Universidad de Zaragoza

Trabajo de fin de grado

Estudio comparativo e interpretabilidad de modelos estáticos y temporales de redes neuronales en la predicción de la evolución de la enfermedad de Alzheimer dentro del TadPole challenge

Comparative study and interpretability of static and temporal models of neural networks in the prediction of Alzheimer's disease evolution within the TadPole challenge

Autor: José Manuel Sánchez Aquilué
Directora: Mónica Hernández Giménez
Codirectora: Elvira Mayordomo Cámara

junio, 2021

Departamento de Informática e Ingeniería de Sistemas

Resumen

En la actualidad la investigación tiene delante el considerable desafío de avanzar en la lucha contra la demencia, cuya manifestación más común se da por medio de la enfermedad de Alzheimer. Se trata de una afección neurodegenerativa que provoca pérdida de memoria, de concentración y alteraciones en el comportamiento, y para la cual todavía no se conoce un método de diagnóstico preciso que no involucre una biopsia post-mortem.

Con la misión de contribuir a la mejora de las aplicaciones biomédicas en el diagnóstico de la enfermedad de Alzheimer, EuroPOND Consortium y ADNI elaboraron el concurso TADPOLE Challenge. Los participantes debían resolver una serie de problemas de regresión y clasificación con los métodos estadísticos o de aprendizaje automático que estimasen oportunos. Entre esos retos, nosotros nos hemos dedicado a estudiar la predicción del estado clínico. Dadas una serie de muestras de biomarcadores de pacientes extraídas de la base de datos de ADNI, buscábamos averiguar su estado de salud en corto y medio plazo, entre uno y cinco años. Dentro de los predictores había de tipo demográfico (edad, raza, sexo...), biológico, puntuaciones de test cognitivos, el factor genético e imagenes cerebrales, y entre las posibles categorías de diagnóstico podíamos distinguir los estados cognitive normal (CN), mild cognitive impairment (MCI) y Alzheimer's disease (AD).

Cuando se dio por finalizado el concurso, se presentó el conjunto de test (D4), sobre el que se evaluaron los modelos presentados para construir el ranking. En muchos otros contextos las redes neuronales han superado a las técnicas clásicas de aprendizaje automático. Sin embargo, aquí ocurrió lo contrario, el mejor método basado en deep learning ocupó la novena posición y el peor, la cuadragésima octava de un total de 58 métodos incluidos en el challenge.

En este trabajo hemos reproducido los modelos de aprendizaje profundo que se presentaron a TADPOLE challenge. Esto ha supuesto, en primer lugar, un estudio del estado del arte de las redes neuronales pre-alimentadas y recurrentes y su aplicación en materia de predicción de diagnósticos. Tras la fase de documentación pudimos implementar sistemas de redes neuronales. En concreto redes profundas, poco profundas, recurrentes y LSTM, un tipo de recurrencia que presenta un buen rendimiento en secuencias largas. Existen dos maneras de plantear este problema, en función del tipo de arquitectura que se utilice. Ora se interpreta cada muestra como un dato aislado, ora fundamentar la predicción en el historial completo del paciente. Por lo general, los métodos recurrentes presentan mejores resultados, debido a la evidente dependencia que existe entre las muestras de un mismo paciente. A pesar de que las redes profundas no tuvieron mucho éxito en el concurso (mAUC de 0,737 y BCA de 0,605), los algoritmos reproducidos en este TFG han logrado una mejor puntuación, una mAUC de 0,857 y un BCA de 0,77 sobre D4. Por el contrario, los modelos recurrentes quedaron ligeramente por debajo que los presentados en TADPOLE. Nuestros modelos alcanzaron hasta una mAUC de 0,861 y un BCA de 0,781, frente a la mejor RNN de TADPOLE, que ostenta una mAUC de 0,897 y un BCA de 0,803.

Una vez garantizado que los modelos indicados son reproducibles, y por consiguiente, se podrían hacer accesibles para el uso clínico, quedaría asegurar la interpretabilidad, condición sine qua non para la profesionalización de estas técnicas en el terreno de la biomedicina. El análisis se ha realizado mediante SHAP y llegando a la conclusión de que los atributos a los que más peso les otorgan las redes son aquellos en los que se apoyan los expertos clínicos a la hora de establecer el diagnóstico. Además se ha completado el estudio con una mirada a la configuración interna de las redes, mediante la herramienta TensorBoard. En esta observación contemplamos como el peso que le asigna la red en la primera capa a cada feature es directamente proporcional a la importancia que le concede SHAP. Asimismo hay una gran cantidad de pesos adquieren el valor de cero, en otras palabras, las redes no están desarrollando todo su potencial, una elevada proporción de neuronas no contribuyen nada, o lo hacen insignificantemente, al resultado final.

Los códigos fuente de este proyecto están recogidos en este repositorio: `https://github.com/jmsaquilue/TFG`

Glosario

En esta sección quedan recogidos todos aquellos acrónimos y siglas que figuran en la memoria, junto a la expresión completa a la que abrevian.

AD: Alzheimer's Disease. Enfermedad de Alzheimer.

ADAS-Cog: Alzheimer's Disease Assessment Scale–Cognitive Subscale.

ADNI: Alzheimer's Disease Neuroimaging Initiative.

APOE: apolipoproteína E.

BCA: Balanced Classification Accuracy. Precisión de Clasificación Equilibrada.

CN: Cognitive Normal. Cognición Normal.

CDRSB: Clinical Dementia Rating Sum of Boxes. Clasificación clínica de demencia.

CSF: Cerebrospinal fluid. Líquido craneoencefálico.

DNN: Deep neural network. Red neuronal profunda.

FDG: fluoro-deoxyglucose. fluorodesoxiglucosa.

FNN: Feed-forward Neural Network. Red neuronal pre-alimentada.

ICV: intracerebroventricular.

LASSO: least absolute shrinkage and selection operator. Operador de selección y contracción mínima absoluta.

$\mathbf{mAUC_{ROC}}$: Multi-class Area Under the Receiver Operating Curve. Área Multiclase Bajo la Curva de Característica Operativa del Receptor.

MCI: Mild Cognitive Impairment. Deterioro Cognitivo Leve.

MMSE: Mini-Mental State Examination.

MRI: magnetic resonance imaging. Imagen por resonancia magnética.

PET: Positron Emission Tomography. Tomografía por emisión de positrones.

RBF: Radial Basis Function. Función de base radial.

RELU: Rectified Linear Unit. Unidad lineal rectificada.

RID: Roster ID. Identificador de lista.

RNN: Recurrent neural network. Red neuronal recurrente.

SHAP: SHapley Additive exPlanations.

SVM: Support Vector Machine. Máquinas de vectores de soporte.

TADPOLE: The Alzheimer's Disease Prediction Of Longitudinal Evolution.

XAI: Explainable Artificial Intelligence. Inteligencia artificial explicable.

Índice

Índice de figuras y tablas

1. Introducción

1.1. Motivación y contexto

Más de 46,8 millones de personas en todo el mundo padecen demencia [6], siendo la enfermedad de Alzheimer (Alzheimer's Disease, AD) la causa más común de esta, en torno al 60 y el 70 por ciento de los casos. Esta neuropatología se desencadena a causa del deterioro progresivo de neuronas en ciertas regiones del tejido cerebral. Generalmente, la enfermedad comienza afectando al hipocampo. Más adelante, conforme avanza la patología, ésta afecta a otras áreas. Si bien la enfermedad del Alzheimer se manifiesta en mayor medida durante la senectud, no se debe considerar un síntoma inherente al envejecimiento. Por ejemplo, se conoce una variante temprana de Alzheimer. La muerte de neuronas se produce por la acumulación de proteínas anómalas en el cerebro, más concretamente placas de proteína beta-amiloide y de ovillos neurofibrilares de proteína Tau. La enfermedad comienza ocasionando una pérdida de memoria que impide recordar sucesos recientes, así como una falta de concentración, orientación y percepción visual, que dificultan la ejecución adecuada de tareas del día a día. En consecuencia también se originan trastornos psicológicos como depresión, ansiedad e irritabilidad. Finalmente, el padecimiento imposibilita ciertas funciones biológicas básicas, como la ingesta de alimentos, y por ende el paciente expira.

La Organización Mundial de la Salud alerta de que el número de víctimas de la enfermedad de Alzheimer se triplicará en los próximos treinta años. Es por ello que se están aglutinando esfuerzos con el propósito de dar con un tratamiento efectivo, ya que hoy en día no existe cura. Por otra parte, es de sumo interés el diagnóstico precoz, cuando se produce un deterioro cognitivo leve (Mild Cognitive Impairment, MCI) porque existen tratamientos para mitigar los síntomas, que son más efectivos en etapas tempranas. Cabe mencionar que el deterioro cognitivo leve no siempre es la antesala del Alzheimer, sin embargo existe un riesgo considerable, puesto que se estima que un 10 % de los pacientes con MCI terminarán degenerando en AD en menos de un año.

En suma, una de las prioridades de la investigación en la lucha contra la enfermedad de Alzheimer es predecir el avance de esta. No se trata de un cometido fácil: el único diagnóstico riguroso, a fecha de hoy, es la biopsia post-mortem. Para determinar el diagnóstico del paciente en la práctica clínica se debe analizar e interpretar la información de muy diversos biomarcadores que se agrupan en cuatro tipos. Primeramente se tienen en cuenta los test cognitivos. Constan de una serie de tareas breves, que involucran capacidad de orientación espacio-temporal, memoria a corto plazo y concentración, entre otras funciones cognitivas. En este tipo de diagnósticos los más utilizados son, por un lado, Mini-Mental State Examination (MMSE), que en castellano se conoce como Mini-Examen Cognoscitivo (MEC) y, por otro lado, Alzheimer's Disease Assessment Scale–Cognitive Subscale (ADAS-Cog). En segundo lugar, contamos con el escáner cerebral que nos proporciona una imagen por resonancia magnética (Magnetic Resonance Imaging, MRI). En tercer lugar, tenemos marcadores biológicos, como los utilizados con el fin de medir los niveles de proteínas beta-amiloide y Tau. Entre ellos podemos destacar el líquido craneoencefálico, limitado por los potenciales efectos secundarios de la punción lumbar (Cerebrospinal fluid, CSF), la tomografía por emisión de positrones (Positron Emission Tomography, PET) o la evaluación del metabolismo cerebral a través de fluorodesoxiglucosa (fluoro-deoxyglucose, FDG). En último lugar, se valora el factor genético, en especial el alelo e4 del gen de la apolipoproteína E (APOE). Sabemos que entre el 40 y el 65 por ciento de los enfermos de AD tienen al menos una copia de este alelo [5]. Cabe señalar que el análisis genético no es una práctica habitual, por lo menos en España.

En virtud de la acuciante necesidad de investigar la capacidad de los métodos de diagnóstico asistidos por computador para la enfermedad de Alzheimer, en 2017 tuvo lugar el concurso The Alzheimer's Disease Prediction Of Longitudinal Evolution (TADPOLE) Challenge, de la mano del consorcio EuroPOND. La finalidad de este reto era elaborar modelos predictivos que resolvieran el diagnóstico de una persona a corto o medio plazo, en un periodo entre uno y cinco años. Si bien en el desafío original se debía predecir el estado clínico, la puntuación del test cognitivo ADAS-Cog y el volumen de ventrículos, en este trabajo nos centraremos en el problema de predecir la clasificación

por estado clínico. Gracias a la colaboración de la iniciativa de neuroimagen de la enfermedad de Alzheimer (Alzheimer's Disease Neuroimaging Initiative, ADNI) contamos con datos de más de 1100 pacientes reales.

En la última década, los algoritmos basados en deep-learning han desbancado a otros métodos de aprendizaje automático gracias a la complejidad del modelo y la disponibilidad de grandes cantidades de datos. Sería de esperar que algún método de la familia hubiese alcanzado el podio del Tadpole challenge. No obstante, cuando en 2019 EuroPOND dió a conocer los resultados del reto, el mejor método de redes neuronales quedó en la novena posición. Cabe cuestionarse el por qué de estas puntuaciones.

Por lo tanto, el primer objetivo de este proyecto consiste en implementar las arquitecturas de redes neuronales propuestas dentro del challenge, y conseguir alcanzar, o incluso superar, la puntuación de aquellas presentadas al concurso TADPOLE Challenge.

