones donde la predicción de la variable de respuesta es constante. Si la relación entre las variables predictoras y la variable de respuesta no se puede describir adecuadamente, la precisión predictiva del árbol será limitada. Por lo tanto, la simplicidad del modelo es su principal ventaja, pero también su principal limitación.

En un problema de clasificación, la homogeneidad de los nodos terminales implicaría que en cada uno de ellos sólo hay elementos de una clase, lo que se conoce como pureza de los nodos. En la práctica, esto siempre se puede lograr construyendo árboles suficientemente profundos, con muchas hojas. Sin embargo, esta solución puede dar lugar a un modelo sobreajustado y de difícil interpretación. Por lo tanto, es necesario encontrar un equilibrio entre la complejidad del árbol y la pureza de los nodos terminales.

La metodología CART (Classification & Regression Trees), introducida por [Breiman et al. \(1984\)](#), es la más popular para la construcción de árboles de decisión. Esta metodología es válida tanto para problemas de regresión, donde la variable de respuesta es numérica, como para problemas de clasificación, donde la variable de respuesta es categórica, que es nuestro caso.

El modelo de árboles de decisión se construye utilizando un conjunto de entrenamiento. Este proceso implica la división del espacio de las variables predictoras en J regiones, denotadas como R_1, R_2, \dots, R_J . Para cada región, se calcula una constante que corresponde a la media de la variable de respuesta Y para las observaciones de entrenamiento que caen dentro de esa región. Estas constantes se utilizan posteriormente para realizar predicciones para nuevas observaciones.

La elección de cómo se divide el espacio de las variables predictoras es crucial. Para tomar esta decisión, utilizamos la suma de los residuos al cuadrado (*Residual Sqared Sum*; RSS, por sus siglas en inglés) como criterio de error. Dado que modelamos la respuesta en cada región como una constante, en la región R_j buscamos minimizar $\min_{c_j} \sum_{i \in R_j} (y_i - c_j)^2$. Este mínimo se alcanza en la media de las respuestas y_i (del conjunto de entrenamiento) en la región R_j , que denotamos como \hat{y}_{R_j} . Por lo tanto, buscamos seleccionar las regiones R_1, R_2, \dots, R_J que minimicen el RSS, que se define como:

$$RSS = \sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \quad (1.1)$$

Sin embargo, éste problema es computacionalmente intratable en la práctica, por lo que necesitamos simplificarlo. El método CART proporciona una solución a este problema al buscar un equilibrio entre el rendimiento y la simplicidad del modelo. En lugar de buscar entre todas las particiones posibles, el método CART sigue un proceso iterativo en el que realiza cortes binarios. En la primera iteración, se consideran todos los datos.

Una variable explicativa X_j y un punto de corte s definen dos hiperplanos $R_1 = X|X_j \leq s$ y $R_2 = X|X_j > s$. Buscamos los valores de j y s que minimicen la siguiente expresión:

$$\sum_{i \in R_1} (y_i - \hat{y}_{R_1})^2 + \sum_{i \in R_2} (y_i - \hat{y}_{R_2})^2 \quad (1.2)$$

A diferencia del problema original, este se resuelve de manera eficiente. Luego, repetimos el proceso en cada una de las dos regiones R_1 y R_2 , y continuamos de esta manera hasta que se cumpla un criterio de parada.

Los árboles de decisión realizan dos concesiones importantes en su construcción: limitan la forma de las particiones del espacio de las variables predictoras y utilizan un criterio de error *greedy*, es decir, minimizan el RSS en cada iteración sin considerar el error en iteraciones futuras.

El proceso de construcción puede detenerse cuando se alcanza una profundidad máxima, aunque lo más común es exigir un número mínimo de observaciones para dividir un nodo. Si el árbol resultante es demasiado grande, el modelo será demasiado complejo, lo que dificultará su interpretación y provocará un sobreajuste de los datos. Por otro lado, si el árbol es demasiado pequeño, tendrá menos varianza pero más sesgo. Por lo tanto, es necesario encontrar un equilibrio entre sesgo y varianza.

Podar un árbol implica colapsar cualquier cantidad de sus nodos internos, dando lugar a un subárbol más pequeño que el árbol original. Aunque el árbol completo minimiza el error en el conjunto de entrenamiento, nuestro objetivo es encontrar el subárbol que minimiza el error en el conjunto de validación.

Para controlar el tamaño del árbol, se utiliza un hiperparámetro α . Dado un subárbol T con R_1, R_2, \dots, R_t nodos terminales, consideramos como medida del error el RSS más una penalización que depende de α y del número de nodos terminales t , como se muestra a continuación:

$$RSS_\alpha = \sum_{j=1}^t \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha t \quad (1.3)$$

Para cada valor del parámetro α , existe un único subárbol más pequeño que minimiza este error. A medida que se incrementa α , se penalizan los subárboles con muchos nodos terminales, dando lugar a una solución más pequeña. El algoritmo consiste en ir colapsando nodos de forma sucesiva de modo que produzca el menor incremento en el RSS . Esto da lugar a una sucesión finita de subárboles que contiene, para todo α , la solución. Para seleccionar el valor del hiperparámetro α lo más habitual es emplear validación cruzada (*cross-validation*), un procedimiento de evaluación de modelos que particiona el conjunto de datos en subconjuntos, realizando el análisis en uno (o varios) subconjunto(s) y validando el análisis en el subconjunto restante, a fin de limitar problemas como el sobreajuste.

Aplicándolo a problemas de clasificación, donde la variable respuesta toma valores discretos $1, 2, \dots, K$, que identifican las K categorías del problema, no se podrá utilizar RSS como medida del error. Será necesario buscar una medida del error adaptado. Fijada una región, vamos a denotar por \hat{p}_k , con $k = 1, 2, \dots, K$, a la proporción de observaciones en la región que pertenecen a la categoría k . Existen tres medidas habituales del error en la región:

- Proporción de errores de clasificación: $1 - \max_k(\hat{p}_k)$
- Índice de Gini: $\sum_{k=1}^K \hat{p}_k(1 - \hat{p}_k)$
- Entropía (cross-entropy): $-\sum_{k=1}^K \hat{p}_k \log(\hat{p}_k)$

La formación de un árbol de decisión implica dos pasos: una fase de crecimiento en que se dividen los datos utilizando criterios como el índice de Gini o la entropía para crear el árbol, y una fase de poda, donde se eliminan las ramas que minimizan la mejora en la precisión para evitar el sobreajuste. Aunque la proporción de errores de clasificación es la medida del error más intuitiva, en la práctica sólo se utiliza para la fase de poda. En el cálculo de esta medida sólo interviene $\max_k(\hat{p}_k)$, mientras que en las medidas alternativas intervienen las proporciones \hat{p}_k de todas las categorías. Para la fase de crecimiento se utilizan indistintamente el índice de *Gini* o la entropía. Cuando nos interesa el error no en una única región sino en varias, se suman los errores de cada región previa ponderación por el número de observaciones que hay en cada una de ellas.

En definitiva, los árboles de decisión son una herramienta flexible para la clasificación. Aunque implican ciertas concesiones como la restricción en la forma de las particiones y el uso de un criterio de error *greedy*, su simplicidad y su capacidad para manejar tanto variables predictoras numéricas como categóricas los hacen muy útiles en la práctica. Finalmente, su estructura permite una interpretación intuitiva de los resultados, aunque habitualmente estos tienden a distar del óptimo.

1.3.2. Bosques aleatorios

Los bosques aleatorios son un método de SL basado en la combinación de árboles de decisión mediante *bagging*. El *bagging* es un procedimiento que se utiliza para disminuir la varianza en un método de SL cuya idea principal es fusionar varios métodos de predicción básicos con una capacidad predictiva limitada para obtener un método de predicción más robusto. Este método, propuesto por

Breiman (1996), fue uno de los primeros métodos de conjunto que se utilizó. Es un método general de reducción de la varianza que utiliza el *bootstrap* junto con un modelo de regresión o de clasificación, como un árbol de decisión en nuestro caso concreto.

Si disponemos de muchas muestras de entrenamiento, podemos utilizar cada una de ellas para entrenar un modelo que luego nos servirá para hacer una predicción. De esta manera, tendremos tantas predicciones como modelos y tantas predicciones como muestras de entrenamiento. El procedimiento tiene dos ventajas importantes: simplifica la solución y reduce significativamente la varianza. El problema es que, en la práctica, suele disponerse de una única muestra de entrenamiento. Aquí es donde entra en juego el *bootstrap* de las observaciones, una técnica especialmente útil para reducir la varianza. Lo que se hace es generar cientos de muestras a partir de una muestra de entrenamiento, y luego utilizar cada una de estas remuestras como una nueva muestra de entrenamiento.

Para los árboles de regresión, se hacen crecer muchos árboles y se calcula la media de las predicciones. En el caso de los árboles de clasificación, lo más sencillo es sustituir la media por la moda y utilizar el criterio del voto mayoritario: cada modelo tiene el mismo peso y, por lo tanto, cada modelo aporta un voto. Un dato que no se utiliza para construir un árbol se denomina un dato fuera de la bolsa (*Out of Bag*; OOB, por sus siglas en inglés). De esta manera, para cada observación se pueden utilizar los árboles para los que esa observación es OOB para generar una única predicción para ella. Al repetir el proceso para todas las observaciones, se obtiene una medida del error.

Aumentar mucho el número de árboles no mejora las predicciones, aunque tampoco aumenta el riesgo de sobreajuste. No obstante, los costos computacionales sí que aumentan con el número de árboles. Por otro lado, si el número de árboles es demasiado pequeño, puede que se obtengan pocas predicciones OOB para alguna de las observaciones de la muestra de entrenamiento. Además, al utilizar *bagging*, se mejora significativamente la predicción, pero se pierde la interpretabilidad de parámetros e hiperparámetros. Sin embargo, existen formas de calcular la importancia de los predictores. A menudo ocurre que los árboles tienen estructuras muy similares, especialmente en la parte superior, aunque luego se diferencian a medida que se desciende por ellos. Esta característica se conoce como correlación entre árboles y se produce cuando el árbol es un modelo adecuado para describir la relación entre los predictores y la respuesta, y también cuando uno de los predictores es especialmente relevante, por lo que casi siempre va a estar en el primer corte.

En la construcción de cada uno de los árboles que finalmente constituirán el bosque aleatorio, se van haciendo cortes binarios, y para cada corte hay que seleccionar una variable predictora. Antes de hacer cada uno de los cortes, de todas las variables predictoras p , se seleccionan al azar $m < p$ predictores que van a ser los candidatos para el corte. El hiperparámetro de los bosques aleatorios es m , y se puede seleccionar mediante las técnicas habituales. Como puntos de partida razonables se pueden considerar $m = \sqrt{p}$ (para problemas de clasificación) y $m = p/3$ (para problemas de regresión).

En conclusión, los bosques aleatorios son una modificación del *bagging* para el caso de árboles de decisión. Se introduce aleatoriedad en las variables, no sólo en las observaciones y, para evitar dependencias, los posibles predictores se seleccionan al azar en cada nodo (e.g. $m = \sqrt{p}$). Estos métodos dificultan la interpretación, pero a cambio mejoran su precisión y existen métodos para medir la importancia de las variables.

1.3.3. Máquinas de Vectores de Soporte

Las Máquinas de Vectores de Soporte (SVM) son un tipo de clasificador de soporte vectorial y representan un tipo de modelo de clasificación de máximo margen. En el caso más simple, los clasificadores de soporte vectorial tratan de determinar el hiperplano de separación óptimo entre clases de datos, maximizando el margen entre las instancias más cercanas a dicho hiperplano, conocidas como vectores de soporte. Este modelo es altamente adaptable y ha demostrado su eficacia en un amplio espectro de contextos.

El proceso de entrenamiento puede representarse como un problema de optimización:

$$\max_{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n} M \quad (1.4)$$

$$(1.5)$$

Sujeto a:

$$\sum_{j=1}^p \beta_j^2 = 1, \quad (1.6)$$

$$y_i(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \dots + \beta_p x_{pi}) \geq M - \epsilon_i, \quad \forall i \quad (1.7)$$

$$\sum_{i=1}^n \epsilon_i \leq K, \quad (1.8)$$

$$\epsilon_i \geq 0, \quad \forall i \quad (1.9)$$

Aquí, ϵ_i son las llamadas variables de holgura (*slack variables*), y K es un hiperparámetro que define la tolerancia al error. Concretamente, el valor de K puede interpretarse como una penalización por la complejidad del modelo, equilibrando entre el sesgo y la varianza. En el caso extremo de utilizar $K = 0$, tendríamos el caso de un *hard margin classifier*. Su selección óptima se puede llevar a cabo utilizando técnicas como el bootstrap o la validación cruzada.

En la práctica, se utiliza una formulación equivalente para simplificar los cálculos, donde se introduce un nuevo parámetro C :

$$\min_{\beta_0, \beta} \frac{1}{2} |\beta|^2 + C \sum_{i=1}^n \epsilon_i \quad (1.10)$$

Tal que:

$$y_i(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \dots + \beta_p x_{pi}) \geq 1 - \epsilon_i, \quad \forall i, \quad (1.11)$$

$$\epsilon_i \geq 0, \quad \forall i, \quad (1.12)$$

Aquí, C juega un papel inverso al de K : penaliza las malas clasificaciones, y cuando es muy alto (equivalente a $K = 0$), se obtiene un *hard margin classifier*.

Los vectores de soporte son los datos de entrenamiento que no solo están a una distancia M del hiperplano, sino también aquellos que están mal clasificados o que están a una distancia menor a M .

Los clasificadores de soporte de vectores, al igual que ocurría con los árboles de decisión, son eficaces en la medida en que los datos se separen de manera adecuada. Para conjuntos de datos que no son linealmente separables, una estrategia común es mapear los datos a un espacio de características de mayor dimensión, donde puedan ser separados linealmente, este es el caso de los SVM. Esto se consigue utilizando funciones kernel, que son funciones de los datos que dependen de las variables predictoras X_1, X_2, \dots, X_n .

Dicha transformación se hace evidente cuando se reemplaza el producto escalar en la fórmula de decisión de la SVM:

$$m(x) = \beta_0 + \sum_{i \in S} \alpha_i \mathbf{x}^t \mathbf{x}_i \quad (1.13)$$

Explícitamente en los SVM se reemplaza el producto escalar por una función kernel $K(\mathbf{x}, \mathbf{x}_i)$:

$$m(x) = \beta_0 + \sum_{i \in S} \alpha_i K(\mathbf{x}, \mathbf{x}_i) \quad (1.14)$$

Algunas funciones kernel comúnmente empleadas incluyen:

- Kernel lineal: $K(x, y) = x^T y$
- Kernel polinómico: $K(x, y) = (1 + \gamma x^T y)^d$
- Kernel radial: $K(x, y) = \exp(-\gamma |x - y|^2)$
- Tangente hiperbólica: $K(x, y) = \tanh(1 + \gamma x^T y)$

La elección del kernel y sus parámetros (γ, d) , así como el parámetro C , es crucial para evitar el sobreajuste y maximizar la efectividad de la SVM.

1.3.4. Aprendizaje Profundo

El aprendizaje profundo, una rama del SL, se basa en el uso de redes neuronales artificiales (*Artificial Neural Networks*; ANN, por sus siglas en inglés) con múltiples capas ocultas para aprender representaciones jerárquicas de los datos. Estas redes neuronales, cuyo concepto original se debe a [McCulloch and Pitts \(1943\)](#), se destacan por su capacidad para manejar un gran número de parámetros, lo que las hace adecuadas para problemas con estructuras subyacentes muy complejas.

En términos matemáticos, una ANN puede ser vista como un aproximador universal de funciones. Dada una función objetivo $f^*(x)$, la meta de una red neuronal es aprender una función $f(x; \theta)$ que minimice la discrepancia entre $f^*(x)$ y $f(x; \theta)$ para todos los x en el conjunto de entrenamiento, donde θ representa los parámetros de la red.

$$\min_{\theta} \sum_{i=1}^n \mathcal{L}(y_i, f(x_i; \theta)) \quad (1.15)$$

Aquí, $\mathcal{L}(y_i, f(x_i; \theta))$ es una función de pérdida que mide la discrepancia entre la etiqueta verdadera y_i y la predicción de la red $f(x_i; \theta)$ para un ejemplo de entrenamiento x_i .

Un modelo básico de red neuronal realiza dos transformaciones de los datos y consta de tres capas (*layers*): una capa de entrada consistente en las variables originales $X = (X_1, X_2, \dots, X_p)$, una capa oculta con M nodos y una capa de salida con la predicción final $m(X)$.

$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]} \quad (1.16)$$

$$a^{[l]} = g^{[l]}(z^{[l]}) \quad (1.17)$$

Aquí, $a^{[l-1]}$ es el vector de activaciones de la capa anterior, $W^{[l]}$ y $b^{[l]}$ son la matriz de pesos y el vector de sesgo para la capa l , respectivamente, y $g^{[l]}$ es la función de activación de la capa l .

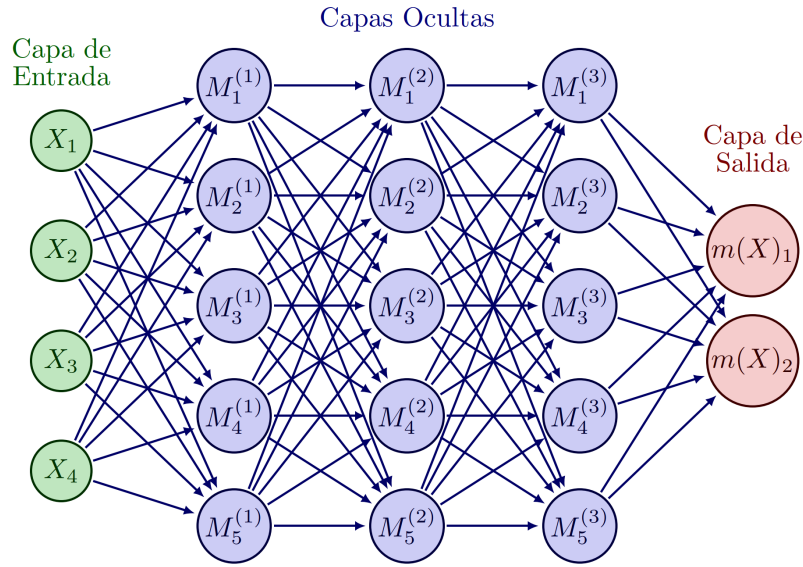


Figura 1.1: Ejemplo de la estructura de una red neuronal superficial con tres capas ocultas.

Un modelo más avanzado puede verse en la Figura 1.1, donde la red neuronal cuenta con 5 capas: una de entrada, tres capas ocultas que generan tres transformaciones, y una capa de salida.

El diseño y entrenamiento de una ANN suelen requerir más tiempo y experimentación que otros algoritmos de SL. La optimización es un problema complejo debido al gran número de hiperparámetros involucrados. Por esta razón, las redes neuronales son demandantes desde el punto de vista computacional y, hasta fechas recientes, no ha sido viable utilizarlas con un número elevado de capas, lo que se conoce como *Deep Learning* o *Deep Neural Networks*, lo que en español se conoce como *aprendizaje profundo*.

Estos modelos son especialmente útiles en problemas muy complejos, pero no tanto en problemas con pocas observaciones o pocos predictores. Además, para que las redes neuronales tengan un rendimiento aceptable, es necesario contar con tamaños muestrales grandes, debido a que son modelos hiperparametrizados.

A pesar de los desafíos, una de las fortalezas de las redes neuronales es su capacidad para realizar de forma automática la detección de características relevantes en los datos a través de las capas intermedias. Sin embargo, a medida que aumentan las capas, la interpretación del modelo se vuelve cada vez más difícil, llegando a convertirse en una auténtica *caja negra*. Esto plantea desafíos con respecto a la justicia, la responsabilidad, y la transparencia en la aplicación de estos modelos (Ahmad et al., 2020).

1.4. Análisis de datos funcionales

El FDA es una rama emergente de la estadística que se centra en el estudio de datos que se pueden representar como funciones. Este enfoque es especialmente útil cuando los datos son intrínsecamente continuos y existen dependencias entre observaciones cercanas, empleando herramientas como el análisis de series temporales y la teoría de funciones para analizar la variabilidad y estructura de los datos funcionales (Ramsay and Silverman, 2005).

El FDA opera en espacios de Hilbert, que son espacios vectoriales completos con un producto interno. El producto interno permite la proyección ortogonal y la descomposición en componentes independientes, lo que facilita la manipulación de los datos funcionales. Estos espacios cumplen una

serie de propiedades como la linealidad, simetría, y positividad definida, que resultan de gran utilidad en este contexto.

Para el análisis FDA, una representación común de los datos es la expansión en series de bases, donde c_k son coeficientes y $\phi_k(x)$ son funciones base, que pueden ser por ejemplo funciones polinomiales, la serie de Fourier, o B-splines, dependiendo de la naturaleza de los datos.

$$f(x) = \sum_{k=1}^K c_k \phi_k(x) \quad (1.18)$$

El número de funciones base K a considerar en la expansión juega un papel crucial en la calidad de la representación. Un valor pequeño de K podría no capturar suficientemente la complejidad de los datos, mientras que un valor demasiado grande podría dar lugar a sobreajuste. Por tanto, la elección de K es un compromiso entre sesgo y varianza.

La obtención de los coeficientes c_k en la serie se realiza a través de la proyección ortogonal de las funciones de datos en el conjunto de funciones base. Esta proyección, en el sentido del producto interno en el espacio de Hilbert, matemáticamente se expresa como:

$$c_k = \langle f, \phi_k \rangle = \int f(x) \phi_k(x) dx \quad (1.19)$$

Aquí, f es la función de datos, ϕ_k es la k -ésima función base y c_k es el coeficiente correspondiente en la expansión de la serie de base.

El FDA ha encontrado aplicaciones en una amplia gama de campos, incluyendo la biología, la medicina, la economía, y la ingeniería. Las técnicas de FDA son especialmente útiles en casos donde las observaciones se registran en forma de curvas o series temporales, como en la monitorización de signos vitales o la predicción de series financieras. Estas aplicaciones se benefician del enfoque del FDA en la estructura funcional de los datos y su capacidad para modelar la variabilidad inherente en las observaciones (Ramsay and Silverman, 2005; Ferraty and Vieu, 2006).

En particular, y con mayor interés para nuestro objeto de estudio, las técnicas de FDA también pueden ser aplicadas al campo de la imagen médica. La extensión del campo al análisis de datos bidimensionales fue desarrollada recientemente por Wang et al. (2020), quienes propusieron un nuevo enfoque para obtener el valor medio de grupos de imágenes y sus correspondientes intervalos de confianza simultáneos (*Simultaneous Confidence Corridors*; SCCs, por sus siglas en inglés), evitando así problemas típicos de las estimaciones en imagen médica.

El artículo propone un modelo de regresión función-sobre-escalar para el análisis de datos de imágenes. En este modelo, la medición de la imagen en una ubicación específica, $Y_i(z)$, se modela como la suma de una función media, $\mu(z)$, un proceso estocástico que caracteriza las variaciones de la imagen a nivel de sujeto, $\eta_i(z)$, y un término de error, $\sigma(z)\epsilon_i(z)$. Aquí, $\sigma(z)$ es una función determinística positiva, y se asume que $\eta_i(z)$ y $\epsilon_i(z)$ son mutuamente independientes. El modelo se puede expresar en notación matemática de la siguiente manera:

$$Y_i(z) = \mu(z) + \eta_i(z) + \sigma(z)\epsilon_i(z), \quad i = 1, \dots, n, \quad z \in \Omega \quad (1.20)$$

En su artículo, los autores demuestran que este método es estadísticamente consistente y asintóticamente normal bajo condiciones de regularidad estándar. Esta técnica aborda la tarea aplicando *splines* bivariados sobre triangulaciones de Delaunay (Chew, 1989), preservando así las complejas características de estas imágenes (Lai and Wang, 2013).

El procedimiento se extiende al caso de dos muestras, donde se comparan las funciones medias de dos poblaciones de datos de imágenes. En este caso, se consideran dos conjuntos de datos de imágenes, $Y_{1i}(z)$ y $Y_{2j}(z)$, para $i = 1, \dots, n_1$ y $j = 1, \dots, n_2$, respectivamente.

$$Y_{1i}(z) = \mu_1(z) + \eta_{1i}(z) + \sigma_1(z)\epsilon_{1i}(z), \quad i = 1, \dots, n_1, \quad z \in \Omega \quad (1.21)$$

$$Y_{2j}(z) = \mu_2(z) + \eta_{2j}(z) + \sigma_2(z)\epsilon_{2j}(z), \quad j = 1, \dots, n_2, \quad z \in \Omega \quad (1.22)$$

Aquí, $\mu_1(z)$ y $\mu_2(z)$ son las funciones medias de las dos poblaciones, $\eta_{1i}(z)$ y $\eta_{2j}(z)$ son procesos estocásticos que caracterizan las variaciones de la imagen a nivel de sujeto para las dos poblaciones, y $\sigma_1(z)$ y $\sigma_2(z)$ son funciones determinísticas positivas. Al igual que en el caso de una sola muestra, se asume que $\eta_{1i}(z)$, $\epsilon_{1i}(z)$, $\eta_{2j}(z)$ y $\epsilon_{2j}(z)$ son mutuamente independientes.

El objetivo en este caso es hacer inferencias sobre las diferencias entre las funciones medias de las dos poblaciones, es decir, $\mu_1(z) - \mu_2(z)$. Para hacer esto, los autores proponen un procedimiento para construir corredores de confianza simultáneos para la diferencia de las funciones medias sobre una rejilla de Triangulaciones de Delaunay. Para más detalles sobre el fundamento teórico, desarrollo matemático, y evaluación sobre datos simulados, se recomienda referirse a la fuente original ([Wang et al., 2020](#)).

Además, la propuesta de [Wang et al. \(2020\)](#), al considerar los datos como funcionales, redirige el foco de atención de usar los píxeles como unidades analíticas básicas a considerar las imágenes como un todo, calculando luego las SCCs correspondientes. Esta aproximación es más eficiente para la estimación de la incertidumbre asociada que las técnicas analíticas predominantes desplegadas en software como SPM, que siguen el *enfoque masivo univariado* que considera los píxeles como unidades independientes y luego los compara con métodos clásicos como una prueba T , corrigiendo luego el problema de las múltiples comparaciones con enfoques como la corrección de Bonferroni o el *Random Field Theory* ([Worsley et al., 2004](#)).

En conclusión, dada la eficiencia de la técnica de SCCs en la estimación de función media y de la incertidumbre asociada a la interpretación de imágenes, y su capacidad para identificar regiones con variaciones significativas en actividad, se ha optado por utilizarla para complementar el algoritmo de aprendizaje profundo en . Esto se lleva a cabo utilizando los píxeles identificados por las SCCs para realizar ingeniería de características, destacando así aquellos elementos más relevantes para el diagnóstico. Una discusión más detallada se ofrece en la Sección 3.5.

1.5. Planteamiento del problema

El problema que aborda este estudio es la implementación y evaluación de un algoritmo de aprendizaje profundo que integre resultados de FDA mediante ingeniería de características para mejorar la precisión en el diagnóstico de pacientes AD o control, basándose en escáneres PET. El objetivo es evaluar el rendimiento de este algoritmo propuesto en comparación con otros enfoques más convencionales como árboles de decisión, bosques aleatorios, SVM, y aprendizaje profundo sin FDA. La hipótesis es que la sinergia entre FDA y técnicas de aprendizaje profundo permitirá la identificación de patrones complejos latentes, lo que implicará a un diagnóstico más preciso.

La importancia de este estudio radica en su potencial para explorar nuevas vías de mejora en la detección temprana de AD, lo que podría tener un impacto en el desarrollo de futuras herramientas diagnósticas. Además, este trabajo también contribuye al crecimiento y expansión de las técnicas de SL y FDA, explorando nuevas aplicaciones y sinergias entre estas disciplinas emergentes en el ámbito de la bioestadística y la neurociencia. Esto es todavía más relevante al tener en cuenta que los datos de trabajo son obtenidos de pacientes reales, lo que implica una variabilidad y complejidad no presente en bases de datos simuladas o previamente procesadas y manipuladas.

1.6. Objetivos y preguntas de investigación

Una vez valorados los antecedentes y descrita la necesidad de mejorar el diagnóstico precoz de AD mediante técnicas de SL, en este trabajo de fin de máster se plantean los siguientes objetivos y preguntas de investigación:

Objetivos

1. Diseñar un algoritmo de aprendizaje profundo que, utilizando ingeniería de características, incorpore resultados de FDA con el fin de mejorar la precisión en el diagnóstico de AD usando escáneres PET.
2. Comparar el rendimiento del algoritmo propuesto con otros cuatro enfoques más convencionales: árboles de decisión, bosques aleatorios, SVM, y aprendizaje profundo sin FDA.

Preguntas de investigación

1. ¿Cuál de los cinco enfoques (árboles de decisión, bosques aleatorios, SVM, aprendizaje profundo sin FDA y aprendizaje profundo con FDA) presenta el mejor rendimiento para el diagnóstico de AD en nuestra base de datos?
2. ¿Mejora el rendimiento diagnóstico al combinar técnicas de aprendizaje profundo y FDA en comparación con no incorporarlas?

Capítulo 2

Datos de Neuroimagen

En esta sección se describen los datos demográficos y de neuroimagen utilizados en el estudio, incluyendo figuras, visualizaciones de archivos de neuroimagen, y detalles sobre el procesamiento de imágenes para garantizar la comparabilidad entre pacientes.

2.1. Descripción del conjunto de datos

El conjunto de datos sobre el que trabajamos consiste en una serie de variables demográficos que incluyen el código identificador anonimizado del paciente, edad, sexo, y grupo de diagnóstico. Disponemos de un total de 126 pacientes, con una edad media de $\bar{X} = 75,18$ años, una desviación estándar de $\hat{S} = 5,99$ años, un mín = 59 años, un máx = 86 años. En la Figura 2.1a viene desglosada la edad por sexo del paciente. En cuanto al grupo de diagnóstico, tenemos un número total de 51 pacientes en el grupo Alzheimer ($\bar{X} = 74,39$ años, $\hat{S} = 7,25$, mín = 59 años, máx = 86 años) y 75 en el grupo Control ($\bar{X} = 75,78$ años, $\hat{S} = 4,96$ años, mín = 62 años, máx = 86 años). En la Figura 2.2a viene desglosada la edad en función del diagnóstico.

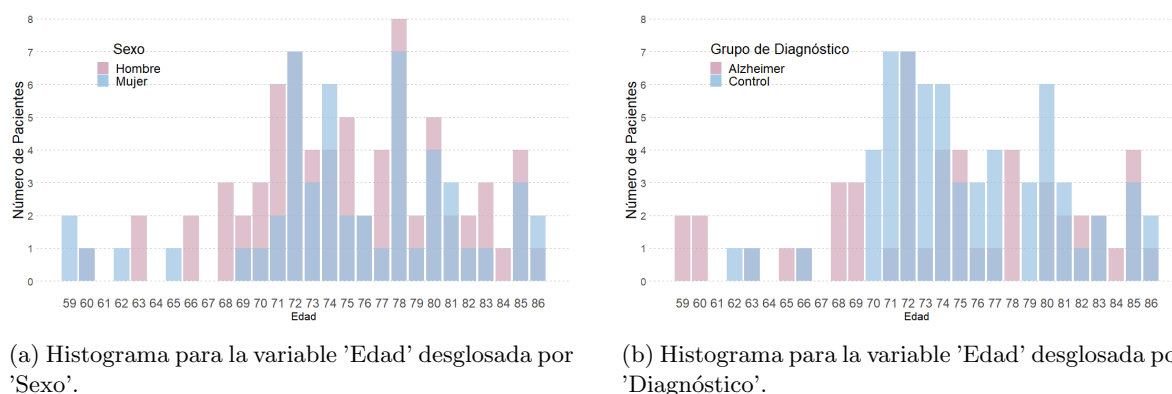


Figura 2.1: Histograma para la variable 'Edad' en función de 'Sexo' y 'Diagnóstico'.

En total se cuenta con un número total de 52 mujeres y 74 hombres (ver Figura 2.1b). El desglose del grupo de diagnóstico por sexos se muestra en la Figura 2.2b.