Ahora bien, cuando se trata de aplicaciones médicas, no basta con entrenar diversos modelos y quedarnos con aquel que reporte mejor rendimiento. Es muy conveniente ir un paso más allá y abrir la caja negra para comprender aquello que impele a la máquina a tomar ciertas determinaciones, y así poder detectar posibles sesgos. Este concepto se conoce como interpretabilidad y, naturalmente, no es algo exclusivo de este problema o de las redes neuronales, sino que es extrapolable a otros procedimientos que hayan sido resueltos mediante técnicas de aprendizaje automático. De hecho, sería ampliamente recomendable averiguar si un modelo se rige por principios racionales antes de su comercialización, especialmente si se emplea en un ámbito tan delicado como es el biomédico.

De ahí que el segundo objetivo de este proyecto sea estudiar la interpretabilidad de los sistemas implementados para descubrir entre un piélago de predictores aquellos que causan un mayor impacto en las predicciones. Este punto no fue desarrollado en TADPOLE, y lo consideramos fundamental, debido a que otorgaría al estudio una mayor credibilidad. Con esto queremos decir que si los biomarcadores más relevantes coinciden con los de la práctica clínica, se podría entender que los métodos son capaces de sustentarse en una base científica y su precisión no ha sido fruto del azar y por otro lado la posibles discrepancias apuntan a potenciales nuevos marcadores para la práctica clínica.

Por último, añadir que este trabajo es la continuación del de Francisco Ferraz García [9], que el año pasado se encargó de reproducir y realizar un análisis de interpretabilidad de los métodos ganadores. En esta ocasión la parte de interpretabilidad ha supuesto una mayor complejidad dado que los *explainers* no están adaptados a todo tipo de redes neuronales y nos vimos obligados a hacer una serie de cambios en las *pipelines* hasta que se consiguió enlazar ambas partes.

1.2. Estado del arte

Como se ha mencionado anteriormente, el concurso de TADPOLE dio comienzo a mediados de 2017. A partir de los datos de ADNI los participantes debían elaborar sus modelos estadísticos o de inteligencia artificial que resolvieran una serie de desafíos, entre ellos la predicción del diagnóstico. Entre los competidores se encontraban grupos de investigación universitarios, empresariales, independientes e, incluso, institutos, que podían aspirar a ser galardonados con premios sustanciosos de hasta cinco mil libras y todo el prestigio que supone el triunfo en una competición de este estilo.

El primer desafío, la predicción del estado clínico, consistía en clasificar a los pacientes en tres categorías: CN, MCI y AD; en función de su diagnóstico a futuro. Al tratarse de un problema de clasificación, los concursantes debían reportar el área multiclase bajo la curva de característica operativa del receptor (Multi-class Area Under the Receiver Operating Curve, $mAUC_{ROC}$ o mAUC) y la precisión de clasificación equilibrada (Balanced Classification Accuracy, BCA), mientras que la parte de estimar la puntuación del test ADAS-Cog y el cálculo de la proporción de ventrículos

cerebrales en el volumen intracraneal se resuelven a través de técnicas de regresión y por tanto se atendían a otros criterios de evaluación: el error absoluto medio (Mean Absolute Error, MAE), la puntuación de error ponderada (weighted error score, WES) y precisión de la probabilidad de cobertura (Coverage Probability Accuracy, CPA).

Con el desenlace del reto en 2019, se publicó un conjunto de test (evaluation set, D4) y, en función de los resultados obtenidos sobre esas muestras, se anunciaron los vencedores. Entre todos los proyectos presentados, tres sistemas clásicos de aprendizaje automático supervisado salieron victoriosos. Seguidamente se muestra un resumen de los algoritmos del concurso que alcanzaron los mejores resultados, en orden decreciente.

1. **Frog**: Keli Liu, Christina Rabe, Paul Manser.
   - **Institución**: Genentech, USA.
   - **Predictores seleccionados**: selección automática y aumento de datos. Las categorías añadidas fueron: las mediciones más recientes, el tiempo transcurrido entre la medición más reciente, la muestra más alta y más baja, lapso de tiempo entre la muestra más grande y la más pequeña y el cambio más reciente en las mediciones.
   - **Algoritmo**: Gradient Boosting, implementado con XGBoost. Se entrenaron diversos modelos con variaciones en el tamaño de la ventana.
   - **Resultados**: mAUC = 0.931, BCA = 0.849.
2. **Threedays**: Paul Moore, Terry J. Lyons, John Gallacher.
   - **Institución**: Mathematical Institute, University of Oxford, 2 Department of Psychiatry, University of Oxford, UK.
   - **Predictores seleccionados**: edad, meses desde la base, género, raza, estado civil, diagnóstico, test cognitivos y estado de APOe.
   - **Algoritmo**: dos modelos de random forest. Uno para transiciones desde CN y otro para MCI. Parte de la suposición de que los pacientes de AD no pueden cambiar de estado clínico.
   - **Resultados**: mAUC = 0.921, BCA = 0.823.
3. **EMC-EB**: Esther E. Bron, Vikram Venkatraghavan, Stefan Klein.
   - **Institución**: Erasmus MC, Países Bajos.
   - **Predictores seleccionados**: 200 características: diagnóstico, FDG, PET, DTI, MRI, CSF y tests cognitivos. Los datos nulos fueron interpolados con la técnica de vecinos más cercanos. Cuando no se podía interpolar se utilizaba la media del conjunto de entrenamiento. Las muestras de pacientes sin diagnóstico fueron excluidas.
   - **Algoritmo**: máquinas de vectores de soporte (Support Vector Machine, SVM). El kernel tenía una función de base radial (Radial Basis Function, RBF), con el hiperparámetro C a 0.5.
   - **Resultados**: mAUC = 0.907, BCA = 0.805.

Este TFG se centra en los algoritmos de redes neuronales que participaron en el challenge. A continuación incluimos un resumen de tres ejemplos de los modelos basados en deep learning: el primero de redes neuronales pre-alimentadas, y otros dos basados en redes neuronales recurrentes, uno de ellos el mejor valorado de la categoría.

1. **SmallHeads – BigBrains**: Jacob Vogel, Andrew Doyle, Angela Tam, Alex Diaz-Papkovich.
   - **Institución**: McGill University, Montreal, Canada.

- **Predictores seleccionados**: solo los que tenían menos de la mitad de datos nulos, es decir, 376 características, que ulteriormente fueron normalizadas. Los datos nulos fueron interpolados con el método de cinco vecinos y basándose en la distancia euclídea.
- **Algoritmo**: red neuronal profunda y completamente conectada. A parte de los predictores se tenían en cuenta las marcas temporales. Cuenta con 5 capas con activación Leaky ReLU de 512, 512, 1024, 1024 y 256 neuronas. Utilizan drop out, optimización Adam y, como función de pérdida, categorical cross-entropy.
- **Resultados**: posición = 48, mAUC = 0.737, BCA = 0.605

2. **CBIL**: Minh Nguyen, Nanbo Sun, Jiashi Feng, Thomas Yeo.
   - **Institución**: National University of Singapore, Singapore.
   - **Predictores seleccionados**: test cognitivos, MRI, AV45, FDG y CSF. Los datos nulos fueron interpolados.
   - **Algoritmo**: red neuronal recurrente adaptada para una duración variable entre marcas temporales.
   - **Resultados**: posición = 9, mAUC = 0.897, BCA = 0.803.
3. **BGU-LSTM**: Aviv Nahon, Yarden Levy, Dan Halbersberg, Mariya Cohen.
   - **Institución**: Ben Gurion University of the Negev, Beersheba, Israel.
   - **Predictores seleccionados**: tests cognitivos, MRI, FDG- PET, AV45, PET, Volumen de las hipointensidades de la sustancia blanca, biomarcadores del CFS. Aumento de datos en 20 variables por cada variables continuas (e.g. media, desviación estándar, media de tendencia, desviación estándar de tendencia, mínimo, media menos media global, valor de línea de base, último valor observado). La LSTM utiliza una máscara para valores nulos.
   - **Algoritmo**: LSTM para variables continuas y una FNN para discretas.
   - **Resultados**: posición = 12, mAUC = 0.883, BCA = 0.779.

Dados estos algoritmos, pretendemos reproducirlos con las bibliotecas TensorFlow y Keras, escritas en python, y averiguar por qué sus puntuaciones están por debajo de los métodos clásicos.

1.3. Objetivos

Los propósitos establecidos para este trabajo de fin de grado son los siguientes:

1. Introducir las técnicas utilizadas actualmente en la práctica clínica en el diagnóstico de AD.
2. Documentar el estado del arte de TADPOLE Challenge.
3. Asimilar los fundamentos del aprendizaje profundo y adentrarnos en arquitecturas recurrentes.
4. Implementar métodos de predicción basados en deep learning con el ánimo de resolver el problema de diagnosticar el estado del paciente a corto y medio plazo.
5. Reproducir los resultados presentados en TADPOLE.
6. Dar con la combinación de características e hiperparámetros que conceda puntuaciones más altas.
7. Interpretar los algoritmos con la finalidad de determinar si las redes fundamentan sus razonamientos en la lógica y existe una base científica detrás de las decisiones tomadas.
8. Dar una explicación de por qué las puntuaciones de las redes neuronales están por debajo de las de las técnicas de tradicionales de aprendizaje automático.

1.4. Organización de la memoria

La memoria de este trabajo de fin de grado está conformada por cinco secciones principales y cuatro anexos. La primera sección nos concede una visión global del contexto en el que fue desarrollado el sistema, un resumen de cómo se llevó a cabo el concurso TADPOLE, sus modelos ganadores y los que vamos a tratar de reproducir. Además es en ese apartado donde se plasman los objetivos del proyecto y la organización de la memoria.

En segundo lugar, en la sección dos encontrarán una descripción de los algoritmos implementados, tanto los fundamentos de las arquitecturas más complejas como los detalles concretos de configuración de nuestros algoritmos. Por otro lado se menciona todo lo relativo a los conjuntos de datos utilizados en el entrenamiento y los test de nuestras redes neuronales, incluido el preprocesamiento.

En tercer lugar, contamos con un capítulo en el que reflejamos todos los experimentos realizados junto a los resultados obtenidos.

En cuarto lugar, hay una sección que introduce los fundamentos de la interpretabilidad: por qué es tan necesaria en un problema cómo este, qué técnicas se han utilizado, qué inconvenientes han surgido y cómo se han solucionado etc.

En último lugar, se exponen las conclusiones a las que se han llegado después de todo el trabajo realizado. Se resumen las aportaciones del TFG y se valora la consecución de los objetivos, la gestión del tiempo y alguna experiencia personal.

En los anexos encontrarán aquella información que por su carácter minucioso no pudo ser incluida en las secciones principales, pero que es imprescindible que conste en la memoria. Se trata de los principios básicos del aprendizaje profundo, los fundamentos de SHAP, el análisis completo de la interpretabilidad del sistema y un estudio sobre la configuración interna de las redes neuronales.

2. Métodos desarrollados

En las próximas páginas que constituyen esta sección, se realizará un análisis de los algoritmos de aprendizaje profundo implementados. Antes de nada, recomendamos echar un vistazo al anexo A, donde pueden encontrar una definición de lo que es el deep learning, entendido como *una serie de algoritmos de aprendizaje automático que están formados por la composición de transformaciones no lineales múltiples con el objetivo de producir representaciones de conocimiento más abstractas y, en última instancia, más útiles* [2], y una explicación fundamental de las nociones básicas. Si bien es un concepto unívoco que acoge tanto a métodos estáticos como a recurrentes, hemos considerado conveniente establecer una dicotomía entre ambas categorías y dedicar este capítulo para definir la arquitectura implementada. Por otra parte, la descripción de los conjuntos de datos y el preprocesamiento de estos también será desarrollada en este capítulo.

2.1. Métodos estáticos

Denominaremos como métodos estáticos a aquellos en los que la información se mueve en una única dirección (hacia delante). Estos métodos se conocen también como redes neuronales prealimentadas (Feed-forward Neural Networks, FNN). Con todos los conceptos introducidos en el anexo A, es hora de recapitular y ofrecer una visión global de los modelos implementados en este TFG.