Estos datos demográficos vienen acompañados de datos de neuroimagen PET en formato Analyze, un formato de archivo ampliamente utilizado en el campo de la neuroimagen para almacenar imágenes tridimensionales. El formato Analyze consta de dos archivos: un archivo de cabecera (.hdr) y uno de imagen (.img). El archivo de cabecera contiene información sobre las dimensiones de la imagen, la

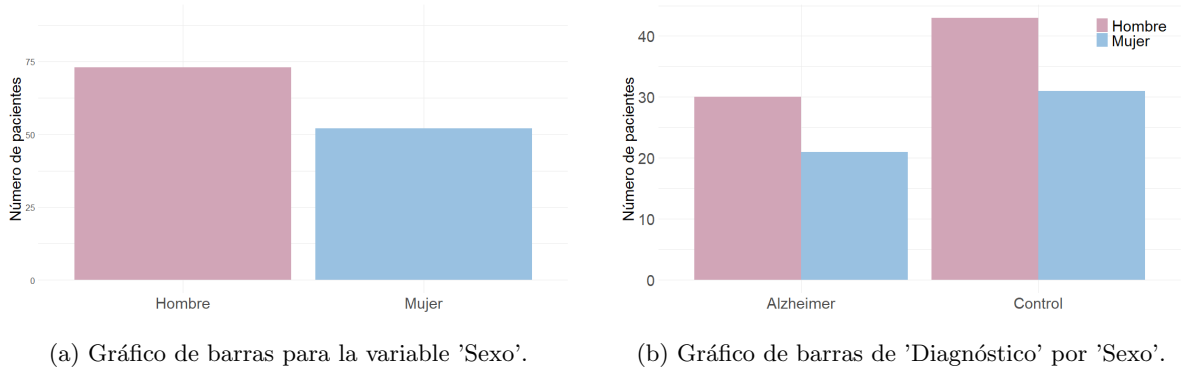


Figura 2.2: Gráficos descriptivos para la variable 'Diagnóstico' y para el desglose por la variable 'Sexo'.

resolución espacial, orientación, y otros metadatos importantes para el procesamiento y visualización de la imagen. Por otro lado, el archivo de imagen contiene los valores de intensidad de los vóxeles (i.e. píxeles tridimensionales) en una matriz que representa la estructura anatómica del cerebro.

Estas imágenes en formato Analyze tienen unas dimensiones de $79 \times 95 \times 79$ (siguiendo una estructura de ejes X , Y , y Z) de las que se extraerá la imagen bidimensional correspondiente a $Z = 30$ para poder compatibilizar con el uso de técnicas de FDA con funcionamiento limitado a dos dimensiones (i.e., SCCs; Wang et al. (2020)). La selección de $Z = 30$ se ha realizado con la intención de cortar por algunas de las estructuras cerebrales que en la literatura se consideran como más relevantes para el diagnóstico de AD (Schöll et al., 2016), como el hipocampo o los córtex temporales. En la Figura 2.3 se puede visualizar uno de estos cortes en $Z = 30$ (a) y las regiones de interés (ROI) superpuestas sobre el plano axial (b), que es el que se utilizará en este estudio. Adicionalmente, se muestra la posición de estas regiones cerebrales en los planos sagital (c) y coronal (d).

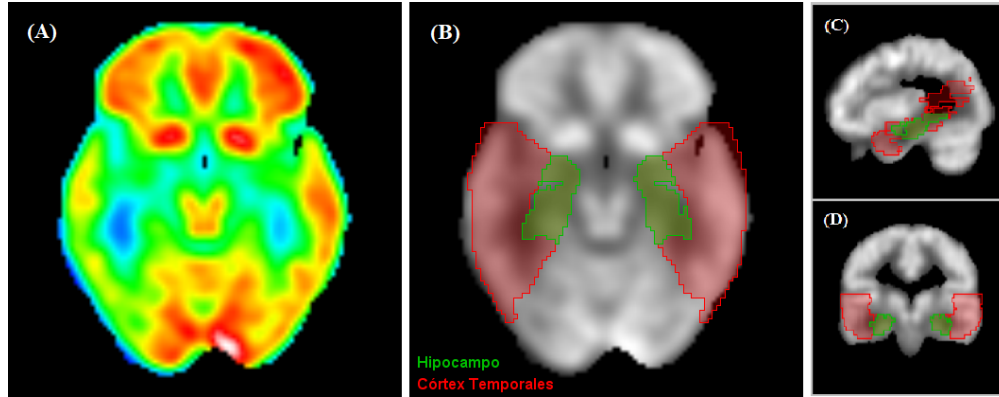


Figura 2.3: Imágenes descriptivas de los datos de neuroimagen PET analizados en este estudio: (a) visualización funcional normal de un archivo de neuroimagen PET, (b) ROI superpuestas sobre el plano axial en $Z = 30$, (c) ROI superpuestas sobre plano sagital central ($X = 37$), y (d) ROI superpuestas sobre plano coronal central ($Y = 47$).

Los datos se recogieron de la base de datos Alzheimer's Disease Neuroimaging Initiative (ADNI), una base de datos de neuroimagen para investigación en el campo del AD y otras enfermedades neurodegenerativas con datos de PET sin procesar provenientes de pacientes reales. El acceso se obtuvo mediante una concesión de uso proporcionada por el Complejo Hospitalario Universitario de Santiago de Compostela. El número final de variables predictoras de que dispondremos será de 7508 por paciente: 7505 píxeles más 3 variables que corresponden al identificador de paciente, la edad, y el sexo.

2.2. Preprocesamiento de datos de neuroimagen

El análisis de imágenes PET ha evolucionado significativamente en las últimas décadas con el desarrollo de técnicas de preprocesado que permiten investigaciones más precisas y comparables. Uno de los paquetes de software más utilizados en el campo de la neuroimagen es SPM, que proporciona herramientas para el análisis de datos de neuroimagen tanto estructural como funcional (Friston, 2007). En esta sección, describimos el proceso completo de preprocesado de neuroimagen PET utilizando en este estudio y realizado con SPM -un *software* manejado con comandos de *Matlab*- incluyendo realineamiento, normalización espacial, normalización de intensidad, y enmascaramiento.

1. **Realineamiento:** El objetivo del realineamiento es corregir las diferencias de movimiento entre las imágenes adquiridas, minimizando las variaciones de posición de los voxeles. Este proceso asegura que todas las imágenes estén alineadas en el mismo espacio.
2. **Normalización espacial:** La normalización espacial, también conocida como "warping", tiene como objetivo registrar las imágenes de un sujeto en un espacio estandarizado de referencia denominado *cerebro estereotático*, una suerte de cerebro idealizado.
3. **Normalización de intensidad:** La normalización de intensidad tiene como objetivo corregir las diferencias en la intensidad de las imágenes, lo que puede deberse a variaciones en el la adquisición o a características anatómicas individuales. En SPM, se utiliza un enfoque de normalización de intensidad basado en la media global del cerebro. Este proceso asegura que las diferencias observadas en las intensidades de las imágenes no sean atribuibles a factores irrelevantes.
4. **Enmascaramiento:** El enmascaramiento es el proceso de selección de una ROI dentro de las imágenes, excluyendo áreas no deseadas o irrelevantes para el análisis. En SPM, se pueden utilizar máscaras binarias creadas a partir de los datos de los sujetos para limitar el análisis a áreas específicas del cerebro. El enmascaramiento permite reducir el ruido y aumentar la sensibilidad en la detección de efectos de interés en el análisis.

En conjunto, el preprocesado de neuroimagen aplicado en este estudio garantiza la máxima calidad y confiabilidad posible para los datos registrados, además de garantizar la comparabilidad voxel-a-voxel entre pacientes. En la Figura 2.4 se pueden ver la imagen en su estado original (2.4a), tras el procesado (2.4b), y tras el enmascaramiento (2.4c).

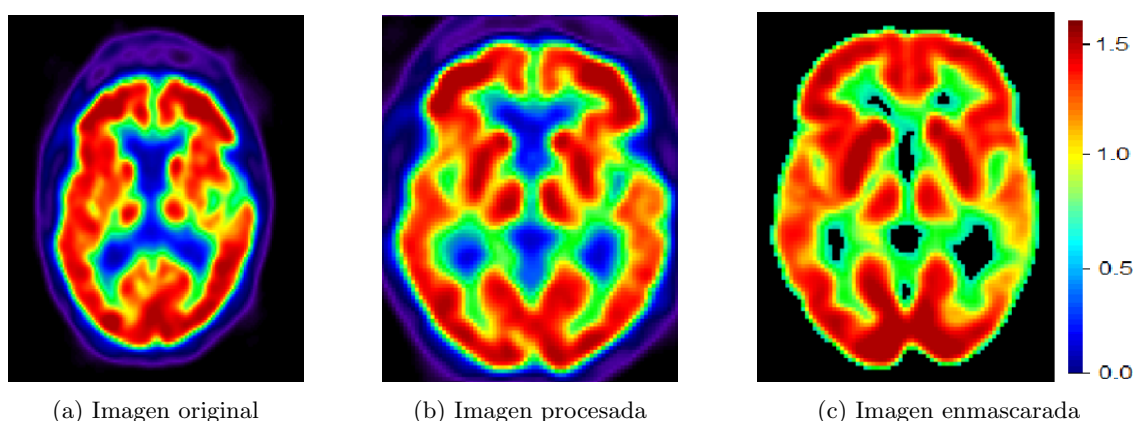


Figura 2.4: Comparación de archivos de neuroimagen PET: original, preprocesada (realineamiento, normalización espacial, y normalización de intensidad), y enmascarada.

Capítulo 3

Resultados

Tras la descripción de los datos de neuroimagen utilizados en este estudio en la sección Sección 2, se procede a presentar y analizar los resultados obtenidos. Esta sección se ha diseñado para proporcionar una visión tanto individual como global de los resultados obtenidos, destacando las métricas de rendimiento que se han considerado más relevantes para evaluar la eficacia de las metodologías implementadas. Estas métricas se han calculado a partir de matrices de confusión como las que se presentan en la Figura 3.1.

Observado/Predicción	Positivo	Negativo
Positivo	Verdaderos Positivos (TP)	Falsos Negativos (FN)
Negativo	Falsos Positivos (FP)	Verdaderos Negativos (TN)

Cuadro 3.1: Matriz de confusión estándar aplicada en este estudio.

Las métricas seleccionadas para evaluar las metodologías propuestas son las siguientes: sensibilidad, especificidad, precisión, precisión balanceada, y el coeficiente Kappa de Cohen.

Sensibilidad La sensibilidad, también conocida como tasa de verdaderos positivos, mide la proporción de positivos reales que son correctamente identificados. Se calcula como:

$$Sensibilidad = \frac{TP}{TP + FN} \quad (3.1)$$

donde TP son los verdaderos positivos y FN son los falsos negativos. Una alta sensibilidad indica que el modelo es eficaz en la detección de los casos positivos. En contextos médicos se suele considerar más importante que la especificidad, dado que habitualmente es preferible un falso positivo que un falso negativo. Oscila entre 0 y 1, donde 1 indica que todos los casos positivos se identificaron correctamente, mientras que un valor de 0 indica que ninguno de los casos positivos se identificó correctamente.

Especificidad La especificidad, o tasa de verdaderos negativos, mide la proporción de negativos reales que son correctamente identificados. Se calcula como:

$$Especificidad = \frac{TN}{TN + FP} \quad (3.2)$$

donde TN son los verdaderos negativos y FP son los falsos positivos. Una alta especificidad indica que el modelo es eficaz en la identificación de los casos negativos y es especialmente útil cuando es importante evitar falsos positivos. Oscila entre 0 y 1, donde un valor de 1 indica que todos los casos negativos se identificaron correctamente, mientras que un valor de 0 indica que ninguno de los casos negativos se identificó correctamente.

Precisión La precisión mide la proporción de identificaciones positivas que fueron realmente correctas. Se calcula como:

$$Precision = \frac{TP}{TP + FP} \quad (3.3)$$

Una alta precisión indica que el modelo no tiene muchos falsos positivos. Sin embargo, esta métrica puede ser engañosa en conjuntos de datos desequilibrados, ya que no tiene en cuenta los falsos negativos. Oscila entre 0 y 1, donde un valor de 1 indica que todas las identificaciones positivas fueron correctas, mientras que un valor de 0 indica que todas las identificaciones positivas fueron incorrectas.

Precisión Balanceada La precisión balanceada es la media aritmética de la sensibilidad y la especificidad. Proporciona una medida de rendimiento que es robusta incluso cuando las clases son muy desequilibradas. Se calcula como:

$$PrecisionBalanceada = \frac{Sensibilidad + Especificidad}{2} \quad (3.4)$$

La precisión balanceada es útil cuando se quiere tener en cuenta tanto la sensibilidad como la especificidad en la evaluación del rendimiento del modelo, y es especialmente relevante en situaciones de clases desequilibradas. Oscila entre 0 y 1, donde un valor de 1 indica que tanto la sensibilidad como la especificidad son perfectas, mientras que un valor de 0 indica que tanto la sensibilidad como la especificidad son nulas.

Coefficiente Kappa de Cohen El coeficiente Kappa de Cohen es una medida de acuerdo entre dos *raters* que clasifican los elementos en categorías mutuamente excluyentes. Se calcula como:

$$Kappa = \frac{p_o - p_e}{1 - p_e} \quad (3.5)$$

donde p_o es la proporción de veces que los *raters* están de acuerdo y p_e es la proporción de veces que se esperaría que estuvieran de acuerdo por casualidad. El coeficiente Kappa de Cohen es una buena medida para evaluar la consistencia en la clasificación en múltiples categorías. El valor de Kappa oscila entre -1 y 1, donde un valor de 1 indica un acuerdo perfecto entre los *raters*, un valor de 0 indica un acuerdo no mejor que el azar, y un valor de -1 indica un desacuerdo total entre los *raters*.

En general, estas métricas proporcionan una visión completa del rendimiento del modelo, teniendo en cuenta tanto su capacidad para identificar correctamente los positivos y los negativos, como la capacidad para evitar falsos positivos y falsos negativos. No obstante, para el contexto de este estudio en que las clases están ligeramente desbalanceadas, consideramos con especial atención la precisión balanceada y el coeficiente de Kappa.

Una vez descritas las métricas que se van a utilizar para evaluar la eficiencia de los modelos, procedemos a describir en detalle la aplicación de las diferentes técnicas de SL utilizados en este estudio. Cada subsección se centrará en la implementación de un enfoque específico y en la evaluación de sus resultados en términos de eficiencia de clasificación. Posteriormente se proporcionará una visión global de las posibles razones de estos resultados en el capítulo de Discusión (Sección 4).

Es importante destacar que todos los modelos presentados en este estudio fueron evaluados mediante búsqueda de hiperparámetros en rejilla para encontrar los óptimos en términos de clasificación de los

datos de entrenamiento. Sin embargo, una vez ajustado el modelo con hiperparámetros óptimos, solo se realizó una única predicción de la categoría de los datos de prueba, generando una sola matriz de confusión y un solo conjunto de métricas de rendimiento. En un estudio con una muestra de mayor tamaño, se habría recomendado utilizar la técnica de remuestreo conocida como *bootstrap*, ampliamente validada y utilizada, para garantizar la robustez de los modelos obtenidos. No obstante, debido al reducido tamaño de la muestra de prueba, no se utilizó *bootstrap* en este estudio. Chernick (2011) advierte que no es recomendable aplicar *bootstrap* en muestras de prueba reducidas, ya que puede generar nuevas muestras demasiado similares a la original, lo que resultaría en una sobreestimación del rendimiento del modelo y en la producción de intervalos de confianza sesgados. Esto se debe a que las muestras *bootstrap* generadas pueden no capturar de manera adecuada la variabilidad inherente presente en los datos originales.

3.1. Árboles de decisión

Se utilizó el lenguaje de programación **R** (R, 2023) y las bibliotecas **caret** y **rpart**, entre otras, para crear y analizar un árbol de decisión con el objetivo de clasificar a pacientes en dos grupos, AD y CN. Se dividió el conjunto de datos en dos subconjuntos: entrenamiento (90 %) y prueba (10 %), con el fin de evaluar la capacidad de generalización del modelo. Los datos de imagen se redujeron a un vector unidimensional para su análisis, mientras que los valores de *NA* fueron reemplazados por ceros (proceso estándar en neuroimagen ya que un *NA* es un dato enmascarado y se considera un valor de actividad nula, no inexistente), dejando la base de datos preparada para uso posterior con los siguientes modelos.

Para encontrar la mejor configuración de hiperparámetros del árbol de decisión, se empleó la técnica de búsqueda exhaustiva junto con la validación cruzada de 10 pliegues utilizando una serie de métricas de evaluación como la sensibilidad, especificidad, precisión, precisión balanceada, y el coeficiente Kappa de Cohen (para más detalles, ver Sección 3). Los hiperparámetros ajustados fueron *cp* (parámetro de complejidad o *complexity parameter*) y *maxdepth* (profundidad máxima del árbol o *maximum depth*), cuyos rangos para la búsqueda en cuadrícula se definieron como 10 valores entre 0.001 a 0.1 y los números enteros de 1 a 10, respectivamente. Los valores óptimos de hiperparámetros se encontraron en *cp* = 0,089 y *Max.Depth* = 1.