En primer lugar tenemos una red neuronal poco profunda o Shallow. Es un modelo secuencial compuesto de una capa de input, cuyo tamaño depende del número de características con las que trabajamos; una capa densa, una de normalización de lotes (BacthNormalization), una ELU, dropout y, finalmente, la capa densa de salida. En la tabla 2.1 se ilustra la arquitectura de esta red.

Capa (tipo)	forma de output	parámetros #
input (InputLayer)	(None, num_features)	0
densa (Dense)	(None, 1024)	(num_features+1)*1024
normalización de lote(BatchNormalization)	(None, 1024)	(num_features+1))*1024
elu (ELU)	(None, 1024)	0
dropout (Dropout)	(None, 1024)	0
output (Dense)	(None, 3)	3075

Tabla 2.1: Resumen de Shallow de 1024 neuronas.

La capa principal posee un número variable de neuronas y la función de activación es RELU. Ulteriormente se le aplica un Batch Normalization, la función de activación ELU (definida en la figura A.2) y dropout. Por lo que se refiere al número de capas, podemos concatenar secuencialmente la estructura de capas ocultas expuesta anteriormente de diferentes maneras. Se han efectuado pruebas con el segundo tipo de modelo, de redes neuronales profundas (DNN) de dos, tres y hasta cuatro bloques de capas.

Se decidió que si el modelo estaba compuesto por dos bloques, las capas del segundo tenían la mitad de neuronas. Si se añadía un tercer agrupamiento, este dispondría de cuatro veces menos neuronas. Si en su lugar eran cuatro, se sigue la distribución marcada en la tabla 2.2. En la tabla 2.3 pueden encontrar el resumen de un esquema concreto.

bloque número	2 bloques	3 bloques	4 bloques
1º	NN	NN	NN
2º	NN/2	NN/2	3*NN/4
3º	-	NN/4	NN/2
4º	-	-	NN/4

Tabla 2.2: Número de neuronas por capas en función del bloque en el que se encuentre.

Capa (tipo)	forma de output	parámetros #
input (InputLayer)	(None, num_features)	0
densa (Dense)	(None, 1024)	(num_features+1)*1024
normalización de lote(BatchNormalization)	(None, 1024)	(num_features+1))*1024
elu (ELU)	(None, 1024)	0
dropout (Dropout)	(None, 1024)	0
densa 2 (Dense)	(None, 512)	(num_features+1)*512
normalización de lote 2(BatchNormalization)	(None, 512)	(num_features+1))*512
elu 2 (ELU)	(None, 512)	0
dropout 2 (Dropout)	(None, 512)	0
densa 3 (Dense)	(None, 256)	(num_features+1)*256
normalización de lote 3(BatchNormalization)	(None, 256)	(num_features+1))*256
elu 3 (ELU)	(None, 256)	0
dropout 3(Dropout)	(None, 256)	0
output (Dense)	(None, 3)	771

Tabla 2.3: Resumen de DNN de 1024 neuronas.

2.2. Métodos recurrentes

A la luz de la naturaleza temporal del proceso neurodegenerativo de la enfermedad de Alzheimer, parece más coherente entrenar el modelo con el historial completo del paciente, con miras a la compresión de la evolución de la enfermedad. Para dar un diagnóstico futuro preciso es necesario apreciar la dependencia temporal de los datos y analizar la evolución de los biomarcadores de meses anteriores. Este planteamiento se basa en incluir nociones de recurrencia y bucles de realimentación. Las RNN (Recurrent neural network; RNN) se diferencian de las FNN por la presencia de ciclos entre sus nodos [25].

Comenzamos variando la forma del input, que ahora no es un vector de características, sino que es un vector de t marcas temporales. Cada una de estas marcas alberga los biomarcadores de ese mes. Lógicamente, los pacientes no se examinan todos los meses, de ahí que no contemos con la mayor parte de la información y sea necesario completarla de alguna manera (imputación). Existen distintos métodos de imputación como, por ejemplo: rellenar con la media, copiar la marca más reciente o rellenar con ceros. Hemos tomado la determinación de completar la información carente con ceros, aunque haciendo uso de una máscara. Así que a la hora de entrenar la red, cuando detecte una secuencia compuesta exclusivamente por ceros, la ignorará. Como muestra la figura 2.4, se aplican las operaciones de la neurona a cada elemento del vector individualmente, y esta devuelve un diagnóstico para cada marca temporal, administrando el conocimiento asimilado en iteraciones precedentes.

Dentro de los modelos recurrentes podemos distinguir entre distintos tipos de arquitecturas en función de la clases de input y output. Si la entrada es un único elemento y la salida es una secuencia, nos encontramos ante un esquema one to many. Si es al revés, se trata de un esquema many to one. Cuando ambas partes son una secuencia, es una arquitectura many to many, como la empleada en este problema. Dentro de sus variantes optaremos por la versión sincronizada, la que se muestra más a la derecha en la figura 2.5. Eso se debe a que se le traslada una secuencia de biomarcadores y devuelve una secuencia de diagnósticos. Recordamos que la i-ésima salida

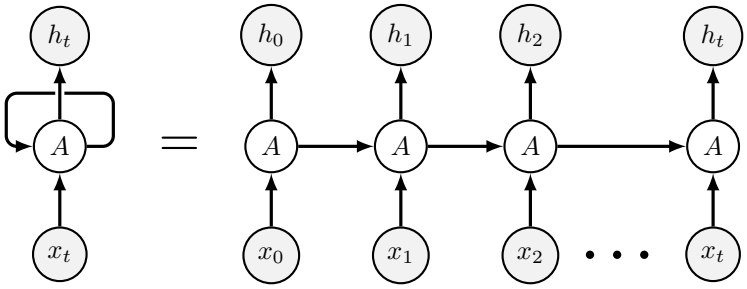


Figura 2.4: Esquema del funcionamiento de una RNN.

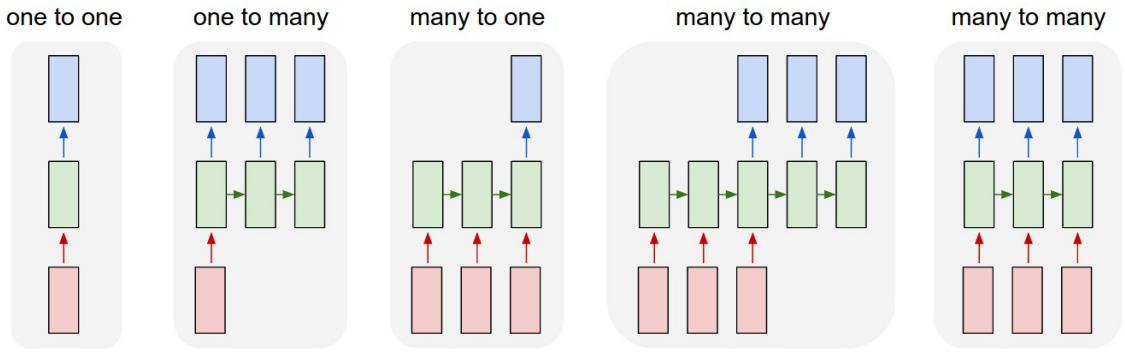


Figura 2.5: Distintas clases de arquitecturas en modelos recurrentes. (Imagen extraída del blog de Andrej Karpathy http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (último acceso 3 de Junio de 2021))

corresponde al estado clínico asociado a la i-ésima entrada.

Las capas de las redes recurrentes pueden agruparse secuencialmente y que el output de una capa actúe como la entrada de la siguiente. Esto ocurre en algunos modelos implementados, al tratarse de una arquitectura many to many, las capas intermedias reciben secuencias y envían secuencias temporales a la siguiente. Más concretamente, se han desarrollado estructuras de una única capa y 128 neuronas; de dos capas, con 25 y 128 neuronas, y de tres, con 64 neuronas en la tercera (tablas 2.6 y 2.7).

Capa (tipo)	forma de output	parámetros #
máscara (Masking)	(None, 120, num_features)	0
lstm (LSTM)	(None, 120, 128)	171008
output (Dense)	(None,120, 3)	387

Tabla 2.6: Resumen de LSTM de 1 capa.

Muchos métodos de prevención de overfitting empleados en los modelos anteriores son válidos también aquí, como el dropout y los regularizadores. Se ha observado que estas técnicas son más sensibles a los cambios en la regularización. Al aplicar una norma L2 con todas las características, se sobreajusta. Lo mismo ocurre con la norma L1 para ciertos valores. Por lo general no se suele utilizar Batch Normalization en modelos recurrentes, porque añaden mucho ruido. Es cierto que algunos autores han conseguido introducirlo en su LSTM, sin embargo no lo hicieron con keras de manera simple, sino que tuvieron que hacer cambios en la capa a bajo nivel, adaptando las fórmulas y los parámetros [4].

Capa (tipo)	forma de output	parámetros #
máscara (Masking)	(None, 120, num_features)	0
lstm_0 (LSTM)	(None, 120, 25)	23100
lstm_1 (LSTM)	(None, 120, 128)	78848
lstm_2 (LSTM)	(None, 120, 64)	49408
output (Dense)	(None,120, 3)	387

Tabla 2.7: Resumen de LSTM de 3 capa.

2.2.1. Red neuronal recurrente simple

Como se ha explicado anteriormente, buscamos añadir a la red neuronal convencional la capacidad de memorizar información de estados previos, de forma que podamos trabajar con secuencias enteras en vez de con datos aislados. Esta es la razón por la cual a la ecuación original se le agrega el factor de activación, también conocido como estado oculto (hidden state). La lógica es la misma, la activación es fruto de aplicar una transformación a los datos de entrada y a la activación anterior. Así es como se concede a los estados previos la alusión buscada. Evidentemente al trabajar una nueva variable, la ecuación tiene un peso más. Ambos pesos se aproximan por medio del entrenamiento, como se hacía en las redes simples. En último lugar, aclarar que la salida se obtiene a partir de la activación actual, tal y como muestra la ecuación [7].

$$\begin{aligned} h_t &= \tanh(W_x x_t + W_h h_{t-1} + b_h), \\ y_t &= softmax(W_y h_t + b_t) \end{aligned} \tag{2.1}$$

Si bien en ocasiones se representa la red neuronal recurrente como varias celdas concatenadas una con otra (figura 2.4 a la derecha), lo cierto es que hay una sola celda y, en consecuencia, un peso de cada tipo, que se van ajustando durante el aprendizaje (figura 2.4 a la izquierda).

En la práctica se pueden concatenar varias capas recurrentes y de hecho, en los experimentos llevados a cabo en este trabajo se han implementado este tipo de modelos. El funcionamiento es análogo, sólo que la activación de la primera actúa como la entrada de la segunda.

$$\begin{aligned} h_t &= \tanh(W_x x_t + W_h h_{t-1} + b_h), \\ h'_t &= \tanh(W'_x h_t + W'_h h'_{t-1} + b'_h), \\ y_t &= softmax(W_y h'_t + b_t) \end{aligned} \tag{2.2}$$

2.2.2. Long short-term memory (LSTM)

El modelo expuesto anteriormente es práctico a la hora de trabajar con secuencias cortas. Sin embargo, con un input más extenso en el tiempo presenta problemas de rendimiento. Esto se debe a que, al desarrollar la fórmula de la red recurrente, se aprecia que, a medida que se itera, el valor de las primeras activaciones se queda anidado en funciones de tanh, lo que supone una pérdida de peso en el resultado final. Este fenómeno se denomina el problema de desvanecimiento de gradiente [12] y se mitiga introduciendo un elemento adicional a la red, una celda de memoria. En esta celda se "almacenarán" los elementos ya comprendidos que la red considere relevantes. Esa será la manera de proporcionar al algoritmo memoria a medio y largo plazo.

Queda por explicar cómo se selecciona esa información. Una celda de LSTM tiene tres puertas: de olvido, entrada y salida (figura 2.8). Los valores de cada puerta se establecen por medio del aprendizaje. Cada una tiene sus propios pesos y sesgos y dependen de los datos de entrada y la activación anterior. Es similar al cálculo de la activación en la red neuronal simple, salvo que se

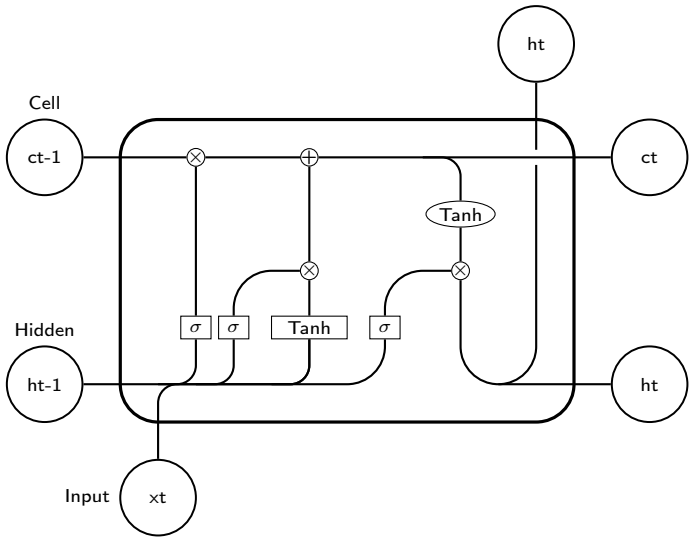


Figura 2.8: Esquema de la composición interna de una celda de LSTM.

utiliza de puerta una sigmoidal en vez de una tangente hiperbólica con el propósito de otorgar un comportamiento de puerta binario.

En cada iteración lo primero que se realiza es el borrado de la célula de los datos que ya no se consideran de utilidad. El descarte se hace por medio de un producto entre el vector f_t (o puerta de olvido) y la memoria. Es muy intuitivo: si el valor de f_t es cercano a cero, al multiplicarlo con la parte correspondiente de la célula, perderemos esa información, mientras que si es cercano a 1, la mantendremos durante una iteración más. La escritura en memoria es fruto del producto entre la puerta de entrada y la entrada de la celda ($\tilde{c}$), se basa en el mismo funcionamiento que la operación anterior. El estado final de la memoria es la unión de los elementos anteriores después de hacer la criba y los agregados. Antes de terminar, la puerta de salida controla que datos de la célula se transmitan a la activación.

$$\begin{aligned} i_t &= \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \\ f_t &= \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \\ \tilde{c}_t &= \tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \\ o_t &= \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \end{aligned} \tag{2.3}$$

$$\begin{aligned} c_t &= f_t c_{t-1} + i_t \tilde{c}_t \\ h_t &= o_t \tanh(c_t) \end{aligned}$$

Este tipo de red ha adquirido popularidad durante los últimos años porque trabaja bien tanto con secuencias largas como cortas, y no tienen el problema del desvanecimiento del gradiente. A día de hoy se utilizan en otras aplicaciones médicas como la detección de homología de proteínas [13] o, como en este caso, el diagnóstico de enfermedades [3].

2.3. Conjuntos de datos

Como se ha indicado anteriormente los datos del concurso TADPOLE fueron extraídos de la base de datos de ADNI. ADNI es un proyecto que surge en 2004 de la iniciativa público-privada y tiene como propósito el estudio de la enfermedad de Alzheimer en las etapas más tempranas.

Con el fin de catalizar la investigación, pusieron a disposición de todos los investigadores que lo solicitaron una copiosa cantidad de información. Gracias a ello, se estima que en 2014 unas mil de publicaciones científicas se valieron de los datos de ADNI [23] y este número se ha incrementado substancialmente en los últimos años.

Dentro de ADNI podemos diferenciar entre cuatro estudios. Primeramente se fundó ADNI-1, a fin de desarrollar biomarcadores como medidas de resultado para ensayos clínicos. En cinco años recopilaron diagnósticos de 200 pacientes CN, 400 MCI y 200 AD. En 2009, se amplió la base de datos con 200 sujetos más en fase temprana de deterioro cognitivo, ya que el proyecto ADNI-GO (Grand Opportunities) buscaba analizar los biomarcadores de este tipo de pacientes. Un par de años más tarde surgió ADNI-2 que añadió información de 150 de personas en CN, 100 en la primera etapa de MCI, 150 de MCI más tardío y 150 diagnosticados de Alzheimer. En la actualidad están desarrollando ADNI 3, cuyo objetivo es reunir datos para el estudio del uso de tau PET y técnicas de imagen. En la tabla 2.9 se resume el tipo de información con la que cuenta cada conjunto.

Study techniques	ADNI-1	ADNI-GO	ADNI-2	ADNI-3
Imaging				
MRI				
Structural	X	X	X	X
Perfusion		X	X	X
Resting state		X	X	X
Diffusion		X	X	X
Connectomics				X
High resolution				X
PET				
Glucose metabolism	X	X	X	
$\beta - amyloid$	[^{11}C] Pittsburgh compound	[^{18}F] florbetapir	[^{18}F] florbetapir	[^{18}F] florbetapir/Florbetaben
Tau				[^{18}F] T807
Biosamples				
CSF -amyloid, tau	X	X	X	X
Genetic analysis				
APOE	X	X	X	X
Genome wide association studies	X	X	X	X
Whole genome sequencing			X	X
Systems biology approaches				X
Neuropsychological tests	X	X	X	X
Autopsy	X	X	X	X

Tabla 2.9: Tipo de información comprendida en cada estudio de ADNI [22].

Cuando se presentó el desafío de TADPOLE, se constituyeron tres conjuntos de datos distintos (figura 2.10), cuya fuente era ADNI-1, ADNI-GO y ADNI-2. El primer conjunto, D1, lo forman 1667 pacientes. De cada uno hay un mínimo de dos visitas y en total hay 8841. Son datos longitudinales extensivos, pues se cuenta con 1886 predictores. En segundo lugar, D2 está conformado por 896 individuos, de los cuales solamente 13 no están registrados en D1. En total hay 5177 nuevos diagnósticos. Finalmente se incluyó D3, con las mismas personas que en D2 pero con un subconjunto de características de cardinalidad 382 y donde sólo figura su última visita. Estas 382 características son las que suele utilizar el personal sanitario en el diagnóstico de la AD. A este conjunto se le conoce también como cross-sectional. Asimismo la organización admitía el aumento de datos o utilización de otros diagnósticos.

Una vez que hubo terminado la competición se publicó el conjunto D4, con información extraída de ADNI-3, que sirvió de test para los modelos presentados. Este dataset no contenía todos los biomarcadores, solamente constaba de 13 features elementales. Aquellos que obtuvieron mayor puntuación con D4 se erigieron como campeones. En la tabla 2.11 se ha facilitado un resumen de la información demográfica de los pacientes de TADPOLE. En algunos casos la suma de porcentajes no llega al cien porque hay una serie de pacientes que no fueron diagnosticados. En D4 los cambios se producen de MCI que anteriormente eran CN o de AD que otrora eran MCI o CN. Los cambios en el resto de conjuntos también han ido a peor, sujetos que eran CN o MCI y han pasado a MCI

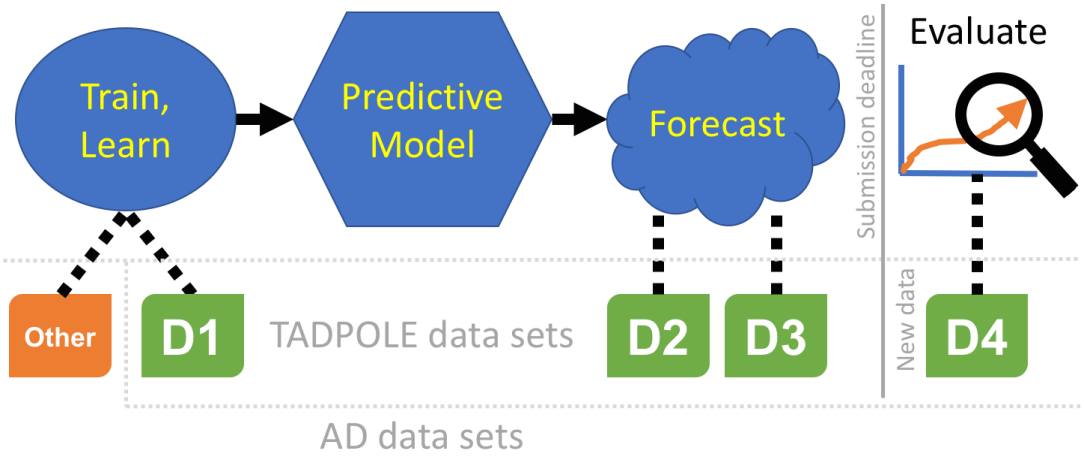


Figura 2.10: Representación de los conjuntos de datos de TADPOLE challegen. (Imagén obtenida de la página web de TADPOLE https://tadpole.grand-challenge.org/Details/#Data%20sets (último acceso 3 de Junio de 2021))

o AD.

		D1	D2	D3	D4
Número de pacientes		1667	896	896	219
CN	Cardinalidad (% sobre el total)	508 (30,5 %)	369(41,2 %)	299(33,4 %)	94(42,9 %)
	Visitas por paciente	8,3±4, 5	8,5±4, 9	1,0±0, 0	1,0±0, 2
	Edad	74,3±5, 8	73,6±5, 7	72,3±6, 2	78,4±7, 0
	Sexo (% de hombres)	48,6 %	47,2 %	43,5 %	47,9 %
	MMSE	29,1±1, 1	29,0±1, 2	28,9±1, 4	29,1±1, 1
	Cambios	18	9	-	-
MCI	Cardinalidad (% sobre el total)	841 (50,4 %)	458(51,1 %)	269(30,0 %)	90(41,1 %)
	Visitas por paciente	8,2±3, 7	9,1±3, 6	1,0±0, 0	1,0±0, 3
	Edad	73,0±7, 5	71,6±7, 2	71,9±7, 1	79,4±7, 0
	Sexo (% de hombres)	59,3 %	56,3 %	58,0 %	64,4 %
	MMSE	27,6±1, 8	28,0±1, 7	27,6±2, 2	28,1±2, 1
	Cambios	117	37	-	9
AD	Cardinalidad (% sobre el total)	318 (19,1 %)	69(7,7 %)	136(15,2 %)	29(13,2 %)
	Visitas por paciente	4,9±1, 6	5,2±2, 6	1,0±0, 0	1,1±0, 3
	Edad	74,8±7, 7	75,1±8, 4	72,8±7, 1	82,2±7, 6
	Sexo (% de hombres)	55,3 %	68,1 %	55,9 %	51,7 %
	MMSE	23,3±2, 0	23,1±2, 0	20,5±5, 9	19,4±7, 2
	Cambios	-	-	-	9

Tabla 2.11: Resumen demográfico de los perfiles del concurso [17]

TADPOLE sugiere utilizar D1 como conjunto de entrenamiento y D2 y D3 como conjuntos de validación. No obstante, en este TFG se ha utilizado D1 y D2 conjuntamente para formar los grupos de aprendizaje y validación.

En cuanto al aumento de datos, se ha seguido el método de Frog, el grupo que quedó en primera posición, para aumentar la información del diagnóstico y así mejorar el rendimiento de los modelos. El conjunto aumentado que se ha construido añade a cada visita el diagnóstico más reciente, el peor hasta la fecha y el mejor; el tiempo transcurrido desde la última consulta, desde el peor diagnóstico hasta la fecha y desde el mejor; el cambio de estado clínico más reciente y cuántos meses hace de ello. Este conjunto aumentado se ha utilizado para algunos de los experimentos.

2.4. Preprocesado de datos.

La primera operación en nuestros programas es la selección de datos. En función del data leakage podemos discernir entre tres métodos de selección. En primer lugar, podemos tener un conjunto de entrenamiento con total leakage, lo que supone que ese conjunto se formará a partir de la unión de D1 y D2, de modo que el conjunto de test estará contenido en el de entrenamiento. En segundo lugar, se puede quitar de este conjunto el conjunto de test, con lo que el conjunto de test ya no está contenido en el de entrenamiento, pero sí existen datos relativos a los pacientes del conjunto de test. Este tipo de data leakage se conoce como patient leakage. En tercer lugar, se puede retirar toda la información relativa a los pacientes del test, consiguiendo un conjunto de entrenamiento sin leakage con el conjunto de test. Esa última opción es la que consideramos más correcta. Sin embargo, muchos de los métodos de TADPOLE trabajaron con el primer y segundo conjunto.