Con estos hiperparámetros se entrenó un modelo utilizando el subconjunto de entrenamiento y se evaluó su eficiencia sobre el subconjunto de prueba. La Figura 3.1 muestra el árbol de decisión resultante, donde los nodos representan las condiciones basadas en las características de los pacientes, y las hojas indican la clasificación final en uno de los dos grupos. Las métricas de eficiencia de este árbol de decisión han sido calculada a partir de la matriz de confusión mostrada en el Cuadro 3.2.

Observado/Predicción	Positivo	Negativo
Positivo	1	1
Negativo	4	6

Cuadro 3.2: Matriz de confusión para el árbol de decisión.

El Cuadro 3.3 muestra las métricas de precisión en la clasificación para este modelo de árbol de decisión. La eficiencia de este árbol ha sido relativamente baja, lo que era esperable ya que los árboles de decisión no son adecuados para capturar la correlación espacial inherente a las imágenes ni para modelar relaciones complejas y no lineales entre grandes cantidades de características, como es el caso de las imágenes de neuroimagen PET. El código para la implementación de este modelo está disponible en el Apéndice A.

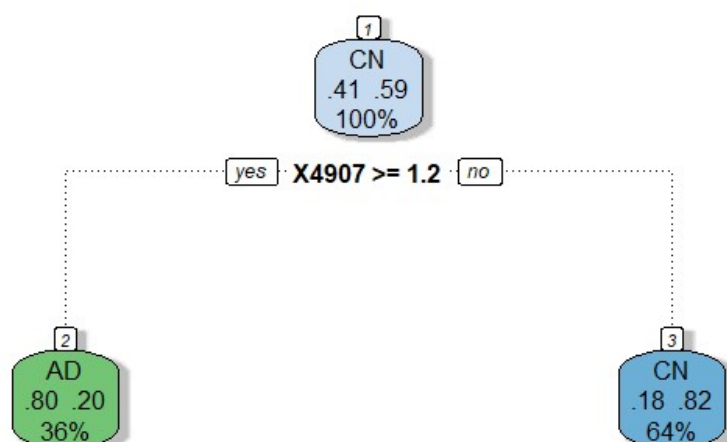



Figura 3.1: Árbol de decisión generado a partir de nuestro conjunto de datos.

Sensibilidad	Especificidad	Precisión	Precisión Balanceada	Kappa Coef.
0.2	0.8571	0.5833	0.5285	0.0625

Cuadro 3.3: Medidas de eficiencia para el árbol de decisión.

3.2. Bosques Aleatorios

Para el modelo de bosques aleatorios se utilizó  y la biblioteca `randomForest`. Para la selección del modelo, se realizó una búsqueda en cuadrícula de hiperparámetros. Estos incluyen el número de árboles en el bosque (`nmtree`) con valores de 100, 200, 300, 400, y 500 y el número de variables disponibles para la división en cada nodo (`mtry`), con los valores 86, 1003, 1920, 2837, y 3754. Se iteró a través de las combinaciones para encontrar el modelo con la mayor capacidad predictiva, que resultó ser el que contaba con los hiperparámetros $N.Tree = 100$ y $M.Try = 1920$. Se dividió el conjunto de datos en dos subconjuntos: entrenamiento (90 %) y prueba (10 %) y se evaluó sobre este segundo subconjunto. El seleccionado como mejor modelo presentó la matriz de confusión que se muestra en el Cuadro 3.4. A partir de esta matriz de confusión se calcularon las métricas de precisión que se muestran en el Cuadro 3.5.

Observado/Predicción	Positivo	Negativo
Positivo	3	2
Negativo	2	5

Cuadro 3.4: Matriz de confusión para el bosque aleatorio.

Adicionalmente, la Figura 3.2 presenta la gráfica de error OOB del modelo. En el eje X se representan los árboles generados durante el entrenamiento y en el eje Y se mide la tasa de error OOB. Las tres líneas corresponden a la tasa de error general (negro), y las tasas de error de la clase CN

Sensibilidad	Especificidad	Precisión	Precisión Balanceada	Kappa Coef.
0.6	0.7142	0.6666	0.6571	0.3142

Cuadro 3.5: Medidas de eficiencia para el bosque aleatorio.

(rojo) y clase AD (verde), respectivamente. Esta visualización proporciona una visión intuitiva de la evolución del error a medida que se incrementa la cantidad de árboles. Se puede observar, por ejemplo, que la tasa de error para el grupo control es mayor que para el grupo AD, lo cual es positivo en un contexto clínico (ver Sección 4). Estos resultados muestran que, pese a la esperable mejoría respecto a los árboles de decisión, el modelo de bosques aleatorios no proporciona una capacidad de clasificación sobresaliente.

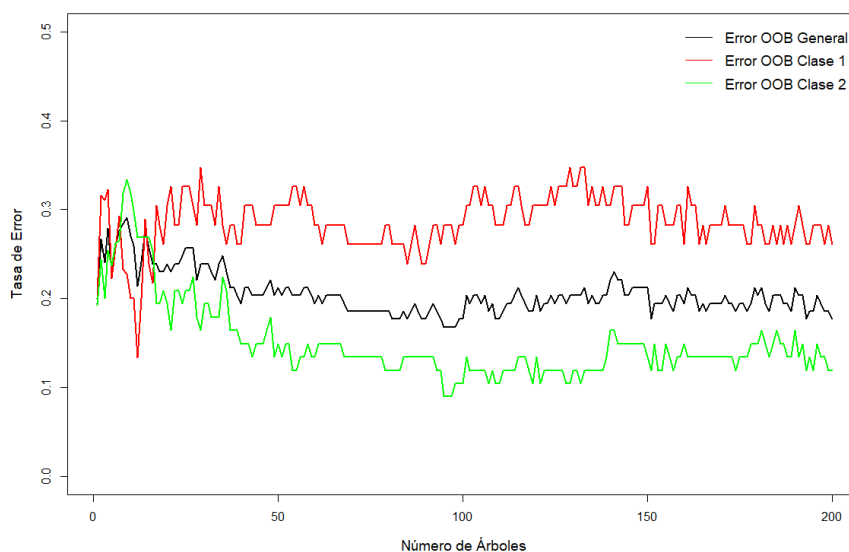


Figura 3.2: Evolución de la tasa de error OOB en el modelo de bosque aleatorio.

Por último, la Figura 3.3 muestra la importancia relativa de variables para el bosque aleatorio resultante. Cada punto en el gráfico representa una variable y su posición en el eje X indica la disminución media del *Índice Gini*, una métrica que indica el promedio de la disminución en impureza que resulta de las divisiones sobre una variable específica, promediada a lo largo de todos los árboles en el bosque. Esta visualización permite identificar rápidamente las variables más influyentes en el modelo.

Por ejemplo, es relevante observar que las variables predictoras que aparecen en esta gráfica como las más relevantes para el diagnóstico son $X4904$, $X5002$, o $X4812$, entre otras. Teniendo en cuenta que se trata de coordenadas convertidas a un vector, se puede deducir que estos píxeles se encuentran en regiones cercanas. Esto nos indica que existen ciertas áreas en nuestros datos que tienden a ser de especial relevancia, lo cual va en línea con lo esperado.

En estudios posteriores sería interesante comprobar, como se comenta en más detalle en la Sección 5.1, qué regiones presentan esta tendencia y comparar con otros estudios neuroanatómicos para verificar si esta relevancia está apoyada por otras investigaciones en el campo del diagnóstico de AD. Esta sugerencia cobra más peso todavía al comprobar que en el caso del árbol de decisión es también un píxel cercano a estos ($X4907$) el seleccionado para realizar la clasificación (ver Figura 3.1).

Como se ilustra en las gráficas anteriores y en el Cuadro 3.5, la eficiencia de este bosque aleatorio

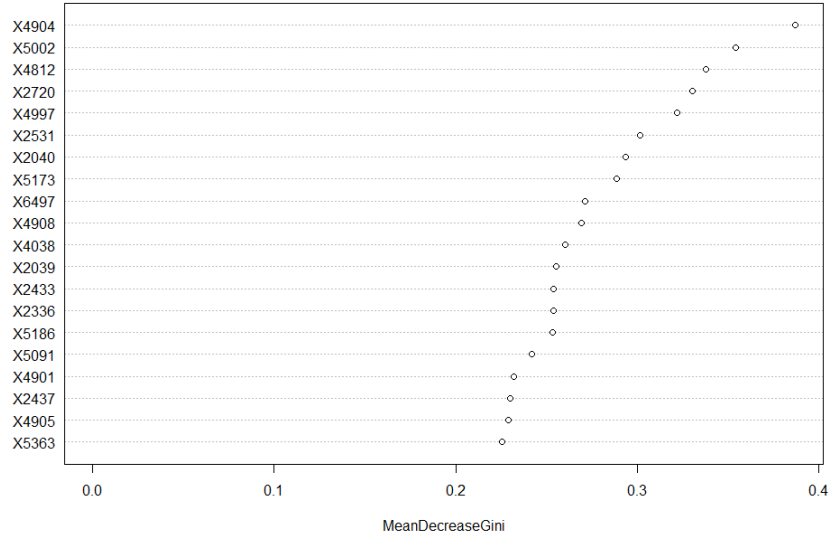


Figura 3.3: Gráfico de importancia de las variables de un modelo de bosque aleatorio.

ha sido ligeramente superior a la del árbol de decisión, lo cual era de esperar dado que -como técnica de *bagging* especialmente diseñada para árboles de decisión- los bosques aleatorios son generalmente más eficaces y manejan una mayor cantidad de características, lo que les facilita el modelar relaciones complejas. Sin embargo, al igual que los árboles de decisión, los bosques aleatorios no son capaces de capturar la correlación espacial ni de modelar relaciones no lineales entre las grandes cantidades de características presentes en los datos de neuroimagen PET. Estas cuestiones se tratarán con más profundidad en la Sección 4.

El código de [R](#) utilizado para el análisis de bosques aleatorios está disponible en el Apéndice A.

3.3. Máquinas de vectores de soporte

A continuación analizamos la metodología usada con los SVM como método de clasificación. A diferencia de los árboles de decisión y los bosques aleatorios, las SVM son especialmente eficaces en espacios de alta dimensión al ser capaces de trazar hiperplanos complejos.

Se utilizó [R](#) con la biblioteca `e1071` para ajustar el modelo SVM. Se realizó una búsqueda en cuadrícula de los mejores hiperparámetros para un *kernel* radial, con diferentes valores de *coste* (2^{-2} , 2^{-1} , 2^0 , 2^1 , 2^2) y de *gamma* (0.1, 0.5, 1, 2 y 5). El parámetro de *coste* controla el equilibrio entre la maximización del margen y la minimización del error de clasificación en el conjunto de entrenamiento, mientras que el parámetro *gamma* es específico del *kernel* de base radial y controla la *forma* de la función de base. Esto nos permitió explorar si un *kernel* no lineal podría mejorar la eficiencia del modelo. Los hiperparámetros óptimos resultaron ser *coste* = 2^{-1} y *gamma* = 0.5. La matriz de confusión resultante del uso de los mejores hiperparámetros se presenta en el Cuadro 3.6.

Como se observa en el Cuadro 3.7, la precisión de este modelo no fue mejor que un clasificador aleatorio. Este resultado es sorprendentemente pobre dado que las SVM suelen ser eficaces en espacios de alta dimensión. Sin embargo, al igual que los árboles de decisión, las SVM tratan cada característica de forma independiente, lo que puede no ser adecuado para datos donde la correlación espacial es relevante. Estos aspectos serán discutidos en más detalle en la Sección 4.

El código de [R](#) utilizado para el análisis de SVM está disponible en el Apéndice A.


Observado/Predicción	Positivo	Negativo
Positivo	0	0
Negativo	5	7

Cuadro 3.6: Matriz de confusión para el SVM.

Sensibilidad	Especificidad	Precisión	Precisión Balanceada	Kappa Coef.
0	1	0.5833	0.5	0

Cuadro 3.7: Medidas de eficiencia para máquinas de vectores de soporte.

3.4. Aprendizaje profundo

La implementación del aprendizaje profundo en este análisis implicó el uso de la biblioteca **keras** en *Python* y otras bibliotecas auxiliares en . Se dividió el conjunto de datos en dos subconjuntos: entrenamiento (90 %) y prueba (10 %) y se codificaron las etiquetas de clase como vectores binarios.

Definimos un modelo inicial con dos capas ocultas, con función de activación *ReLU* (Rectified Linear Unit, una función que devuelve 0 para los valores de entrada negativos y el mismo valor para los valores de entrada positivos), y un optimizador *RMSprop* (una variante del descenso de gradiente estocástico que utiliza la media cuadrática para normalizar el gradiente). El modelo se entrenó durante 20 épocas en lotes de 32 casos.

Dependiendo del grupo de investigación y los datos de estudio se suelen utilizar hiperparámetros distintos y, a día de hoy, no existe un *patrón oro* en la investigación de SL en neurociencias. Por lo tanto, a falta de referencia, tratamos de mejorar el modelo realizando una búsqueda en cuadrícula de hiperparámetros amplia. Los valores examinados se presentan resumidos en el Cuadro 3.8, **resaltando en rojo** aquellos que fueron seleccionados como hiperparámetros óptimos en la búsqueda de rejilla.

Hiperparámetro	Valores examinados
Número de capas ocultas	2, 3, 4, 5
Número de nodos en cada capa	32, 64, 128, 256
Función de activación	'relu' , 'tanh'
Tasa de aprendizaje	0.001 , 0.01, 0.1
Número de épocas de entrenamiento	10 , 20, 50, 100

Cuadro 3.8: Hiperparámetros considerados en la búsqueda de rejilla para aprendizaje profundo.

Para cada combinación de hiperparámetros definimos, compilamos, ajustamos, y evaluamos un modelo de aprendizaje profundo. La matriz de confusión correspondiente al modelo con los hiperparámetros óptimos se muestra en el Cuadro 3.9.

Las métricas de rendimiento correspondientes al mejor modelo se muestran en el Cuadro 3.10.

Observado/Predicción	Positivo	Negativo
Positivo	4	1
Negativo	2	5

Cuadro 3.9: Matriz de confusión para el aprendizaje profundo.

Sensibilidad	Especificidad	Precisión	Precisión Balanceada	Kappa Coef.
0.6667	0.8333	0.75	0.75	0.5

Cuadro 3.10: Medidas de eficiencia para aprendizaje profundo.

Basándonos en esta tabla, podemos decir que el aprendizaje profundo proporcionó los mejores resultados hasta este punto. Esta relativamente alta eficiencia podría deberse a la capacidad de las redes neuronales profundas para manejar relaciones altamente no lineales y multidimensionales, que son comunes en los datos de neuroimagen PET. Los detalles sobre el código implementado pueden encontrarse en el Apéndice A.

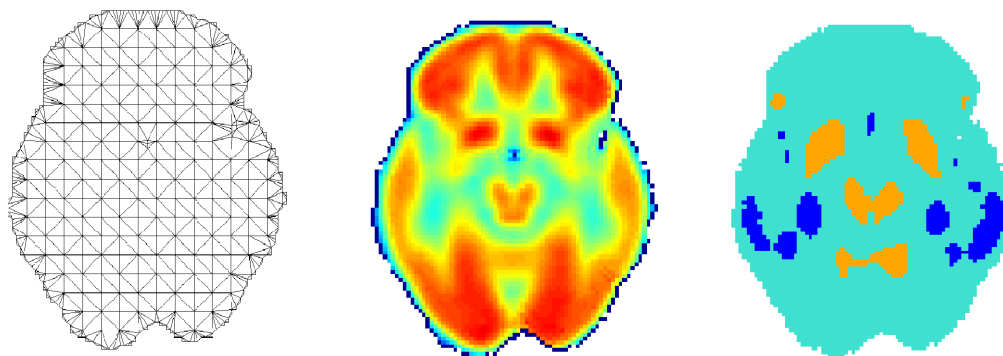
3.5. Aprendizaje profundo con datos funcionales

Finalmente, procedemos a implementar un algoritmo de aprendizaje profundo que se beneficie del proceso de *ingeniería de características* basada en resultados obtenidos previamente mediante técnicas de FDA. Este enfoque se centra en identificar áreas de nuestros datos que son particularmente relevantes para el diagnóstico, creando nuevas características centradas en estas regiones para proporcionárselas al algoritmo y comprobar si éstas ayudan en el proceso de clasificación.

En trabajos de investigación previos, la base de datos utilizada para este estudio fue analizada bajo los mismos parámetros de pre-procesamiento y dimensionalidad (Arias-Lopez et al., 2021). En dicho estudio, se obtuvieron las imágenes medias estimadas para los grupos CN y AD (ver Figura 3.4b) mediante las técnicas de FDA propuestas por Wang et al. (2020) que utilizan *splines* bivariados sobre una rejilla de Triangulaciones de Delaunay (ver Figura 3.4a) aplicadas sobre datos PET bidimensionales. En base a estas imágenes medias estimadas, se calcularon los SCCs para la diferencia entre grupos de imágenes (i.e., la diferencia entre imágenes del grupo AD y CN). Estos SCCs nos indican las regiones que presentan diferencias en actividad cerebral -tanto positivas como negativas- que caen fuera de los SCCs estimados (ver Figura 3.4c).

Las regiones identificadas por los SCCs son aquellas cuya actividad PET cae fuera de los intervalos de confianza obtenidos, lo que indica cambios significativos en los valores PET respecto a los esperados en esa región. Estos cambios pueden ser atribuidos a pérdidas o ganancias en la actividad cerebral subyacente, que a su vez pueden ser atribuibles al progreso de la AD en esas regiones, caracterizado por la progresiva muerte neuronal y subsecuente pérdida de actividad cerebral. Por lo tanto, es lógico pensar que cualquier información obtenida sobre estas regiones concretas de especial valor será de gran relevancia para el diagnóstico de AD con técnicas de SL como las aquí desarrolladas. En conclusión, la información obtenida de estas regiones de especial relevancia será utilizada para complementar -mediante técnicas de ingeniería de características- el algoritmo de aprendizaje profundo propuesto en este estudio.

En la Figura 3.4 se puede observar un ejemplo de los resultados obtenidos tras el proceso de triangulación de Delaunay (3.4a), estimación de la imagen media para un fragmento bidimensional de datos PET (3.4b), y cálculo de las regiones fuera de los intervalos de confianza (3.4c). Las nuevas



(a) Triangulaciones de Delaunay (b) Imagen media del grupo AD (c) Regiones fuera de los SCC

Figura 3.4: Resultados de Arias-Lopez et al. (2021) que identifican regiones de relevancia para diagnóstico de AD. Sub-figura (c) muestra en azul las regiones hipoactivas y en naranja las hiperactivas.

características generadas a partir de los datos de estas regiones de relevancia vienen definidas en el Cuadro 3.11, se crean individualmente para cada paciente y, dado que ciertos píxeles son relevantes debido a mostrar hipoactividad, mientras que otros lo son por mostrar hiperactividad, estas se crearon por triplicado: (1) para todos los píxeles relevantes, (2) para aquellos que son relevantes por mostrar hipoactividad, y (3) para aquellos relevantes por mostrar hiperactividad.

Estas características se le proporcionan por separado al algoritmo de aprendizaje profundo para que les otorgue los pesos convenientes y tenga la libertad de, en caso de ser beneficioso, darle más o menos peso a ciertas características según provengan del grupo hipoactivo o del hiperactivo.

Característica	Descripción
Media	Media de los valores de los píxeles de mayor relevancia.
Mediana	Mediana de los valores de los píxeles de mayor relevancia.
Máximo	Valor máximo de los píxeles de mayor relevancia.
Mínimo	Valor mínimo de los píxeles de mayor relevancia.
Desv. Estándar	Desviación estándar de los valores de los píxeles de mayor relevancia.
Varianza	Varianza de los valores de los píxeles de mayor relevancia.
Asimetría	Asimetría de los valores de los píxeles de mayor relevancia.
Kurtosis	Kurtosis de los valores de los píxeles de mayor relevancia.
Dif. Vecindario	Suma de las diferencias absolutas entre un píxel de relevancia y sus vecinos.

Cuadro 3.11: Nuevas variables generadas mediante ingeniería de características.

Una vez realizada la ingeniería de características, se dividió el conjunto de datos en entrenamiento (90%) y prueba (10%), se codificaron las etiquetas, y se realizó un bucle sobre la misma combinato-

ria de hiperparámetros que para el aprendizaje profundo, obteniéndose los mismos hiperparámetros óptimos, **resaltados en rojo** en el Cuadro 3.8. La matriz de confusión correspondiente al modelo con los hiperparámetros óptimos se muestra en el Cuadro 3.12


Observado/Predicción	Positivo	Negativo
Positivo	4	1
Negativo	1	6

Cuadro 3.12: Matriz de confusión para el aprendizaje profundo con ingeniería de características.

Las métricas de rendimiento correspondientes al mejor modelo se muestran en el Cuadro 3.13.

Sensibilidad	Especificidad	Precisión	Precisión Balanceada	Kappa Coef.
0.8	0.86	0.8	0.83	0.6571

Cuadro 3.13: Medidas de eficiencia para aprendizaje profundo con ingeniería de características.

El bucle sobre todas las combinaciones de hiperparámetros y el entrenamiento del modelo permitió identificar la combinación que proporcionaba las mejores métricas de eficiencia. Estas se muestran en el Cuadro 3.13. Para más detalles sobre la implementación en , consúltese el Apéndice A.

Resultados Finales

Modelo	Sensibilidad	Especificidad	Precisión	P. Balanceada	Kappa Coef.
Árboles	0.2	0.8571	0.5833	0.5285	0.0625
Bosques	0.6	0.7142	0.6666	0.6571	0.3142
SVM	0	1	0.5833	0.5	0
AP	0.6667	0.8333	0.75	0.75	0.5
AP & FDA	0.8	0.86	0.8	0.83	0.6571

Cuadro 3.14: Métricas de eficiencia para las diferentes técnicas evaluadas.

Se presentan los datos de forma resumida en el Cuadro 3.14 y también visualmente mediante una gráfica desglosada tanto por modelo como por métrica (ver Figura 3.5). Estos resultados proporcionan una visión clara y completa del rendimiento de las diferentes metodologías aplicadas, para poder así debatir sobre ellas en la Discusión (Sección 4).

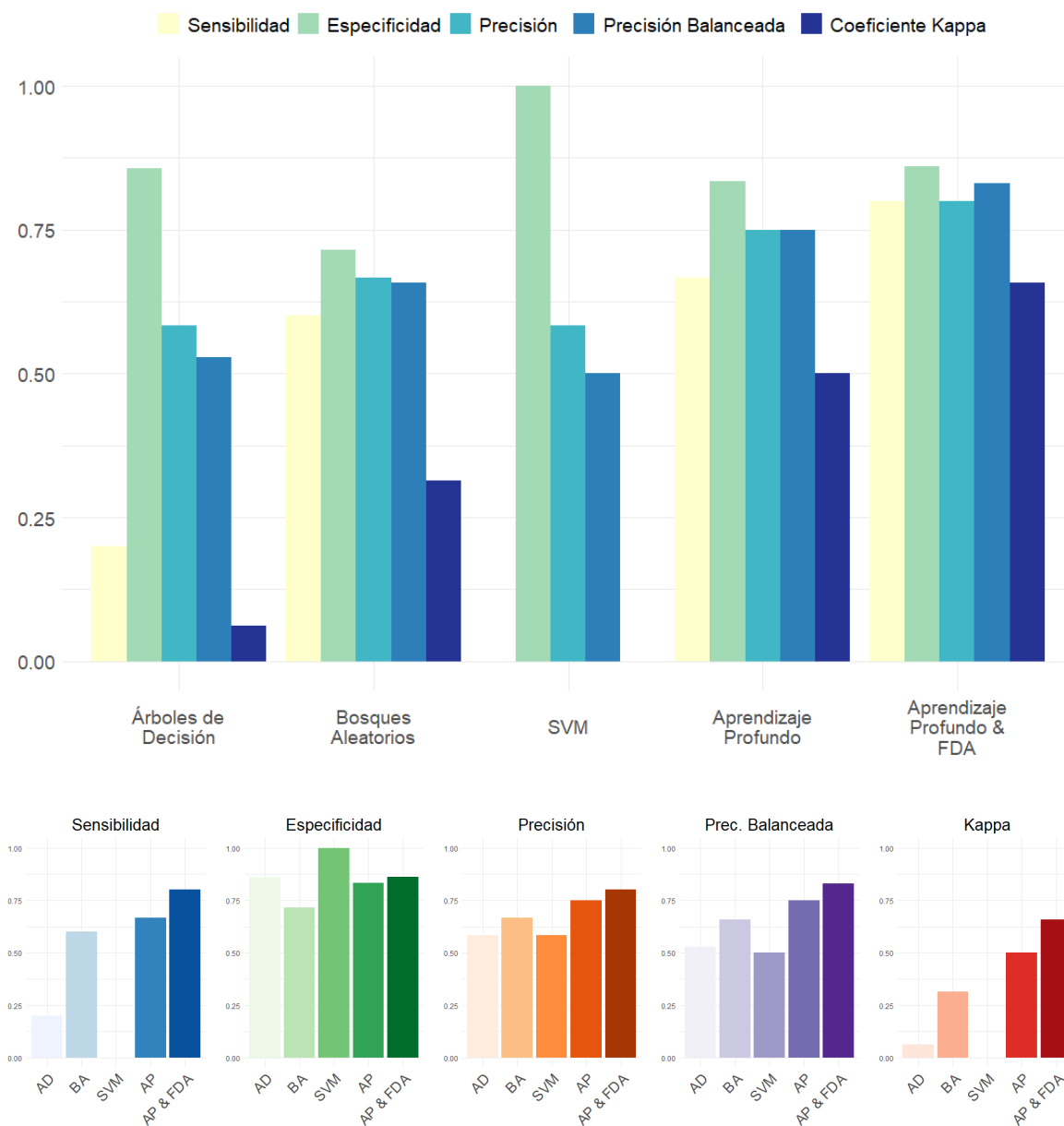


Figura 3.5: Visualización de resultados agrupados por metodología y por métrica (AD: Árbol de Decisión; BA: Bosque Aleatorio; AP: Aprendizaje Profundo).

Capítulo 4

Discusión

4.1. Interpretación de resultados

Los resultados obtenidos en este estudio reflejan, para empezar, la complejidad inherente de los datos de neuroimagen PET y la dificultad de aplicar ciertas técnicas estadísticas de SL a este tipo de datos. La naturaleza espacial de los datos y la alta dependencia entre los píxeles hacen que algunas de estas técnicas, como los árboles de decisión, sean menos eficaces, mientras que otras, como el aprendizaje profundo, pueden proporcionar mejores resultados.

Árboles de Decisión: el uso de árboles de decisión, como se esperaba, no proporcionó resultados satisfactorios. Esta técnica de SL, aunque útil en muchos contextos, no es adecuada para los datos de neuroimagen PET ya que la decisión de diagnóstico se está tomando en base a un número reducido de píxeles individuales, lo cual no tiene sentido desde el punto de vista neurobiológico ya que, hay que recordar, la actividad cerebral es un fenómeno global y altamente interconectado. Por lo tanto, la decisión basada en la actividad de una minúscula área del cerebro -representada por un píxel- puede no ser representativa de la actividad global ni de regiones más relevantes para el diagnóstico de AD dentro del cerebro y puede dar lugar a diagnósticos erróneos. Esto se ve claramente reflejado en los resultados obtenidos (Sección 3.1), que son poco mejores que los propios de un clasificador aleatorio.

Bosques Aleatorios: los bosques aleatorios, a pesar de no haber demostrado una eficacia sobresaliente en este estudio, han superado a los árboles de decisión en términos de rendimiento. Esta mejora puede atribuirse a la naturaleza de los bosques aleatorios como método de *bagging*, que combina múltiples árboles de decisión para generar un modelo más robusto. Al considerar múltiples píxeles de diferentes regiones, los bosques aleatorios pueden llegar a ser capaces proporcionar una visión más global de la actividad cerebral, lo que puede ser más representativo de la realidad neurobiológica que la decisión basada en un solo píxel o un pequeño número de píxeles. Sin embargo, la mejora en el rendimiento no fue tan significativa como cabría esperar. Una posible explicación sería que, aunque los bosques aleatorios son capaces de manejar la alta dimensionalidad de los datos de neuroimagen PET y de modelar relaciones más complejas que los árboles de decisión, todavía pueden tener dificultades para capturar la correlación espacial de nuestros datos. Los bosques aleatorios, al igual que los árboles de decisión, tratan cada píxel de forma independiente, lo que no es adecuado para la naturaleza de estos datos donde la actividad de un píxel está altamente correlacionada con la de sus vecinos, lo cual se puede apreciar en las métricas presentadas en la Sección 3.1. Cabe mencionar que, como se ve en la Figura 3.2, la capacidad de clasificación de este modelo es superior para el grupo AD que para el Control, lo cual se suele considerar beneficioso en contextos clínicos ya que habitualmente es preferible un falso positivo que un falso negativo.

Máquinas de Vectores de Soporte: las SVM, aunque son efectivas para gestionar espacios de alta dimensión, proporcionaron resultados sorprendentemente pobres en este estudio, no superando la eficacia de un clasificador aleatorio. Aunque las SVM son capaces de trazar fronteras de decisión complejas utilizando funciones Kernel, es posible que hayan tenido dificultades para manejar la alta correlación espacial presente en los datos de neuroimagen PET. En este sentido, las SVM, al igual que los árboles de decisión y los bosques aleatorios, tratan cada característica de forma independiente, lo que no es adecuado para datos donde la correlación espacial es relevante. No obstante, los resultados son claramente inferiores a estas otras técnicas que incurren en la misma limitación. Por otro lado, la similitud entre datos de los diferentes pacientes tras la normalización puede haber sido una limitación a la hora de trazar un hiperplano de separación entre clases. Esta limitación, junto con la incapacidad de las SVM para manejar adecuadamente la correlación espacial, han sido con toda probabilidad las causantes de los pobres resultados mostrados en la Sección 3.3. Otra razón factible es que, pese a haber trabajado sobre una rejilla de hiperparámetros recomendados, puede que hayamos logrado acercarnos a los hiperparámetros apropiados para este estudio.

Aprendizaje Profundo: el aprendizaje profundo demostró ser una técnica eficaz para este estudio, proporcionando resultados significativamente mejores que los métodos anteriores. Esta técnica, debido a su capacidad para modelar relaciones complejas, parece ser más adecuada para manejar la complejidad presente en los datos de neuroimagen PET. La eficacia del aprendizaje profundo puede atribuirse a su capacidad para manejar relaciones altamente no lineales, que son comunes en nuestro conjunto de datos. Las redes neuronales profundas son capaces de modelar interacciones complejas entre las características, lo que les permite capturar patrones en los datos que pueden ser indetectables para técnicas más simples. En el caso de la neuroimagen PET, esto significa que el aprendizaje profundo puede capturar patrones de actividad cerebral que son indicativos de AD, incluso cuando puedan ser altamente complejos y no lineales. Sin embargo, aunque los resultados del aprendizaje profundo fueron prometedores (ver Sección 3.4), no fueron excepcionales. Esto puede deberse a que, aunque contamos con una cantidad considerable de datos, el aprendizaje profundo se beneficia enormemente de grandes volúmenes de datos. En este caso, puede que no tuviéramos suficientes sujetos o suficiente diversidad en las naturaleza de las características, limitadas a la actividad cerebral en cada píxel y tres variables demográficas, para aprovechar plenamente el potencial del aprendizaje profundo.

Aprendizaje Profundo con Datos Funcionales: finalmente, el aprendizaje profundo con apoyo de ingeniería de características basada en resultados de FDA proporcionó los mejores resultados en este estudio. Esta técnica no sólo aprovecha la complejidad del aprendizaje profundo y su capacidad para tener en cuenta la relación espacial entre los píxeles, sino que también utiliza características adicionales basadas en los píxeles que se ha identificado como más relevantes para el diagnóstico de AD. Estas características adicionales, como la media, la mediana, la desviación estándar, la varianza, la kurtosis, o las diferencias absolutas entre píxeles de relevancia y su vecindario, proporcionan información adicional que puede ser útil para mejorar la eficacia del modelo.

La superioridad de este enfoque puede atribuirse a varias razones. En primer lugar, la ingeniería de características permite la extracción de información relevante a partir de los datos brutos, lo que puede mejorar la eficacia del modelo de aprendizaje profundo al proporcionarle un mayor número de variables predictoras. En segundo lugar, la ingeniería de características puede ayudar a reducir la dimensionalidad en el modelo, lo que puede aliviar el problema de la maldición de la dimensionalidad, mejorando la eficiencia del modelo (Yang and Xiang, 2022). En tercer lugar, la ingeniería de características ayuda a incorporar el conocimiento del dominio por parte del investigador, lo que puede mejorar la interpretabilidad del modelo y su capacidad para generalizar a nuevos datos (Liu, 2022).