Existen estudios en la literatura relativa al diagnóstico de Alzheimer en los que existe o se sospecha de data leakage. Esto supone un problema puesto que los resultados conseguidos en dichos estudios están contaminados por un mal diseño experimental y las métricas obtenidas suelen ser muy superiores a las verdaderas capacidades del modelo. Este problema se está estudiando hacia la obtención de métodos reproducibles y resultados generalizables entre diferentes datasets y debería ser estudiado en los métodos del Tadpole challenge [24].

Una vez definido cada dataset se deben someter a operaciones de preprocesado. La primera característica a procesar es la de la columna DXCHANGE, dónde se encasillan a los diagnosticados por su cambio de estado clínico. En la tabla 2.12 se detalla la codificación de DXCHANGE y la sustitución aplicada. Se trata de reemplazar la información de esa columna por el diagnóstico actual y además se renombra la columna como Diagnosis.

Cambio	Código DXCHANGE	Sustitución
CN $\rightarrow CN$	1	1
MCI $\rightarrow MCI$	2	2
AD $\rightarrow AD$	3	3
CN $\rightarrow MCI$	4	2
MCI $\rightarrow AD$	5	3
CN $\rightarrow AD$	6	3
MCI $\rightarrow CN$	7	1
AD $\rightarrow MCI$	8	2
AD $\rightarrow CN$	9	1
Dato no disponible	-1	-1

Tabla 2.12: Representación de la codificación de DXCHANGE junto a los nuevos valores asignados.

La segunda operación en realizarse es la suma de los meses a la edad, para disponer de la medida exacta. Acto seguido desestimamos aquellos predictores basados en conocimientos previos. En la columna DX_bl ocurre algo similar a DXCHANGE, hallamos variables categóricas definidas como cadenas de caracteres, por consiguiente, habrá que sustituir las etiquetas por enteros. DX_bl no es la única columna que contiene este tipo de variables textuales, algo análogo ocurre con el sexo, la raza y el estado civil. En la tabla 2.13, se han recogido todos los reemplazos realizados en esas columnas. Cabe decir que antes de procesar las cadenas de texto, se deben obliterar todos los espacios por medio de una expresión regular.

Para el resto de categorías también se debe hacer una transformación numérica, en la medida de lo dable. Aquellos datos que no puedan ser convertidos, se sustituirán por NaN.

En cuanto al volumen intracerebroventricular (ICV), si ICV_bl es cero, se sustituye por la media. Además se añade una nueva columna Ventricles_ICV definida como ratio de ventriculos

Columna	Valor inicial	Codificación
Gender	Male	0
	Female	1
Race	White	0
	Black	1
	Asian	2
Marital status	Married	0
	Divorced	1
	Never married	2

Tabla 2.13: Codificación de las variables categóricas.

por ICV. Por otro lado, se elimina la columna EXAMDATE, ya que es un dato que no aporta información al entrenamiento. Al final del preprocesado se ordenan los elementos por identificador de lista (roster ID, RID) y edad.

Todavía queda pendiente efectuar la interpolación y el tratamiento de los NaN. Para este trabajo se ha decidido sustituir los valores nulos por la media de ese predictor. Aún con ello no fue suficiente, por eso a los escasos NaN que restaban se les asignó cero como valor.

Dentro de cada conjunto, conviene hacer la división entre datos y etiquetas (target). En este problema el output es el diagnóstico del paciente a futuro. Por supuesto, las personas que no cuenten con una evaluación definida serán descartadas. No hay que olvidar que la salida de la red es un vector de tamaño tres con el estado clínico codificado en formato one-hot (CN = [1,0,0]; MCI = [0,1,0]; AD = [0,0,1]), así que hay que hacer los cambios pertinentes antes del aprendizaje. Del mismo modo, el input se debe normalizar. Se ha utilizado la clase StandardScaler de sklearn, que aplica la fórmula de normalización.

$$\hat{x} = \frac{x - \mu}{\sigma} = \frac{x - \frac{1}{N}\sum_{i=1}^{N} x_i}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}} \tag{2.4}$$

En las redes neuronales recurrentes se sigue el mismo procedimiento pero con dos excepciones. Por un lado, la normalización se hace aplicando la fórmula, en vez de llamar a la clase StandardScaler. Por otro lado, la forma del input es distinta, ya no se trabaja con datos aislados, sino con secuencias. Hace falta entonces agrupar el dataset por pacientes y de cada paciente almacenar los biomarcadores (o el diagnóstico, si se trata de la secuencia de salida) por meses. Recordamos que no es cuestión de tener una secuencia variable con los meses ordenados, la secuencia es de longitud fija, de diez años de duración. En esos diez años solo se tendrá información de un conjunto de meses, al resto se les aplicará la máscara. Aclarar también que el índice de cada elemento de la secuencia corresponde a los meses transcurridos desde el primer diagnóstico.

Se debe señalar que hay un par de filas con información duplicada para un paciente en un mismo mes en el conjunto de datos original. Es un hecho de lo más anecdótico pero, si no se tiene en cuenta, durante la ejecución pueden aparecer errores.

3. Resultados

Tras haber expuesto los modelos desarrollados y todos los hiperparámetros que afectan al aprendizaje (véase el apéndice A y la sección 2), llega el momento de poner a prueba los algoritmos y razonar qué ajuste es el óptimo. La forma de determinar cómo de útiles son nuestros modelos será a través de comparaciones con los resultados presentados por los aspirantes de TADPOLE.

Como se ha dicho con anterioridad (sección 1.2), las medidas de evaluación son el área multiclase bajo la curva de característica operativa del receptor ($mAUC_{ROC}$) y la precisión de clasificación equilibrada (BCA), cuyas fórmulas definimos aquí debajo [11].

$$\begin{aligned} mAUC &= \frac{2}{L(L-1)} \sum_{i=2}^{L} \sum_{j=1}^{i} \hat{A}(c_i, c_j), \\ \hat{A}(c_i, c_j) &= \frac{\hat{A}(c_i|c_j) + \hat{A}(c_j|c_i)}{2} \\ \hat{A}(c_i|c_j) &= \frac{S_i - n_i(n_i+1)/2}{n_i n_j} \end{aligned} \tag{3.1}$$

donde L es el número de clases, $\hat{A}(c_i|c_j)$ la formula del AUC, n_i la cardinalidad de c_i y S_i la suma de rangos de los puntos de test de la clase i.

$$\begin{aligned} BCA &= \frac{1}{L} \sum_{i=1}^{L} BCA_i, \\ BCA_i &= \frac{1}{2}(Sensibilidad + Especificidad) = \frac{1}{2}(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}) \end{aligned} \tag{3.2}$$

En todos los test se realizarán dos tipos de predicciones, una sobre el conjunto D2 y otra sobre D4. La primera es propia de un problema de clasificación, dados unos biomarcadores nos dice cuál es el diagnóstico, mientras que la segunda es una predicción en toda regla, ya que hay que atribuir el diagnóstico a futuro. En lo alto de cada celda de las tablas de este capítulo se indican los resultados de las predicciones sobre D2 y debajo los de D4.

Los resultados de TADPOLE muestran que con una selección manual o automática de características se obtienen mejores resultados. Por este motivo hemos realizado dos clases de experimentos: con todas las features y con un cribado. Dentro de la elección de las mejores características se han realizado pruebas con las 210 características que seleccionó el grupo EMC-EB (diagnóstico, FDG, PET, DTI, MRI, CSF y tests cognitivos) y las 27 de CBIL (test cognitivos, MRI, AV45, FDG y CSF).

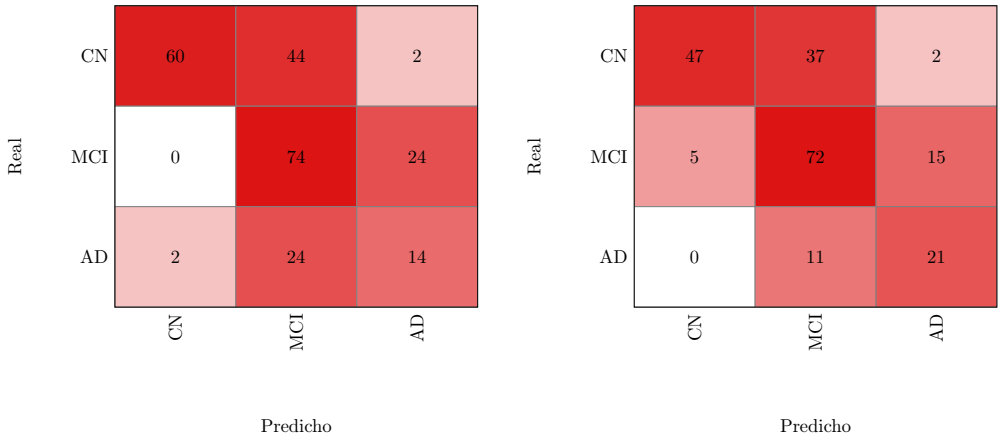
3.1. Reproducibilidad en modelos estáticos

La primera de las pruebas se realizará con las 1822 características (tabla 3.1 y figura 3.2). Los hiperpárametros que se van a variar son el drop-out, el número de neuronas y la norma (L1 o L2). Como la dimensionalidad del problema puede crecer de manera combinatoria se han acotado los valores que pueden tomar estas variables. En el caso del dropout, nos moveremos con cifras de 0,15 ó 0,3. El número máximo de neuronas que contiene una capa son de 20 ó 1024, en la sección anterior (tabla 2.2) explicamos cómo se determina la cantidad de neuronas para cada capa en los modelos profundos, dado que no es la misma en todas. En cuanto a la regularización, hacemos distinción entre LASSO y Ridge con valores de alfa de 0.01.

La conclusión más evidente que podemos extraer es que la norma L1 da mejores resultados en este tipo de modelos que la norma L2. Por otro lado se observa que, con tantos predictores, las redes

Todos los predictores			Shallow		DNN 2 capas		DNN 3 capas		DNN 4 capas	
norma	nº de neuronas	dropout	*mAUC*	*BCA*	*mAUC*	*BCA*	*mAUC*	*BCA*	*mAUC*	*BCA*
L1	20	0.15	0.963	0.840	0.982	0.892	**0.981**	**0.858**	**0.983**	**0.863**
			0.817	0.726	0.862	0.744	**0.845**	**0.772**	**0.863**	**0.763**
		0.3	**0.977**	**0.873**	0.952	0.875	0.954	0.837	0.974	0.879
			0.833	**0.739**	0.796	0.715	0.816	0.716	0.817	0.723
	1024	0.15	0.958	0.806	**0.989**	**0.911**	**0.985**	**0.926**	0.936	0.750
			0.819	0.673	**0.857**	**0.755**	**0.848**	**0.760**	0.806	0.689
		0.3	0.947	0.865	0.976	0.925	0.978	0.913	0.986	0.874
			0.818	0.730	0.806	0.712	0.837	0.744	0.825	0.740
L2	20	0.15	0.935	0.782	0.887	0.729	0.935	0.803	0.921	0.818
			0.837	0.683	0.809	0.686	0.822	0.739	0.812	0.726
		0.3	0.931	0.808	0.917	0.759	0.916	0.793	0.903	0.763
			0.829	0.700	0.815	0.697	0.827	0.694	0.810	0.666
	1024	0.15	0.939	0.777	0.901	0.753	0.945	0.815	0.917	0.791
			0.838	0.687	0.829	0.713	0.858	0.718	0.802	0.691
		0.3	0.927	0.755	0.924	0.803	0.912	0.778	0.917	0.766
			0.834	0.689	0.814	0.695	0.822	0.711	0.820	0.705

Tabla 3.1: Tests realizados sobre los modelos estáticos con todos los predictores. (En la parte de arriba de cada celda aparecen los resultados sobre D2 y debajo los de D4).



(a) Sobre D2

(b) Sobre D4

Figura 3.2: Matriz de confusión de red Shallow con todas las features.

profundas son más eficientes que la Shallow. Además advertimos que no existe mucha diferencia entre las variaciones de dropout y número de neuronas ni ninguna combinación que domine a las demás.