En nuestro caso, las características adicionales proporcionan información valiosa que no puede ser capturada por los píxeles individuales. Por ejemplo, la media y la mediana capturan la tendencia central de la actividad cerebral en dicha región, mientras que la desviación estándar y la varianza capturan su variabilidad; la kurtosis, por otro lado, captura la forma de la distribución de la actividad

cerebral; y las diferencias absolutas con píxeles vecinos fuera de la región de relevancia aportan más información sobre la relación espacial entre los píxeles y sus diferencias. Todas estas características pueden ser útiles para un modelo de aprendizaje profundo a la hora de realizar diagnóstico diferencial, como evidencian los resultados mostrados en la Sección 3.5.

En conclusión, los resultados obtenidos en este estudio reflejan la alta complejidad de los datos de neuroimagen PET, así como la dificultad de aplicar técnicas de SL a este tipo de datos. Estos datos son por naturaleza altamente complejos y representan un sistema casi tan intrincado como el propio cerebro humano. Además, estos datos son limitados en su capacidad para capturar la actividad cerebral completa, ya que proporcionan una imagen estática de la actividad cerebral en un momento dado. Por lo tanto, es de esperar que las técnicas de SL que pueden manejar mejor la complejidad y tener en cuenta la relación espacial entre los píxeles proporcionen mejores resultados. Sin embargo, incluso con las técnicas más avanzadas, todavía se observa la existencia de un margen para mejorar la eficacia del diagnóstico.

4.2. Limitaciones

Este estudio, aunque ha proporcionado resultados valiosos, no está exento de limitaciones. En primer lugar, el tamaño de la muestra, aunque considerable para un estudio sobre AD en pacientes reales, podría no ser suficiente para aprovechar plenamente el potencial de las técnicas de SL, especialmente de aquellas que más se benefician del tamaño del conjunto de datos como el aprendizaje profundo. Del mismo modo, este tamaño muestral reducido nos ha forzado a trabajar con unos datos de prueba reducidos, lo que provoca que pequeños cambios en la matriz de confusión -por ejemplo, clasificar un paciente más como correcto o incorrecto respecto a otro método- tenga unos efectos desproporcionados y engañosos sobre las métricas finales de eficiencia. El reducido número de pacientes de prueba también fue un impedimento para llevar a cabo la técnica de remuestreo *bootstrap*, lo que habría sido útil para garantizar la robustez de los modelos y los resultados obtenidos (Chernick, 2011).

En segundo lugar, la diversidad de las características y variables potenciales para el modelo propuesto (aprendizaje profundo con FDA) es limitada. Aunque hemos incorporado una variedad de características basadas en los píxeles más relevantes para el diagnóstico, es posible que existan otras que podrían mejorar la eficacia del modelo de haber estado disponibles.

En tercer lugar, la interpretabilidad de los modelos, especialmente los de aprendizaje profundo, es limitada por definición. Aunque estos modelos seguirán siendo útiles para el diagnóstico, su naturaleza de *caja negra* implica que no nos proporcionarán información adicional sobre el origen de la enfermedad. En este sentido merecen una mención especial los bosques aleatorios que, pese a no tener una eficiencia muy alta, permiten analizar qué variables han sido más relevantes, lo cual abre la puerta a un estudio pormenorizado y a la búsqueda de posibles patrones recurrentes por parte del investigador.

En cuarto lugar, la generalización de los modelos a nuevos datos puede ser una limitación. Aunque hemos utilizado técnicas de validación cruzada para evaluar la eficacia de los modelos, es posible que estos no se generalicen bien a nuevos datos, especialmente si difieren significativamente de los datos de entrenamiento.

Por último, también es relevante mencionar que, aunque se trata de una limitación metodológica impuesta por la incapacidad del método de Wang et al. (2020) de trabajar con más de dos dimensiones, el estar reduciendo nuestros datos originales a un plano bidimensional está forzosamente reduciendo la cantidad de información disponible y haciendo imposible que las técnicas de SL accedan a la correlación entre valores de ciertos píxeles y los inmediatamente superiores e inferiores en el plano anatómico.

Capítulo 5

Conclusiones

5.1. Contribuciones al diagnóstico del Alzheimer y al Aprendizaje Estadístico

Las contribuciones de este estudio para el diagnóstico de AD y para el SL son múltiples. En primer lugar, se ha logrado cumplir con los objetivos y evaluar las hipótesis planteados, demostrando que el uso combinado de diferentes técnicas estadísticas avanzadas puede mejorar la precisión del diagnóstico de AD en estudios aplicados. Este logro es particularmente relevante en el campo de la neurociencia clínica, donde a menudo se utilizan técnicas más antiguas debido a su fiabilidad y a la dependencia de la senda. Nuestro estudio subraya la importancia de adoptar las técnicas más avanzadas en estadística para manejar datos complejos y difíciles de interpretar, una necesidad que ya se ha expuesto de forma recurrente en anteriores investigaciones ([Miotto et al., 2017](#)).

En segundo lugar, este estudio ha combinado con éxito el SL y los FDA, creando una sinergia aplicada que ha demostrado ser eficaz para el diagnóstico de AD. Esta combinación de técnicas ha permitido aprovechar la complejidad del aprendizaje profundo y su capacidad para tener en cuenta la relación espacial entre los datos, así como la incorporación de características adicionales basadas en métricas construidas a partir de los píxeles identificados como más relevantes mediante técnicas de FDA.

Por último, este estudio tiene implicaciones significativas para la sociedad contemporánea, especialmente la occidental, donde el envejecimiento de la población y la falta de una cura hacen del diagnóstico de AD una absoluta prioridad. Cuanto más temprano y eficiente sea el diagnóstico, antes se podrá aplicar el tratamiento disponible que retrase el avance de la enfermedad, reduciendo así su prevalencia al tratarse de una enfermedad propia de la vejez ([Brookmeyer et al., 2007](#)).

En resumen, este estudio ha mostrado que el uso de técnicas de SL avanzadas, en combinación con técnicas de FDA, puede mejorar significativamente la precisión del diagnóstico de AD. Estos hallazgos tienen implicaciones importantes para la práctica médica y la sociedad en general, y subrayan la necesidad de seguir investigando y adoptando las técnicas estadísticas más avanzadas al campo de la medicina aplicada.

5.2. Recomendaciones para futuras investigaciones

En vista de las limitaciones de este estudio vistas en la Sección 4.2, en futuras investigaciones -dirigidas a la consecución de una Tesis Doctoral- el autor considerará las siguientes mejoras en el estudio. En primer lugar, se debería aumentar el tamaño de la muestra y la diversidad de las características disponibles. El conjunto de datos utilizado en este estudio fue seleccionado por su realismo, no obstante, un conjunto de datos más grande y diverso podría permitir a los modelos de aprendizaje profundo aprovechar su plena capacidad para manejar grandes volúmenes de datos y

relaciones complejas y no lineales. Además, un mayor tamaño de muestra permitiría el uso de la técnica de remuestreo *bootstrap*, lo que sería beneficioso para garantizar la robustez de los resultados obtenidos en futuros estudios (Chernick, 2011). Adicionalmente, sería estratégico el uso de datos simulados para comprobar de forma fiable la eficiencia de la metodología propuesta.


En segundo lugar, se podrían explorar técnicas para mejorar la interpretabilidad de los modelos de aprendizaje profundo. Aunque estos modelos son efectivos para el diagnóstico, su naturaleza de *caja negra* puede limitar su utilidad para entender la enfermedad en sí. Técnicas muy novedosas como el Brain-Inspired Modular Training (Liu et al., 2023), que reduce el número de capas y ayuda a visualizar las más relevantes para la toma de decisiones, están todavía siendo desarrolladas y en un futuro próximo podrían ser útiles para entender qué características están siendo utilizadas por el modelo y por qué.

En tercer lugar, como hemos comentado anteriormente, los bosques aleatorios -si bien no han demostrado una alta eficiencia en este estudio- presentan ciertas características que los hacen especialmente interesantes para el trabajo clínico. En la Figura 3.3 observamos que podemos obtener el nombre de las variables que han demostrado tener más peso en las clasificaciones. A partir de estos datos se pueden rastrear las regiones que están teniendo más relevancia y un profesional clínico podría obtener conclusiones relevantes para el diagnóstico de AD. Es por ello que también se recomienda explorar esta vía, utilizando optimizadores para bosques aleatorios como **XGBoost**, o *Extreme Gradient Boosting*, un algoritmo especialmente diseñado para esta clase de metodologías estadísticas.

Finalmente, sería interesante aplicar modelos de aprendizaje profundo complementados con ingeniería de características en base a FDA en un entorno tridimensional. Esto permitiría el acceso del algoritmo a los datos de correlación espacial con los datos que rodean a cada pixel en todas las direcciones posibles, no solo a aquellos colindantes en el mismo plano anatómico. No obstante, esto implicaría la extensión de las metodologías desarrolladas en (Wang et al., 2020) para datos bidimensionales a datos tridimensionales, lo cual conlleva un complejo desarrollo matemático para garantizar la fiabilidad de la metodología.

Apéndice A

Código R

A continuación se presenta el código de  para implementar las diferentes fases de este estudio. Una versión más completa de los archivos procesados, matrices de resultados, gráficas, y otros contenidos puede encontrarse en nuestro [repositorio de acceso abierto](#) de Github. Garantizando así el acceso para futuras investigaciones y usuarios interesados en ampliar o replicar el estudio, y promoviendo la cultura de la investigación y los datos en abierto.

Árbol de Decisión

```
1 # Paso 1: Importar las bibliotecas necesarias
2
3 install.packages(c("caret", "rpart.plot", "oro.nifti"))
4
5 library("fs") # Para trabajar con rutas de archivos
6 library("readr") # Para cargar y trabajar con archivos CSV
7 library("matrixStats") # Para trabajar con matrices y datos numéricos
8 library("caret") # Para Aprendizaje Estadístico
9 library("rpart") # Para entrenar modelos
10 library("rpart.plot") # Para visualizar el árbol de decisión
11 library("RNifti"); library("oro.nifti") # tamaño de las imágenes
12 library("neurobase") # por si a caso
13 library("dplyr") # para las cañerías
14
15 # Paso 2: Cargar y aplanar los datos
16
17 # Establecer la ruta a la carpeta que contiene las imágenes NIfTI
18 data_folder <- "~/GitHub/TFM-Statistical-Learning/PETmasked"
19
20 # Cargar los archivos con extensión .hdr en la carpeta
21 nifti_data_z30 <- list()
22
23 # Loop de carga y quedarnos ya con Z=30 solamente
24 for (filename in list.files(data_folder)) {
25   if (endsWith(filename, ".hdr")) {
26     nifti_file <- file.path(data_folder, filename)
27     nifti_full <- oro.nifti::readNIfTI(nifti_file)
28     nifti_data_z30 <- c(nifti_data_z30, list(as.array(nifti_full)[, ,30]))
```

```

29   }
30 }
31
32 # Trasponer a 1x7505
33 nifti_matrix_flat <- matrix(nrow = length(nifti_data_z30),
34                             ncol = 79 * 95)
35
36 for (i in 1:length(nifti_data_z30)) {
37   nifti_matrix_flat[i,] <- as.vector(t(nifti_data_z30[[i]]))
38 }
39
40 dim(nifti_matrix_flat)
41
42 # Leer los datos demográficos desde un archivo CSV
43 demographic_data <- read.csv("Demographics.csv", sep=";", header=TRUE)
44
45 # Mantener solo las variables "Group", "Age" y "Sex"
46 demographic_data <- demographic_data[, c("Group", "Age", "Sex")]
47
48 # Combinar los datos demográficos con los datos de imagen aplanados
49 combined_data <- cbind(demographic_data, as.matrix(nifti_matrix_flat))
50
51 # Reemplazar todos los valores NaN con 0
52 combined_data[is.na(combined_data)] <- 0
53
54 # Paso 3: Dividir los datos en conjuntos de entrenamiento y prueba
55
56 # Dividir los datos en conjuntos de entrenamiento y prueba
57 set.seed(42)
58 train_index <- createDataPartition(combined_data[, "Group"], p = 0.9,
59                                    list = FALSE)
60 train_data <- combined_data[train_index, ]
61 test_data <- combined_data[-train_index, ]
62
63 # Paso 4: Entrenar el árbol de decisiones
64
65 # Convertir las columnas "Group", "Age" y "Sex"
66 train_data$Group <- as.factor(train_data$Group)
67 train_data$Age <- as.numeric(train_data$Age)
68 train_data$Sex <- as.factor(train_data$Sex)
69
70 # Asegurarse de que las columnas restantes sean numéricas
71 train_data[, -c(1,2,3)] <- apply(train_data[, -c(1,2,3)], 2, as.numeric)
72
73 # Crear el modelo del árbol de decisiones
74 library(rpart)
75
76 # Cálculo de Hiper-parámetros
77
78 # hiperparámetros con validación cruzada y búsqueda en cuadrícula
79 set.seed(42)
80

```



```

81 # Definir los rangos de hiperparámetros para la búsqueda en cuadrícula
82 tune_grid <- expand.grid(
83   .cp = seq(0.001, 0.1, length.out = 10),
84   .maxdepth = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
85 )
86
87 # Función para calcular la métrica de evaluación
88 compute_metric <- function(model, data) {
89   predictions <- predict(model, newdata = data, type = "class")
90   accuracy <- mean(predictions == data$Group)
91   return(accuracy)
92 }
93
94 # Bucle para buscar en la cuadrícula de hiperparámetros
95 best_accuracy <- 0
96 best_model <- NULL
97
98 for (cp in tune_grid$.cp) {
99   for (maxdepth in tune_grid$.maxdepth) {
100     tree_model <- rpart(Group ~ ., data = train_data, method = "class",
101                        control = rpart.control(minsplit = 10,
102                                                cp = cp,
103                                                maxcompete = 4,
104                                                maxsurrogate = 5,
105                                                usesurrogate = 2,
106                                                xval = 10,
107                                                surrogatestyle = 0,
108                                                maxdepth = maxdepth))
109     accuracy <- mean(tree_model$cptable[, "xerror"])
110     if (accuracy > best_accuracy) {
111       best_accuracy <- accuracy
112       best_model <- tree_model
113     }
114   }
115 }
116
117 # Obtener los hiperparámetros óptimos del objeto de resultados
118 best_cp <- bestmodel$control$cp
119 best_maxdepth <- bestmodel$control$maxdepth
120
121 # Entrenar el modelo utilizando los hiperparámetros óptimos
122 final_tree <- rpart(
123   Group ~ .,
124   data = train_data,
125   method = "class",
126   control = rpart.control(
127     minsplit = 10,
128     cp = best_cp,
129     maxcompete = 4,
130     maxsurrogate = 5,
131     usesurrogate = 2,
132     xval = 10,

```

```

133     surrogatestyle = 0,
134     maxdepth = best_maxdepth
135   )
136 )
137
138 # Visualizar el árbol de decisión
139 library(rpart.plot)
140 rpart.plot(final_tree, type = 3, box.palette = "auto",
141            shadow.col = "gray")
142
143 # Testear contra 'test_data'
144 test_data$Group <- as.factor(test_data$Group)
145 test_data$Age <- as.numeric(test_data$Age)
146 test_data$Sex <- as.factor(test_data$Sex)
147 test_data[, -c(1,2,3)] <- apply(test_data[, -c(1,2,3)], 2, as.numeric)
148 predicted_labels <- predict(final_tree, test_data, type = "class")
149
150 # Calcular la matriz de confusión
151 confusion_matrix <- confusionMatrix(predicted_labels, test_data$Group)
152
153 # Imprimir la precisión general
154 cat("Accuracy: ", confusion_matrix$overall["Accuracy"], "\n")
155
156 # Exportar la matriz de confusión
157 saveRDS(confusion_matrix, "Tree_confusion_matrix.RDS")

```

Bosque Aleatorio

```

1
2 install.packages("randomForest")
3
4 # Cargar paquetes necesarios
5 library(randomForest)
6 library(caret)
7
8 # Establecer el directorio de trabajo
9 setwd("~/GitHub/TFM-Statistical-Learning")
10
11 # Cargar los datos pre-procesados
12 combined_data <- read.csv("combined_data.csv")
13
14 # Dividir los datos en conjuntos de entrenamiento y prueba
15 set.seed(123)
16 train_index <- createDataPartition(y = combined_data$Group,
17                                   p = 0.8, list = FALSE)
18 train_data <- combined_data[train_index, ]
19 test_data <- combined_data[-train_index, ]
20
21 # Convertir la variable objetivo en un factor
22 train_data$Group <- as.factor(train_data$Group)

```

```

23 test_data$Group <- as.factor(test_data$Group)
24
25 # Definir la cuadrícula de hiperparámetros
26 hyper_grid <- expand.grid(
27   ntree = seq(100, 500, 100),
28   mtry = seq(floor(sqrt(ncol(train_data) - 1)),
29             floor((ncol(train_data) - 1) / 2),
30             length.out = 5)
31 )
32
33 # Crear una función para calcular la precisión
34 accuracy <- function(actual, predicted) {
35   sum(actual == predicted) / length(actual)
36 }
37
38 # Inicializar un data.frame para almacenar los resultados
39 results <- data.frame()
40 set.seed(123)
41
42 # Bucle a través de los hiperparámetros
43 for (i in 1:nrow(hyper_grid)) {
44   ntree <- hyper_grid[i, "ntree"]
45   mtry <- hyper_grid[i, "mtry"]
46
47   # Ajustar el modelo con hiperparámetros actuales
48   model <- randomForest(Group ~ ., data = train_data,
49                         ntree = ntree, mtry = mtry)
50
51   # Calcular la precisión en el conjunto de prueba
52   pred <- predict(model, newdata = test_data)
53   acc <- accuracy(test_data$Group, pred)
54
55   # Agregar los resultados al data.frame
56   results <- rbind(results, data.frame(ntree = ntree, mtry = mtry,
57   accuracy = acc))
58 }
59
60 # write.csv2(results, file = "decisiontree.results.csv")
61
62 # Encontrar los hiperparámetros que producen la mayor precisión
63 best_params <- results[which.max(results$accuracy), ]
64
65 # Ajustar el modelo final utilizando los mejores hiperparámetros
66 rf_model_tuned <- randomForest(Group ~ ., data = train_data,
67                               ntree = best_params$ntree,
68                               mtry = best_params$mtry)
69
70 # Predecir en el conjunto de prueba
71 predictions_rf_best <- predict(rf_model_tuned, newdata = test_data)
72
73 # Crear matriz de confusión y calcular la precisión
74 confusion_matrix_rf_best <- confusionMatrix(predictions_rf_best,

```

```

75                                     test_data$Group)
76 accuracy_rf_best <- confusion_matrix_rf_best$overall["Accuracy"]
77
78 # Accuracy del modelo
79 cat("Accuracy del modelo Random Forest: ", accuracy_rf_best, "\n")

```

Máquinas de Vectores de Soporte

```

1
2 # Instalar el paquete si no está instalado
3 if (!requireNamespace("e1071", quietly = TRUE)) {
4   install.packages("e1071")
5 }
6
7 # Cargar el paquete
8 library(e1071)
9 library(caret)
10
11 # Establecer el directorio de trabajo
12 setwd("~/GitHub/TFM-Statistical-Learning")
13
14 # Cargar los datos preprocesados
15 combined_data <- read.csv("combined_data.csv")
16
17 # Dividir los datos en conjuntos de entrenamiento y prueba
18 set.seed(123)
19 train_index <- createDataPartition(y = combined_data$Group,
20 p = 0.9, list = FALSE)
21 train_data <- combined_data[train_index, ]
22 test_data <- combined_data[-train_index, ]
23
24 # Convertir la variable objetivo en un factor
25 train_data$Group <- as.factor(train_data$Group)
26 test_data$Group <- as.factor(test_data$Group)
27
28 # Ajustar el modelo SVM utilizando un kernel lineal
29 svm_model <- svm(Group ~ ., data = train_data,
30 kernel = "linear", cost = 1)
31
32 # probar con otros tipos de kernel
33 # svm_model <- svm(Group ~ ., data = train_data,
34 kernel = "radial", gamma = 0.1, cost = 1)
35
36 # Predecir en el conjunto de prueba
37 svm_pred <- predict(svm_model, newdata = test_data)
38
39 # Calcular la matriz de confusión y la precisión
40 svm_confusion_matrix <- confusionMatrix(svm_pred, test_data$Group)
41 svm_accuracy <- svm_confusion_matrix$overall["Accuracy"]
42

```

```

43 print(svm_confusion_matrix)
44 print(svm_accuracy)
45
46 # Definir la cuadrícula de hiperparámetros
47 svm_hyper_grid <- expand.grid(
48   cost = 2^(-2:2),
49   gamma = c(0.1, 0.5, 1, 2, 5)
50 )
51
52 # Realizar la búsqueda en cuadrícula
53 set.seed(123)
54 svm_tuned <- tune(
55   svm, Group ~ ., data = train_data, kernel = "radial",
56   ranges = svm_hyper_grid,
57   tunecontrol = tune.control(sampling = "cross",
58   cross = 5)
59 )
60
61 # Ajustar el modelo SVM con los mejores hiperparámetros
62 svm_model_tuned <- svm(Group ~ ., data = train_data,
63 kernel = "radial", cost = svm_tuned$best.parameters$cost,
64 gamma = svm_tuned$best.parameters$gamma)
65
66 # Predecir en el conjunto de prueba
67 svm_tuned_pred <- predict(svm_model_tuned, newdata = test_data)
68
69 # Calcular la matriz de confusión y la precisión
70 svm_tuned_confusion_matrix <- confusionMatrix(svm_tuned_pred,
71 test_data$Group)
72 saveRDS(svm_tuned_confusion_matrix, file =
73 "svm_tuned_confusion_matrix.RDS")
74 svm_tuned_accuracy <- svm_tuned_confusion_matrix$overall["Accuracy"]
75
76 print(svm_tuned_confusion_matrix)
77 print(svm_tuned_accuracy)

```

Aprendizaje Profundo

```

1
2 # Dependencias Python
3 library(reticulate)
4 reticulate::use_condaenv("base")
5 reticulate::py_install(c("tensorflow", "numpy", "keras"),
6   envname = "base")
7
8 # Instalar el paquete keras si no está instalado
9 if (!requireNamespace("keras", quietly = TRUE)) {
10   install.packages("keras")
11 }
12

```

```

13 # Cargar el paquete
14 library(keras)
15 library(caret)
16 library(tensorflow)
17
18 # Establecer el directorio de trabajo
19 setwd("~/GitHub/TFM-Statistical-Learning")
20
21 # Cargar los datos preprocesados
22 combined_data <- read.csv("combined_data.csv")
23
24 # Las Deep Learning solo trabajan con números asique se
25 # codifican asi:
26
27 # Asignar 0 a "CN" y 1 a "AD"
28 combined_data$Group <- ifelse(combined_data$Group == "AD", 1, 0)
29
30 # Asignar 1 a "F" y 2 a "M"
31 combined_data$Sex <- ifelse(combined_data$Sex == "M", 2, 1)
32
33 # Dividir los datos en conjuntos de entrenamiento y prueba
34 set.seed(123)
35 train_index <- createDataPartition(y = combined_data$Group,
36 p = 0.9, list = FALSE)
37 train_data <- combined_data[train_index, ]
38 test_data <- combined_data[-train_index, ]
39
40 # Codificar las etiquetas de clase como vectores binarios
41 train_data$Group <- as.factor(train_data$Group)
42 train_data$Group <- as.integer(train_data$Group) - 1
43 train_labels <- to_categorical(train_data$Group)
44
45 test_data$Group <- as.factor(test_data$Group)
46 test_data$Group <- as.integer(test_data$Group) - 1
47 test_labels <- to_categorical(test_data$Group)
48
49 # Número de características
50 num_features <- ncol(train_data[, -which(names(train_data)
51 %in% c("Group"))])
52
53 # Definir la arquitectura del modelo
54 model <- keras_model_sequential() %>%
55   layer_dense(units = 256, activation = "relu",
56   input_shape = c(num_features)) %>%
57   layer_dense(units = 128, activation = "relu") %>%
58   layer_dense(units = length(unique(train_data$Group)),
59   activation = "softmax")
60
61 # Compilar el modelo
62 model %>% compile(
63   optimizer = "rmsprop",
64   loss = "categorical_crossentropy",

```

```

65     metrics = c("accuracy")
66 )
67
68 # Ajustar el modelo
69 history <- model %>% fit(
70     x = as.matrix(train_data[, -which(names(train_data)
71         %in% c("Group"))]),
72
73     y = train_labels,
74     epochs = 20,
75     batch_size = 32,
76     validation_split = 0.1
77 )
78
79 # Evaluar el modelo de nuevo
80 score <- model %>% evaluate(
81     x = as.matrix(test_data[, -which(names(test_data) == "Group")]),
82     y = test_labels
83 )
84
85 # Imprimir la pérdida y la precisión
86 cat('Test_loss:', score[[1]], '\n')
87 cat('Test_accuracy:', score[[2]], '\n')
88
89 ## Hyper-grid:
90
91 # Lista de hiperparámetros para probar
92 hyper_grid <- expand.grid(
93     units = c(64, 128, 256),
94     activation = c("relu", "tanh", "sigmoid"),
95     lr = c(0.001, 0.01, 0.1),
96     epochs = c(10, 20, 30)
97 )
98
99 # Inicializar un data frame para almacenar los resultados
100 results <- data.frame()
101
102 # Bucle sobre todas las combinaciones de hiperparámetros
103 for (i in 1:nrow(hyper_grid)) {
104
105     # Extraer los hiperparámetros para esta iteración
106     units <- hyper_grid[i, "units"]
107     activation <- hyper_grid[i, "activation"]
108     lr <- hyper_grid[i, "lr"]
109     epochs <- hyper_grid[i, "epochs"]
110
111     # Definir el modelo
112     model <- keras_model_sequential() %>%
113         layer_dense(units = units, activation = activation,
114             input_shape = c(ncol(train_data) - 1)) %>%
115         layer_dense(units = length(unique(train_data$Group)),
116             activation = "softmax")

```

```

117
118 # Compilar el modelo
119 model %>% compile(
120   optimizer = optimizer_rmsprop(learning_rate = lr),
121   loss = "categorical_crossentropy",
122   metrics = c("accuracy")
123 )
124
125 # Entrenar el modelo
126 history <- model %>% fit(
127   x = as.matrix(train_data[, -which(names(train_data) ==
128     "Group")]),
129   y = train_labels,
130   epochs = epochs,
131   batch_size = 128,
132   validation_split = 0.1
133 )
134
135 # Evaluar el modelo en el conjunto de prueba
136 score <- model %>% evaluate(
137   x = as.matrix(test_data[, -which(names(test_data) ==
138     "Group")]),
139   y = test_labels
140 )
141
142 # Añadir los resultados a la tabla de resultados
143 results <- rbind(results, cbind(hyper_grid[i, ], Loss =
144   score[[1]], Accuracy = score[[2]]))
145 }
146
147 # Ver los resultados
148 print(results)
149 saveRDS(results, "deeplearning_results.RDS")
150
151 library(tensorflow)
152
153 # Predecir las probabilidades
154 test_probabilities <- model %>% predict(as.matrix
155 (test_data[, -which(names(test_data) == "Group")]))
156
157 # Convertir las probabilidades en clases
158 test_predictions <- test_probabilities %>%
159   '>'(0.5) %>%
160   k_cast("int32") %>%
161   as.array() %>%
162   as.integer()
163
164 # Convertir las probabilidades en clases
165 test_predictions <- apply(test_probabilities, 1, which.max) - 1
166
167 # Asegúrate de que es un factor con los niveles correctos
168 test_predictions <- as.factor(test_predictions)

```



```

169
170 # Convertir test_labels_vector en factor
171 test_labels_vector <- as.factor(test_labels_vector)
172
173 # Y deberías poder calcular la matriz de confusión sin problemas
174 cm_ap <- confusionMatrix(test_predictions, test_labels_vector)
175 print(cm_ap)
176
177 # INGENIERÍA DE CARACTERÍSTICAS
178
179 # 1. Carga las coordenadas con pesos segun SCC_COMP_1
180
181 # Dimensiones originales de la imagen
182 dim1 <- 79
183 dim2 <- 95
184
185 # Inicializar un vector de ceros con longitud 7505 para representar
186 # todos los píxeles
187 pixel_weights <- rep(0, dim1 * dim2)
188
189 # Función para convertir las coordenadas de 2D a 1D
190 convert_2D_to_1D <- function(row, col, dim1) {
191   return((row - 1) * dim1 + col)
192 }
193
194 # Actualizar los valores en 'points.N' y 'points.P'
195 for(i in 1:nrow(list_points$points.N)) {
196   idx <- convert_2D_to_1D(list_points$points.N[i, "row"],
197     list_points$points.N[i, "col"], dim1)
198   pixel_weights[idx] <- 1
199 }
200
201 for(i in 1:nrow(list_points$points.P)) {
202   idx <- convert_2D_to_1D(list_points$points.P[i, "row"],
203     list_points$points.P[i, "col"], dim1)
204   pixel_weights[idx] <- -1
205 }
206
207 # Asignar nombres a pixel_weights
208 names(pixel_weights) <- paste0("V", 1:(dim1 * dim2))
209
210 # 2. Generar nuevas variables
211
212 library(e1071)
213
214 # Crear nuevas características basadas en puntos relevantes
215
216 combined_data$Avg_Important_Pixels <-
217 rowMeans(combined_data[, -which(names(combined_data) %in%
218   c("Group", "Age", "Sex"))] * pixel_weights)
219
220 combined_data$Median_Important_Pixels <-

```

```

221 apply(combined_data[, -which(names(combined_data) %in%
222 c("Group", "Age", "Sex"))] * pixel_weights, 1, median, na.rm = TRUE)
223
224 combined_data$Max_Important_Pixels <-
225 apply(combined_data[, -which(names(combined_data) %in% c("Group",
226 "Age", "Sex"))] * pixel_weights, 1, max, na.rm = TRUE)
227
228 combined_data$Min_Important_Pixels <-
229 apply(combined_data[, -which(names(combined_data) %in% c("Group",
230 "Age", "Sex"))] * pixel_weights, 1, min, na.rm = TRUE)
231
232 combined_data$StdDev_Important_Pixels <-
233 apply(combined_data[, -which(names(combined_data) %in% c("Group",
234 "Age", "Sex"))] * pixel_weights, 1, sd, na.rm = TRUE)
235
236 combined_data$Var_Important_Pixels <-
237 apply(combined_data[, -which(names(combined_data) %in% c("Group",
238 "Age", "Sex"))] * pixel_weights, 1, var, na.rm = TRUE)
239
240 # Crear nuevas características basadas en points.N
241
242 combined_data$Avg_N_Pixels <-
243 rowMeans(combined_data[, -which(names(combined_data) %in% c("Group",
244 "Age", "Sex"))] * (pixel_weights == 1))
245
246 combined_data$Median_N_Pixels <-
247 apply(combined_data[, -which(names(combined_data) %in% c("Group",
248 "Age", "Sex"))] * (pixel_weights == 1), 1, median, na.rm = TRUE)
249
250 combined_data$Max_N_Pixels <-
251 apply(combined_data[, -which(names(combined_data) %in% c("Group",
252 "Age", "Sex"))] * (pixel_weights == 1), 1, max, na.rm = TRUE)
253
254 combined_data$StdDev_N_Pixels <-
255 apply(combined_data[, -which(names(combined_data) %in% c("Group",
256 "Age", "Sex"))] * (pixel_weights == 1), 1, sd, na.rm = TRUE)
257
258 combined_data$Var_N_Pixels <-
259 apply(combined_data[, -which(names(combined_data) %in% c("Group",
260 "Age", "Sex"))] * (pixel_weights == 1), 1, var, na.rm = TRUE)
261
262 combined_data$Skewness_N_Pixels <-
263 apply(combined_data[, -which(names(combined_data) %in% c("Group",
264 "Age", "Sex"))] * (pixel_weights == 1), 1, skewness, na.rm = TRUE)
265
266 combined_data$Kurtosis_N_Pixels <-
267 apply(combined_data[, -which(names(combined_data) %in% c("Group",
268 "Age", "Sex"))] * (pixel_weights == 1), 1, kurtosis, na.rm = TRUE)
269
270 # Crear nuevas características basadas en points.P
271
272 combined_data$Avg_P_Pixels <-