Si nos detenemos a observar las curvas de aprendizaje (figura 3.3), se llega a la máxima precisión relativamente pronto. Después converge sobre 0.93, aunque la línea no es del todo recta. Al hacer zoom sobre la curva de la función de pérdida se ve con más claridad la sinuosidad. Es evidente que se han procesado más épocas de las necesarias.

A continuación procedemos a realizar los mismos experimentos con una selección de predictores a priori (tabla 3.4 y figura 3.5).

Ya no existe una diferencia tan grande entre LASSO y Ridge. Parece que un modelo sencillo de 20 neuronas y poco dropout es la mejor opción cuando se trabaja con 210 características, en caso de que aumente el número de neuronas es aconsejable aumentar el dropout para mantener los resultados. Asimismo se observa que a mayor número de capas, mejor precisión.

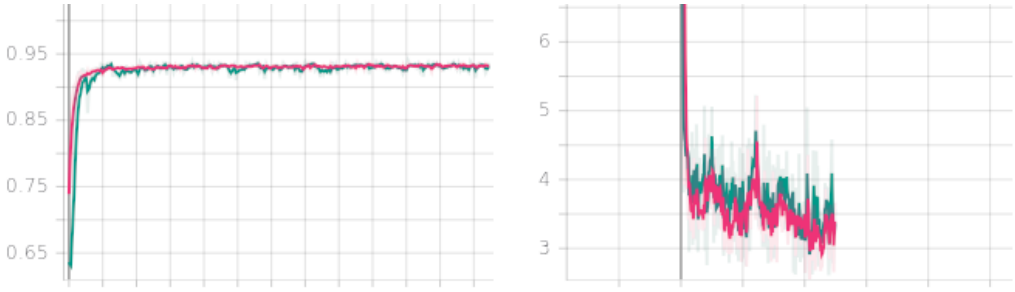
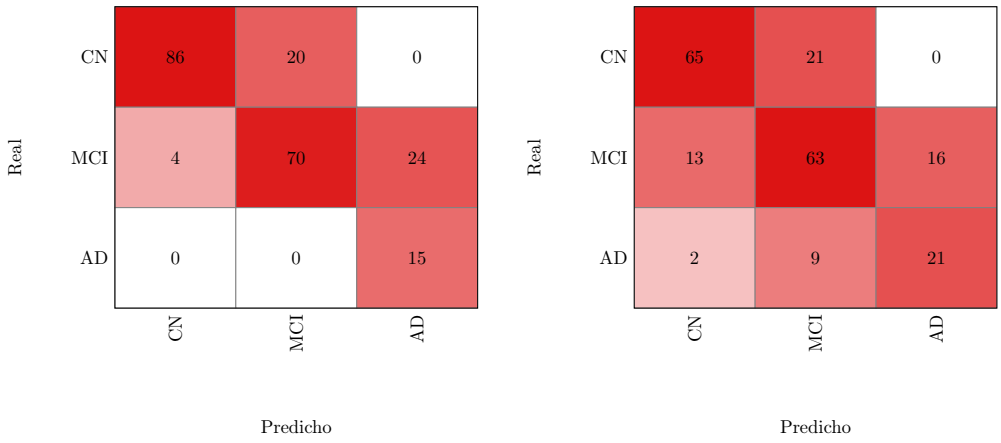


Figura 3.3: Curva de aprendizaje de la red Shallow con todos los predictores. (Precisión por época (izq.) y pérdida por época (dcha.))

Selección de predictores			Shallow		DNN 2 capas		DNN 3 capas		DNN 4 capas	
nº predictores	nº de neuronas	dropout	*mAUC*	*BCA*	*mAUC*	*BCA*	*mAUC*	*BCA*	*mAUC*	*BCA*
210	20	0.15	0.962	0.855	**0.984**	**0.892**	**0.993**	**0.947**	**0.984**	**0.921**
			0.851	0.751	**0.857**	**0.770**	**0.842**	**0.758**	**0.847**	**0.774**
		0.3	0.965	0.857	0.959	0.799	0.936	0.740	0.972	0.892
			0.853	0.746	0.837	0.712	0.828	0.694	0.834	0.748
	1024	0.15	0.930	0.765	0.983	0.887	0.869	0.766	0.968	0.840
			0.832	0.695	0.841	0.745	0.791	0.708	0.827	0.683
		0.3	**0.967**	**0.868**	0.976	0.902	0.986	0.934	0.987	0.919
			0.845	**0.772**	0.848	0.748	0.822	0.741	0.837	0.772
27	20	0.15	0.974	0.866	0.947	0.780	0.975	0.875	0.984	0.920
			0.855	0.754	0.859	0.713	0.849	0.738	0.847	0.768
		0.3	0.978	0.907	0.982	0.886	0.977	0.865	0.963	0.852
			0.856	0.766	0.858	0.754	0.827	0.721	0.824	0.721
	1024	0.15	0.951	0.821	0.997	0.954	0.977	0.864	0.983	0.916
			0.847	0.758	0.856	0.749	0.809	0.708	0.822	0.767
		0.3	0.966	0.859	0.975	0.892	0.922	0.749	0.980	0.879
			0.860	0.744	0.853	0.743	0.825	0.711	0.846	0.758

Tabla 3.4: Tests realizados sobre los modelos estáticos tras una selección de características (En la parte de arriba de cada celda aparecen los resultados sobre D2 y debajo los de D4).



(a) Sobre D2

(b) Sobre D4

Figura 3.5: Matriz de confusión de red Shallow tras seleccionar predictores.

La selección no nos ha supuesto una mejoría como a otros participantes. Seguramente, y en la próxima sección lo corroboramos, se debe a que la red está dando un peso sustancialmente mayor a las features más importantes, de ahí que la gran mayoría de las 1822 features están teniendo un impacto ínfimo en el diagnóstico final.

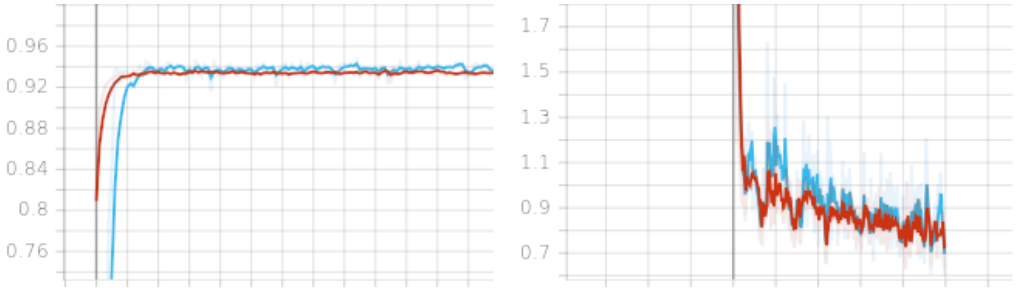


Figura 3.6: Curva de aprendizaje de la red Shallow tras seleccionar predictores. (Precisión por época (izq.) y pérdida por época (dcha.))

La gráfica del aprendizaje (figura 3.6) muestra una mayor velocidad, comienza con una precisión más alta que anteriormente (figura 3.3) y se estabiliza antes en 0.93. Además la función de pérdida ya no presenta esos dientes de sierra tan pronunciados y oscila dentro de un rango de valores más pequeño.

A la vista de estas puntuaciones, cabe preguntarse qué pudo fallar en el modelo de Small Heads, que recordamos que consiguió un mAUC de 0,737 y un BCA de 0,605 sobre el conjunto de test D4. En primer lugar, su arquitectura era bastante más compleja, contaba con cinco capas de cientos de neuronas con dropout bastante alto. Otro rasgo característico es el input delta_t que añadieron a la red. La finalidad era añadir un factor temporal que tuviera en cuenta la evolución de los pacientes. A primera vista no podemos asegurar si ese elemento supuso algún beneficio, ya que no disponemos de las puntuaciones con y sin delta. A mi juicio es una concepción un tanto enrevesada, simplemente podrían haber probado con modelos recurrentes para evaluar el carácter temporal del problema. Tampoco terminamos de descartar la conjetura de que ellos se desentendieran de algún atributo importante, como puede ser el estado clínico en la visita de baseline (DX_bl).

3.2. Reproducibilidad en modelos recurrentes.

En problemas con una escasa cantidad de datos los métodos de redes neuronales con un gran número de parámetros sufren de sobreajuste. Hasta ahora, para las técnicas de aprendizaje automático sencillas y las redes neuronales simples, contábamos con un volumen de datos suficiente. No obstante al pasar a trabajar con redes recurrentes nos vamos a encontrar con muchísimo overfitting que solo puede deshacerse a través de un filtrado de predictores. Para ello, hemos decidido desprendernos de los features UCS, que en su mayoría son combinaciones lineales de otros datos de ADNIMERGE.

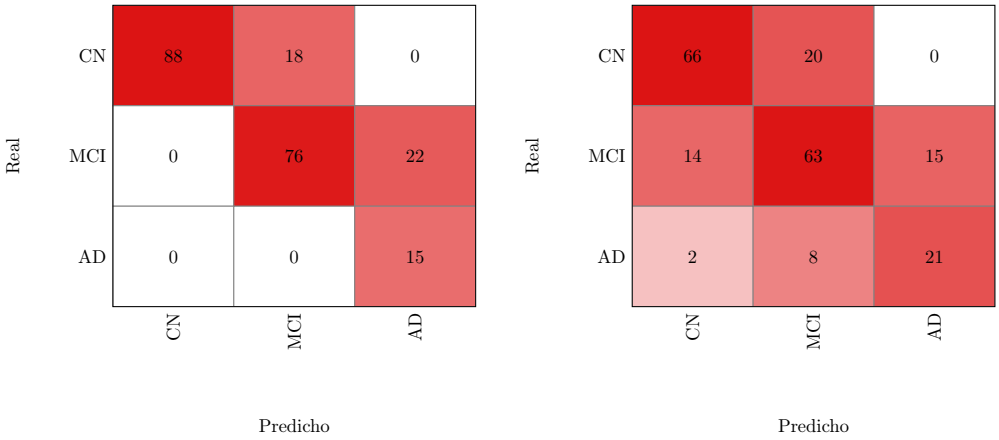
Tras el filtrado se observó una disminución considerable del sobreajuste en la mayoría de pruebas. No obstante, la red seguía fallando en aquellas que utilizaban la norma L2 en vez de la norma L1. Como se ha mencionado previamente, estos algoritmos son muy sensibles a los cambios de los valores de la norma, así que en las pruebas finales se han tenido en cuenta el valor alfa (0.0001) y el regularizador (L1) que no presentaban inconvenientes. El resto de variantes sólo han producido experimentos fallidos.

Nuevamente volvemos a asignar dropout de 0,15 ó 0,3 y esta vez no consideramos tanto el número de neuronas sino que nos centramos en la estructura en general. Las tablas 2.6 y 2.7 resumen las arquitecturas de red que vamos a testear, en esa sección (2.2) pueden encontrar información más detallada.

Lo primero que llama la atención es que la precisión es mayor que en los modelos simples (tabla 3.7 RNN, tabla 3.1 DNN y figura 3.8). Sobre D2 los modelos de tres capas ofrecen una

Todos los predictores salvo los UCS		RNN simple		LSTM	
nº de capas	dropout	*mAUC*	*BCA*	*mAUC*	*BCA*
2	0.15	0.984	0.90	0.951	0.810
		0.850	0.768	0.859	0.745
	0.3	**0.976**	**0.892**	0.965	0.863
		0.860	**0.782**	0.872	0.771
1	0.15	0.983	0.929	0.972	0.898
		0.865	0.777	0.853	0.775
	0.3	0.971	0.873	**0.961**	**0.878**
		0.860	0.775	**0.857**	**0.788**
3	0.15	0.988	0.941	0.977	0.879
		0.839	0.756	0.850	0.774
	0.3	0.993	0.953	0.979	0.879
		0.848	0.748	0.842	0.765

Tabla 3.7: Tests realizados sobre los modelos recurrentes con todos los predictores salvo los UCS (En la parte de arriba de cada celda aparecen los resultados sobre D2 y debajo los de D4).



(a) Sobre D2

(b) Sobre D4

Figura 3.8: Matriz de confusión de red recurrente simple con todas las features.

excelente puntuación, llegando a una mAUC de 0,993 y un BCA de 0,953. Mientras que sobre D4 parece que es preferible un menor número de capas. Entre la versión simple o la LSTM, en D2 es preferible la simple y en D4 no hay un claro vencedor. Por último, como era esperable, no se generan diferencias significativas al variar el dropout. Percibimos curvas más lisas que antes. Si bien existe cierta distancia entre la curva de valoración y la de entrenamiento, no es comparable a la que se forma en un modelo sobreajustado (figura 3.9).

Lejos de lo esperado, la selección de features no ha supuesto una mejora en la precisión del método (tabla 3.10 y figura 3.11). Ahora parece que la LSTM funciona mejor sobre D2 y sigue sin haber un tipo de modelo que destaque en D4, de nuevo nos estamos moviendo entre puntuaciones muy similares y es complicado aseverar que combinación es la más apropiada. Se detecta una mejor precisión con 27 características, aunque aparecen excepciones.

Las gráficas (figura 3.12) adquieren la misma forma que sin selección (figura 3.9), a excepción de que la distancia en las curvas de pérdida es un tanto mayor. Sin embargo, sigue dentro de lo razonable y es signo de un aprendizaje adecuado.

En líneas generales, la puntuación alcanzada es ligeramente menor que la de los proyectos de redes neuronales recurrentes presentados en TADPOLE, que tratábamos de reproducir. Por ejemplo, CBIL obtuvo una mAUC de 0,897 y un BCA de 0,803, que le llevaron hasta la novena

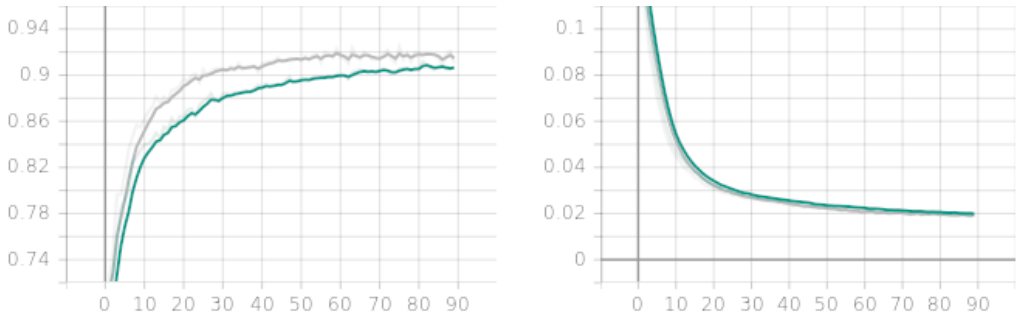


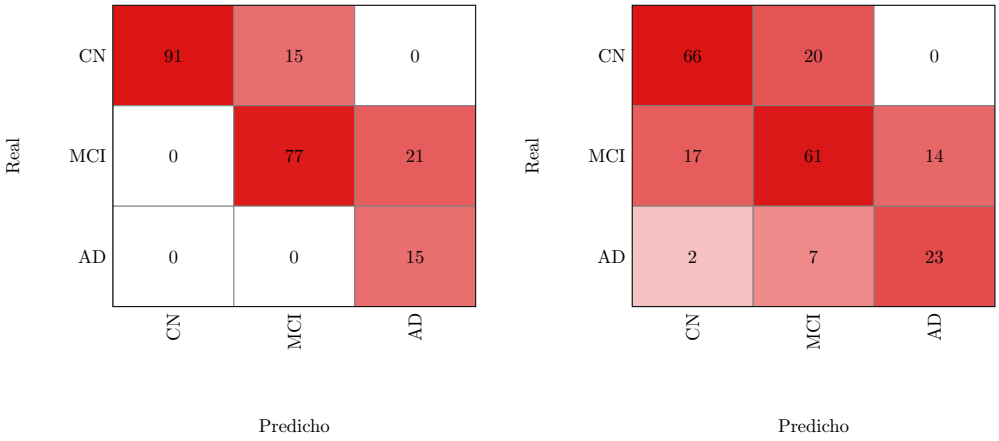
Figura 3.9: Curva de aprendizaje de la LSTM con todos los predictores. (Precisión por época (izq.) y pérdida por época (dcha.))

Selección de predictores			RNN simple		LSTM	
nº predictores	nº de capas	dropout	*mAUC*	*BCA*	*mAUC*	*BCA*
210	2	0.15	**0.949**	**0.834**	0.956	0.839
			0.856	**0.782**	0.852	0.758
		0.3	0.941	0.760	0.965	0.852
			0.862	0.733	0.860	0.760
	1	0.15	0.955	0.822	0.967	0.894
			0.855	0.749	0.855	0.778
		0.3	0.956	0.825	0.956	0.874
			0.865	0.754	0.866	0.784
	3	0.15	0.945	0.830	0.969	0.830
			0.847	0.752	0.842	0.710
		0.3	0.945	0.804	0.979	0.873
			0.854	0.760	0.847	0.758
27	2	0.15	0.951	0.857	0.975	0.895
			0.841	0.758	0.853	0.769
		0.3	0.962	0.841	**0.985**	**0.903**
			0.856	0.760	**0.861**	**0.781**
	1	0.15	0.965	0.887	0.977	0.908
			0.851	0.778	0.855	0.768
		0.3	0.963	0.866	0.976	0.908
			0.856	0.763	0.864	0.768
	3	0.15	0.960	0.851	0.991	0.905
			0.847	0.755	0.852	0.780
		0.3	0.955	0.835	0.989	0.901
			0.844	0.761	0.846	0.773

Tabla 3.10: Tests realizados sobre los modelos recurrentes después de haber seleccionado predictores (En la parte de arriba de cada celda aparecen los resultados sobre D2 y debajo los de D4).

posición. El código del grupo está disponible en TADPOLE share. Se trata de algoritmos en pytorch, posiblemente, al trabajar un nivel por debajo y, en consecuencia, poder desarrollar modelos de manera más flexible (y más compleja), pudieron adaptar y optimizar mejor sus redes al problema.

Por otro lado, BGU utilizó un modelo mixto de redes neuronales pre-alimentadas y LSTM. Al igual que nosotros escribieron el código en keras. Reportaron una mAUC más alta que nosotros,



(a) Sobre D2 (b) Sobre D4

Figura 3.11: Matriz de confusión de red LSTM después de haber seleccionado predictores.

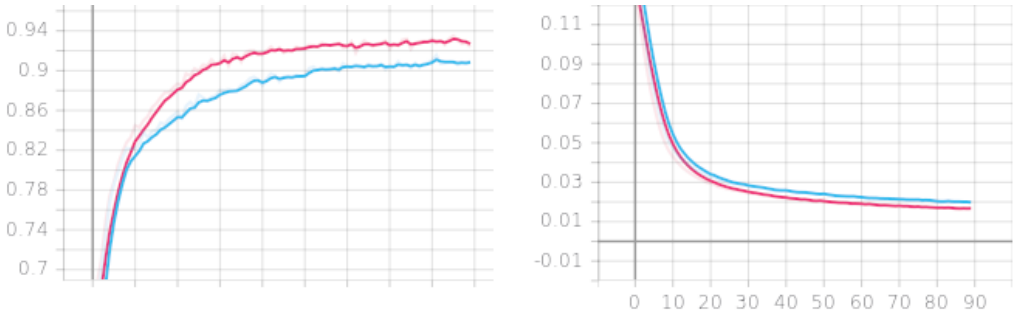


Figura 3.12: Curva de aprendizaje de la LSTM tras seleccionar predictores. (Precisión por época (izq.) y pérdida por época (dcha.))

0,883 frente nuestra 0,857. Sin embargo nuestro BCA es superior, 0,788 contra su 0,779. Se ha de tener en cuenta que BGU apostó por el aumento de datos. Creemos que con un aumento como el que realizaron ellos, o el que realizó mi compañero Francisco Ferraz en su TFG, podríamos haber cubierto esas décimas de mAUC que nos separan de sus resultados.

Métodos	resultados sobre D2		resultados sobre D4	
	mAUC	*BCA*	*mAUC*	*BCA*
Gradient Boosting de Frog	-	-	0.931	0.849
Random forest de Threedays	-	-	0.921	0.823
SVM de EMC-EB	-	-	0.907	0.805
DNN de SmallHeads	-	-	0.737	0.605
Shallow	**0.967**	**0.868**	**0.845**	**0.772**
DNN	**0.984**	**0.921**	**0.847**	**0.774**
LSTM de CBIL	-	-	0.897	0.803
LSTM de BGU	-	-	0.883	0.779
RNN simple	**0.976**	**0.892**	**0.860**	**0.782**
LSTM	**0.985**	**0.903**	**0.861**	**0.781**

Tabla 3.13: Resumen de los resultados obtenidos con los métodos desarrollados en TADPOLE challenge y en este TFG (en negrita).

En la tabla 3.13 figuran las mejores puntuaciones de cada tipo de modelo y las obtenidas por los algoritmos presentados en TADPOLE. En el primer grupo de filas están los tres métodos ganadores

del concurso, debajo las FNN y al final las redes recurrentes.

4. Interpretabilidad

Dentro del aprendizaje automático se puede establecer la dualidad entre los modelos de caja blanca (o explicables) y los de caja negra, en función de si los mecanismos de aprendizaje son interpretables por seres humanos [15]. A modo de ejemplo, los árboles de decisión o reglas de asociación tienen una representación visual que facilita la comprensión de por qué se ha tomado una cierta acción. Por el contrario, en una red neuronal las decisiones se llevan a cabo según múltiples operaciones matemáticas internas y complejas. Generalmente cuanto más complicado de interpretar es un algoritmo, más preciso es. En determinadas aplicaciones no es necesario detenerse a entender los razonamientos de la máquina. Ahora bien, este no es el caso de las prácticas biomédicas, en donde las deducciones deben estar fundamentadas en una sólida base científica.

Se entiende por inteligencia artificial explicable (Explainable Artificial Intelligence, XAI) a una serie de técnicas que permiten que los resultados de los algoritmos sean entendibles para los seres humanos. Esta cuestión es imprescindible en un trabajo como este, que resuelve un problema médico por medio de algoritmos de aprendizaje profundo. Se requiere un análisis que muestre la coherencia de los razonamientos llevados a cabo por las redes implementadas. Nos vamos a valer de la herramienta SHAP (SHapley Additive exPlanations) para realizar ese análisis. Esta aproximación está basada en teoría de juegos, más concretamente en el cálculo de valores de Shapley, que se da en los juegos cooperativos, en los que todos los jugadores luchan por el bien común. En el anexo B se explica de manera más detallada el funcionamiento a bajo nivel de los modelos de SHAP. Recomendamos una lectura previa de ese apéndice para comprender las matemáticas y teoría de juegos que hay detrás de la interpretabilidad y qué son exactamente los valores de Shapley. Únicamente recalcaremos la idea de que el Kernel Explainer de SHAP devuelve los valores de Shapley de cada predictor y con esa información se puede construir gráficas que nos indique el impacto de los atributos en las predicciones.

Antes de analizar ejemplos concretos, vamos a introducir unas nociones básicas de lectura de las gráficas. Los gráficos de violín se encargan de describir qué características son las más influyentes para la predicción de una única clase (CN, MCI, AD). Estas features se encuentran a la izquierda de la gráfica y están ordenadas por relevancia en la contribución. El eje de las x indica el valor de Shapley, aunque parezca contrario a lo expuesto en el anexo, pueden aparecer valores negativos porque no se tiene en cuenta la aportación de la esperanza φ_0. Para cada atributo se muestra su distribución en el eje de las y, de ahí que se formen "burbujas" en algunas zonas y otras se vean más llanas. Una mayor voluminosidad cuando se toma un determinado valor de Shapley indica que existen numerosas muestras a las que se les asocia esa contribución para en ese predictor en concreto. Aún queda otra dimensión, el color, que señala cómo de cerca está el valor de ese atributo del máximo.