```

```

273 rowMeans(combined_data[, -which(names(combined_data) %in% c("Group",
274   "Age", "Sex"))] * (pixel_weights == -1))
275 combined_data$Median_P_Pixels <-
276 apply(combined_data[, -which(names(combined_data) %in% c("Group",
277   "Age", "Sex"))] * (pixel_weights == -1), 1, median, na.rm = TRUE)
278 combined_data$Max_P_Pixels <-
279 apply(combined_data[, -which(names(combined_data) %in% c("Group",
280   "Age", "Sex"))] * (pixel_weights == -1), 1, max, na.rm = TRUE)
281 combined_data$StdDev_P_Pixels <-
282 apply(combined_data[, -which(names(combined_data) %in% c("Group",
283   "Age", "Sex"))] * (pixel_weights == -1), 1, sd, na.rm = TRUE)
284 combined_data$Var_P_Pixels <-
285 apply(combined_data[, -which(names(combined_data) %in% c("Group",
286   "Age", "Sex"))] * (pixel_weights == -1), 1, var, na.rm = TRUE)
287 combined_data$Skewness_P_Pixels <-
288 apply(combined_data[, -which(names(combined_data) %in% c("Group",
289   "Age", "Sex"))] * (pixel_weights == -1), 1, skewness, na.rm = TRUE)
290 combined_data$Kurtosis_P_Pixels <-
291 apply(combined_data[, -which(names(combined_data) %in% c("Group",
292   "Age", "Sex"))] * (pixel_weights == -1), 1, kurtosis, na.rm = TRUE)
293
294 # Función para calcular la diferencia con los vecinos
295
296 neighbor_diff_basic <- function(image, dim1, dim2) {
297   image_matrix <- matrix(image, nrow = dim1, ncol = dim2)
298   diff_sum <- 0
299   for (row in 1:dim1) {
300     for (col in 1:dim2) {
301       if (pixel_weights[(row - 1) * dim1 + col] == 1) {
302         neighbors <- image_matrix[max(1, row - 1):
303           min(dim1, row + 1),
304             max(1, col - 1):
305             min(dim2, col + 1)]
306         diff_sum <- diff_sum +
307           sum(abs(neighbors - image_matrix[row, col]))
308       }
309     }
310   }
311   return(diff_sum)
312 }
313
314 # Función para calcular la diferencia con los vecinos
315
316 neighbor_diff <- function(image, dim1, dim2, pixel_weights) {
317   image_matrix <- matrix(image, nrow = dim1, ncol = dim2)
318   diff_sum <- 0
319   for (row in 1:dim1) {
320     for (col in 1:dim2) {
321       if (pixel_weights[(row - 1) * dim1 + col] == 1) {
322         neighbors <- image_matrix[max(1, row - 1):
323           min(dim1, row + 1),
324             max(1, col - 1):

```

```

325         min(dim2, col + 1)]
326         diff_sum <- diff_sum + sum(abs(neighbors -
327         image_matrix[row, col]))
328     }
329 }
330 }
331 return(diff_sum)
332 }
333
334 # Inicializar un vector de ceros con longitud 7505 para representar
335 todos los píxeles
336 pixel_weights_N <- rep(0, dim1 * dim2)
337 pixel_weights_P <- rep(0, dim1 * dim2)
338
339 # Actualizar los valores en pixel_weights_N y pixel_weights_P
340 for(i in 1:nrow(list_points$points.N)) {
341     idx <- convert_2D_to_1D(list_points$points.N[i, "row"],
342     list_points$points.N[i, "col"], dim1)
343     pixel_weights_N[idx] <- 1
344 }
345
346 for(i in 1:nrow(list_points$points.P)) {
347     idx <- convert_2D_to_1D(list_points$points.P[i, "row"],
348     list_points$points.P[i, "col"], dim1)
349     pixel_weights_P[idx] <- -1
350 }
351
352 # Calcular la diferencia con los vecinos para cada imagen
353
354 combined_data$Neighbor_Diff <-
355 apply(combined_data[, -which(names(combined_data) %in%
356 c("Group", "Age", "Sex"))], 1, neighbor_diff_basic,
357 dim1 = dim1, dim2 = dim2)
358
359 combined_data$Neighbor_Diff_N <-
360 apply(combined_data[, -which(names(combined_data) %in%
361 c("Group", "Age", "Sex"))], 1, neighbor_diff,
362 dim1 = dim1, dim2 = dim2, pixel_weights = pixel_weights_N)
363
364 combined_data$Neighbor_Diff_P <-
365 apply(combined_data[, -which(names(combined_data) %in%
366 c("Group", "Age", "Sex"))], 1, neighbor_diff,
367 dim1 = dim1, dim2 = dim2, pixel_weights = pixel_weights_P)
368
369
370 # 3. Divide los datos en conjuntos de entrenamiento y prueba
371 set.seed(123)
372 train_index <- createDataPartition(y = combined_data$Group,
373 p = 0.9, list = FALSE)
374 train_data <- combined_data[train_index, ]
375 test_data <- combined_data[-train_index, ]
376

```

```

377 # 4. Codifica las etiquetas de clase como vectores binarios
378 train_data$Group <- as.factor(train_data$Group)
379 train_data$Group <- as.integer(train_data$Group) - 1
380 train_labels <- to_categorical(train_data$Group)
381
382 test_data$Group <- as.factor(test_data$Group)
383 test_data$Group <- as.integer(test_data$Group) - 1
384 test_labels <- to_categorical(test_data$Group)
385
386 # 5. Define el número de características
387 num_features <- ncol(train_data[, -which(names(train_data)
388 %in% c("Group"))])
389
390 # 6. Bucle sobre todas las combinaciones de hiperparámetros
391 hyper_grid <- expand.grid(
392   units = c(64, 128, 256),
393   activation = c("relu", "tanh", "sigmoid"),
394   lr = c(0.001, 0.01, 0.1),
395   epochs = c(10, 20, 30)
396 )
397
398 # 7. Inicializar un data frame para almacenar los resultados
399 ponderated_results <- data.frame()
400
401 # 8. Bucle sobre todas las combinaciones de hiperparámetros
402 for (i in 1:nrow(hyper_grid)) {
403
404   # Extraer los hiperparámetros para esta iteración
405   units <- hyper_grid[i, "units"]
406   activation <- hyper_grid[i, "activation"]
407   lr <- hyper_grid[i, "lr"]
408   epochs <- hyper_grid[i, "epochs"]
409
410   # Definir el modelo
411   model <- keras_model_sequential() %>%
412     layer_dense(units = units, activation = activation,
413       input_shape = c(num_features)) %>%
414     layer_dense(units = length(unique(train_data$Group)),
415       activation = "softmax")
416
417   # Compilar el modelo
418   model %>% compile(
419     optimizer = optimizer_rmsprop(learning_rate = lr),
420     loss = "categorical_crossentropy",
421     metrics = c("accuracy")
422   )
423
424   # Entrenar el modelo SIN ponderación de muestras
425   history <- model %>% fit(
426     x = as.matrix(train_data[, -which(names(train_data) ==
427       "Group"))],
428     y = train_labels,

```

```

429     epochs = epochs ,
430     batch_size = 128,
431     validation_split = 0.1
432 )
433
434 # Evaluar el modelo en el conjunto de prueba
435 score <- model %>% evaluate(
436   x = as.matrix(test_data[, -which(names(test_data) ==
437     "Group")]),
438   y = test_labels
439 )
440
441 # Añadir los resultados a la tabla de resultados
442 ponderated_results <- rbind(ponderated_results,
443   cbind(hyper_grid[i, ], Loss = score[[1]],
444     Accuracy = score[[2]]))
445 }
446
447
448 # Ver los resultados
449 print(ponderated_results) # 0.91
450 saveRDS(ponderated_results, "deeplearning_ponderado_results.RDS")
451
452 library(tensorflow)
453
454 # Predecir las probabilidades
455 test_probabilities <- model %>% predict(as.matrix
456 (test_data[, -which(names(test_data) == "Group")]))
457
458 # Convertir las probabilidades en clases
459 test_predictions <- test_probabilities %>%
460   '>'(0.5) %>%
461   k_cast("int32") %>%
462   as.array() %>%
463   as.integer()
464
465 # Convertir las probabilidades en clases
466 test_predictions <- apply(test_probabilities, 1, which.max) - 1
467
468 # Asegúrate de que es un factor con los niveles correctos
469 test_predictions <- as.factor(test_predictions)
470
471 # Convertir test_labels_vector en factor
472 test_labels_vector <- as.factor(test_labels_vector)
473
474 # Y deberías poder calcular la matriz de confusión sin problemas
475 cm_ap <- confusionMatrix(test_predictions, test_labels_vector)
476 print(cm_ap)

```

Bibliografía

- ADNI. Alzheimer’s disease neuroimaging initiative. *RRID:SCR_003007*.
- Ahmad, M. A., Teredesai, A., and Eckert, C. (2020). Fairness, accountability, transparency in ai at scale: Lessons from national programs. page 690.
- Arias-Lopez, J., Cadarso-Suarez, C., and Aguiar-Fernandez, P. (2021). Simultaneous confidence corridors for neuroimaging data analysis: applications to alzheimer’s disease diagnosis. *Arxiv*.
- Arias-Lopez, J., Cadarso-Suarez, C., and Aguiar-Fernandez, P. (2022). Functional data analysis for imaging mean function estimation: Computing times and parameter selection. *Computers*, 11:91.
- Brejijyeh, Z. and Karaman, R. (2020). Comprehensive review on alzheimer’s disease: Causes and treatment. *Molecules*, 25(24):5789.
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140.
- Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, CA.
- Brookmeyer, R., Johnson, E., Ziegler-Graham, K., and Arrighi, H. (2007). Forecasting the global burden of alzheimer’s disease. *Alzheimer’s & Dementia*, 3(3):186–191.
- Castellazzi, G., Bruno, S., Toosy, A., Casiraghi, L., Palesi, F., Savini, G., Fazekas, F., D’Angelo, E., and Wheeler-Kingshott, C. (2020). Promises and pitfalls of relating functional connectivity to clinical outcomes. *Frontiers in Neuroinformatics*, 14.
- Chernick, M. R. (2011). *Bootstrap methods: A guide for practitioners and researchers*. John Wiley & Sons.
- Chew, L. (1989). Constrained delaunay triangulations. *Algorithmica*, 4:97–108.
- Ferraty, F. and Vieu, P. (2006). *Nonparametric functional data analysis: theory and practice*. Springer Science & Business Media.
- Friston, K. (2007). *Statistical Parametric Mapping*. Elsevier.
- Gocheva-Ilieva, S., Iliev, I., and Ivanov, S. (2021). *Cognitive Computing in Data-Intensive Applications*. MDPI-Multidisciplinary Digital Publishing Institute.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer.
- Hippius, H. and Neundorfer, G. (2003). The discovery of alzheimer’s disease. *Dialogues Clin Neurosci*, 5:101–108.

- Hort, J., O'Brien, J., Gainotti, G., Pirttila, T., Popescu, B., Rektorova, I., Sorbi, S., and Scheltens, P. (2012). Efn guidelines for the diagnosis and management of alzheimer's disease. *European Journal of Neurology*, 17:1318–1322.
- Jalbert, J., Daiello, L., and Lapane, K. (2008). Dementia of the alzheimer type. *Epidemiologic Reviews*, 30(1):15–34.
- Kelleher, J. (2019). *Deep Learning*. The MIT Press.
- Lai, M. and Wang, L. (2013). Data-driven sparse pls. *Statistica Sinica*, pages 1399–1417.
- Lane, C. A., Hardy, J., and Schott, J. M. (2018). Alzheimer's disease. *European Journal of Neurology*, 25(1):59–70.
- Liu, S. (2022). *Design and Analysis of Experiments in Computational Studies of Physical and Biological Systems*. PhD thesis, Northeastern University, Boston, Massachusetts.
- Liu, Z., Gan, E., and Tegmark, M. (2023). Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *ArXiv*.
- McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5:115–133.
- Millican, P. (2021). *Human-Like Machine Intelligence*. Oxford: Oxford Academic.
- Miotto, R., Wang, F., Wang, S., Jiang, X., and Dudley, J. (2017). Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, 19(6):1236–1246.
- Prince, M., Ali, G., Guerchet, M., Prina, A., Albanese, E., and Wu, Y. (2016). Recent global trends in the prevalence and incidence of dementia, and survival with dementia. *Alzheimers Res Ther*, 8(1).
- R, C. T. (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Ramsay, J. and Silverman, B. (2005). *Functional Data Analysis*. Springer Series in Statistics. Springer.
- Rosenblatt, F. (1960). Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309.
- Schöll, M., Lockhart, S. N., Schonhaut, D. R., O'Neil, J. P., Janabi, M., Ossenkoppele, R., Baker, S. L., Vogel, J. W., Faria, J., Schwimmer, H. D., Rabinovici, G. D., and Jagust, W. J. (2016). PET Imaging of Tau Deposition in the Aging Human Brain. *Neuron*, 89(5):971–982.
- Wang, Y., Wang, G., Wang, L., and Ogden, R. (2020). Simultaneous confidence corridors for mean functions in functional data analysis of imaging data. *Biometrics*, 76:427–437.
- Wenk, G. (2006). Neuropathologic changes in alzheimer's disease: potential targets for treatment. *J Clin Psychiatry*, 67(Suppl 3):3–7; quiz 23.
- Worsley, K., Taylor, J., Tomaiuolo, F., and Lerch, J. (2004). Unified univariate and multivariate random field theory. *NeuroImage*, 23:S189–S195.
- Yang, Y. and Xiang, Y. (2022). Approximation of functionals by neural network without curse of dimensionality. *Journal of Machine Learning*, 1(4):342–372.
- Zerr, I. (2015). *Alzheimer's Disease - Challenges for the Future*. IntechOpen, Rijeka.
- Zhang, X., Tian, Y., Wang, Z., Ma, Y., Tan, L., and Yu, J. (2021). The epidemiology of alzheimer's disease modifiable risk factors and prevention. *J Prev Alzheimers Dis*, 8(3):313–321.

Índice de figuras

1.1. Ejemplo de la estructura de una red neuronal superficial con tres capas ocultas.	9
2.1. Histograma para la variable 'Edad' en función de 'Sexo' y 'Diagnóstico'.	13
2.2. Gráficos descriptivos para la variable 'Diagnóstico' y para el desglose por la variable 'Sexo'.	14
2.3. Imágenes descriptivas de los datos de neuroimagen PET analizados en este estudio: (a) visualización funcional normal de un archivo de neuroimagen PET, (b) ROI superpuestas sobre el plano axial en $Z = 30$, (c) ROI superpuestas sobre plano sagital central ($X = 37$), y (d) ROI superpuestas sobre plano coronal central ($Y = 47$).	14
2.4. Comparación de archivos de neuroimagen PET: original, preprocesada (realineamiento, normalización espacial, y normalización de intensidad), y enmascarada.	15
3.1. Árbol de decisión generado a partir de nuestro conjunto de datos.	20
3.2. Evolución de la tasa de error OOB en el modelo de bosque aleatorio.	21
3.3. Gráfico de importancia de las variables de un modelo de bosque aleatorio.	22
3.4. Resultados de Arias-Lopez et al. (2021) que identifican regiones de relevancia para diagnóstico de AD. Sub-figura (c) muestra en azul las regiones hipoactivas y en naranja las hiperactivas.	25
3.5. Visualización de resultados agrupados por metodología y por métrica (AD: Árbol de Decisión; BA: Bosque Aleatorio; AP: Aprendizaje Profundo).	27

Índice de cuadros

3.1. Matriz de confusión estándar aplicada en este estudio.	17
3.2. Matriz de confusión para el árbol de decisión.	19
3.3. Medidas de eficiencia para el árbol de decisión.	20
3.4. Matriz de confusión para el bosque aleatorio.	20
3.5. Medidas de eficiencia para el bosque aleatorio.	21
3.6. Matriz de confusión para el SVM.	23
3.7. Medidas de eficiencia para máquinas de vectores de soporte.	23
3.8. Hiperparámetros considerados en la búsqueda de rejilla para aprendizaje profundo.	23
3.9. Matriz de confusión para el aprendizaje profundo.	24
3.10. Medidas de eficiencia para aprendizaje profundo.	24
3.11. Nuevas variables generadas mediante ingeniería de características.	25
3.12. Matriz de confusión para el aprendizaje profundo con ingeniería de características.	26
3.13. Medidas de eficiencia para aprendizaje profundo con ingeniería de características.	26
3.14. Métricas de eficiencia para las diferentes técnicas evaluadas.	26

Acrónimos

AD	Alzheimer's Disease	Enfermedad de Alzheimer
SPM	Statistical Parametric Mapping	Mapeo Paramétrico Estadístico
PET	Positron Emission Tomography	Tomografía de Emisión por Positrones
SL	Statistical Learning	Aprendizaje Estadístico
FDA	Functional Data Analysis	Análisis de Datos Funcionales
SVM	Support Vector Machines	Máquinas de Vectores de Soporte
CART	Classification And Regression Trees	Árboles de Clasificación y Regresión
RSS	Residual Squared Sum	Suma de Residuos al Cuadrado
OOB	Out Of Bag	Fuera de la Bolsa
ANN	Artificial Neural Networks	Redes Neuronales Artificiales
SCC	Simultaneous Confidence Corridors	Intervalos de Confianza Simultáneos
ROI	Region Of Interest	Región de Interés