Fijémonos en la figura 4.1, que corresponde al analisis de interpretabilidad sobre CN en un modelo de DNN con todos los predictores. Ahí el atributo más importante es el diagnóstico, seguido de DX_bl. Eso lo averiguamos al prestar atención al orden en el que aparecen las características a la izquierda. Ahora miremos únicamente la fila del diagnóstico, encontraremos una "burbuja" morada a la izquierda. La parte más gruesa de esa "burbuja" está en torno a -0.2, un poco menos. Eso significa que hay bastantes muestras que toman ese valor de Shapley en el diagnóstico. Al tratarse de una puntuación negativa, podemos inferir que esos pacientes no son clasificados como CN. En el extremo, pasado -0,4, la burbuja se aplana, eso quiere decir que los datos que alcanzan ese valor son un grupo reducido. En lo que se refiere al color, vemos que la parte izquierda es más morada y la de la derecha más azul, es decir, los pacientes con un Diagnóstico alto tienen un valor de Shapley negativo y los que lo tienen bajo al revés. Esto quiere decir que para la clase CN valores bajos de diagnóstico favorecen la predicción de la clase CN y valores altos no. Esto es coherente con el conocimiento del problema, pues si un paciente ha sido diagnosticado con AD (valor 2) es muy difícil que su evolución retroceda a CN (valor 0).

Un gráfico de barras es más sencillo de interpretar, las características también se presentan a

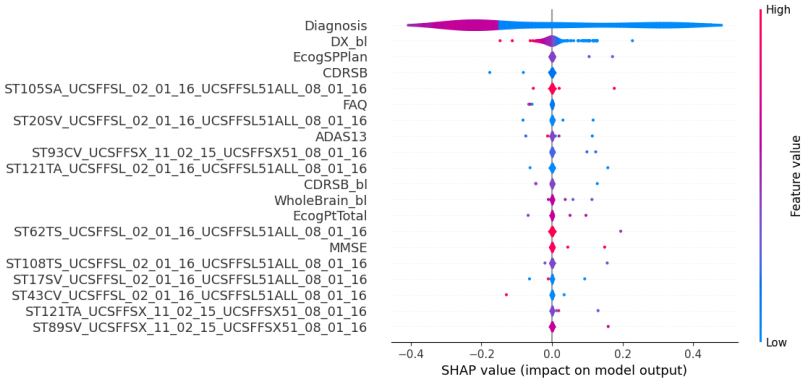


Figura 4.1: Ejemplo de gráfico de violin, corresponde a la interpretabilidad sobre CN en un modelo de DNN con todos los predictores.

mano izquierda ordenadas por trascendencia. En el eje de las x se encuentra el valor de Shapley medio y las barras no son más que la suma de las medias para cada predicción.

En los modelos estáticos afortunadamente no fue necesario realizar cambios significativos en la pipeline para adaptar hacer funcionar el kernel explainer. Debemos recordar que esta clase de SHAP es agnóstica al modelo, sólo necesita una función f para elaborar el análisis. En este caso concreto la función utilizada es la de predicción de nuestra red ya entrenada.

En contraste con lo anterior, a la hora de poner en práctica el kernel explainer en los modelos recurrentes, nos encontramos con un inconveniente considerable. Esa visión de un programa agnóstico al modelo, capaz de explicar cualquier tipo de algoritmo de aprendizaje automático con sólo pasar como argumento la función de predicción y de ofrecernos un analisis de interpretabilidad bastante profundo y rápido siguiendo principios de teoría de juegos, se rompe cuando uno se percata de este impedimento. Resulta que el explainer está pensado para que el input sea un dataset cuyas muestras las conformen vectores con un número fijo de características. En cambio, nuestro modelo de red neuronal recurrente necesitaba un número fijo de marcas temporales por paciente y cada marca temporal con la asignación de características correspondiente en ese momento, en otros términos más precisos, es un modelo multivariable con marcas temporales. En síntesis:

Forma del input de SHAP: (nº muestras, nº features)

Forma del input de nuestra red: (nº muestras, marcas temporales, nº features)

Una posible solución para ajustar la forma de nuestro input a la aceptada por SHAP, que se nos puede ocurrir a priori, sería definir cada característica en un momento concreto como un tipo de predictor nuevo. Así, por ejemplo, un paciente tendría como características el estado clínico del primer mes, del segundo, etc y así para todos los predictores. Lo que ocurre es que si antes teníamos 1800 biomarcadores, ahora tenemos 216.000 y el explainer de kernel no es un algoritmo que se caracterice precisamente por su velocidad ni por su escalabilidad.

Por otra parte, debemos de ser conscientes del tipo de inputs que tenemos. Como se ha dicho, son 120 marcas temporales pero los pacientes no van todos los meses a realizarse pruebas, de ahí que el tensor de input sea cuasivacío (sparse). En D4 directamente tenemos un solo registro por paciente, en vez de una secuencia entera. Por tanto, sabemos de antemano que la amplia mayoría de predictores no intervienen en absoluto en la predicción final, porque son valores de relleno a los que se le aplica la máscara.

André Ferreira en su artículo [8] en el que trata bastante bien este problema propone una

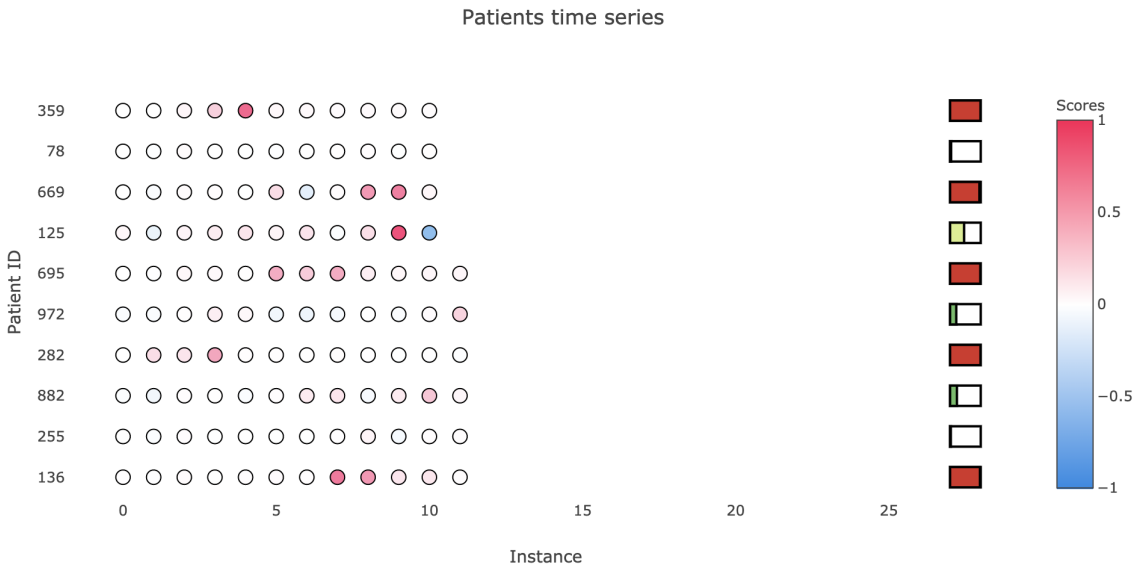


Figura 4.2: Ejemplo de visualización de importancia en modelos multivariable con marcas temporales. (Imagen extraida del artículo de André Ferreira [8])

modificación del código fuente de SHAP para adaptar el explainer a su RNN en torch. No obstante, el objetivo de programar un parche ad hoc para el problema ha quedado fuera del alcance de este TFG.

En un escenario ideal, antes de nada, el conjunto de test lo conformarían datos secuenciales. Del mismo modo, SHAP contaría con algún tipo de explainer válido para modelo multivariable con marcas temporales, que a parte de decirnos qué features son las más importantes, nos indicaría en qué mes adquieren ese protagonismo, algo parecido al esquema de la figura 4.2.

A pesar de que SHAP es una librería relativamente reciente, de 2018, y todavía tiene pendiente dar soporte a modelos más complejos; se podría hacer una análisis de este problema en concreto gracias a que el conjunto de test lo componen datos aislados. El planteamiento es el siguiente: pasarle al explainer los datos de test y una función que cambie la forma para poder aplicar el modelo y obtener las predicciones. Con este cambio en la pipeline, ya estamos preparados para conocer la importancia de cada característica.

En el anexo C pueden encontrar el análisis de interpretabilidad completo para ambas clases de modelos. Adelantamos que se comenzó ejecutando SHAP con todas las características y observamos que la de mayor relevancia era el diagnóstico, tanto en RNN como en FNN. Entonces se probó a extraer esa información, con el propósito de corroborar que las redes siguieran manteniendo la coherencia. Sin duda las pruebas fueron un éxito, en el momento en el que se desconoce el estado clínico, la red se apoya en los biomarcadores utilizados para elaborarlo. Decidimos ir un paso más allá y deshacernos también de esas variables utilizadas en ADNI en el diagnóstico, que son los test cognitivos CDRSB y MMSE. Las redes supieron encajar bien esa carencia y le otorgaron el peso a otros tipos de test cognitivos. Cuando ya no dejamos ni los test cognitivos la precisión del modelo bajó considerablemente y los resultados de la interpretabilidad perdieron su solidez.

Las explicaciones que nos ofrece SHAP nos pueden servir para obtener una visión rápida y precisa de las características más influyentes en el proceso de clasificación. Sin embargo, hemos decidido adentrarnos en las configuraciones internas de las redes neuronales para complementar esta investigación. Por un lado, se podrá revisar los pesos que le asigna la red a cada característica en la primera capa y hacer una lectura similar a la de SHAP, aunque, por supuesto, más simplista. Por otro lado, podremos buscar una respuesta a la pregunta de por qué las redes neuronales no nos ofrecen tan buenas prestaciones como otros modelos clásicos de aprendizaje automático. En

el apéndice D podrán encontrar una exposición detallada de los resultados. Advertimos que una gran cantidad de neuronas tienen pesos muy próximos a cero. Tal vez, ese sea el motivo por el que aunque aumentemos la complejidad del modelo, no se ve acompañado de una mejora de los resultados. Eso explicaría por qué no hay tanta diferencia entre la red de una capa y la de cuatro, porque las capas de más tan apenas pueden contribuir al resultado final.

5. Conclusiones

El fin último de este trabajo era reproducir los métodos de deep learning presentados en TADPOLE challenge y contribuir a la interpretabilidad. Gracias al concurso se reunieron distintos métodos de aprendizaje automático, algunos de un excelente rendimiento. Ahora bien, no basta con quedarnos con la idea de que se puede aplicar el machine learning al diagnóstico de la enfermedad de Alzheimer y conseguir predicciones precisas. Deberíamos ir un paso más allá y llevar estas técnicas a terreno práctico. Así que el primer paso es demostrar que cualquiera puede tener acceso a esta tecnología. Al menos hemos podido averiguar que en redes neuronales es posible, que una vez que disponemos de los datos podemos reproducir los experimentos con keras y obtener las mismas puntuaciones, e incluso superarlas.

En la tabla 3.13 encontramos un resumen de los resultados obtenidos en TADPOLE challenge (aunque no contamos con la puntuación en el problema de clasificación sobre D2) y en este trabajo. En las primeras filas se muestra el rendimiento de los métodos ganadores, distintos tipos de técnicas clásicas de aprendizaje automático. Por extraño que parezca, a menudo sucede que un modelo simple supera a uno más complejo. Los algoritmos más elaborados sufren en mayor medida el sobreajuste y el problema de la dimensión. Sin ir más lejos, la red LSTM y algunas versiones de RNN desarrolladas en este trabajo producían *overfitting*. Por supuesto, esto no implica que debamos abandonar la idea de utilizar deep learning para este problema. En el análisis de características (apéndice C) y en el de pesos (apéndice D) se concluye que los predictores más influyentes, por detras del diagnostico, son los test cognitivos, en tanto que los biomarcadores de imagen no contribuyen a la predicción. En general esas características son valores numéricos que aglutinan demasiada información extraída de una imagen. Tal vez, de cara a un futuro trabajo, se podría hacer un *ensamble* de modelos, en el que las imágenes de ADNI se traten con redes neuronales convolucionales recurrentes (CRNN, figura 5.1) y el resto de información con alguna red recurrente, debido a que hemos comprobado que los modelos recurrentes dan mejor resultado que los estáticos (tabla 3.13).

Siguiendo con la lectura del tabla de resultados, vemos como no hay tan apenas diferencias entre el esquema poco profundo y la DNN. En el apéndice D vemos como al aumentar el número de capas, las neuronas de las capas ocultas asignan pesos iguales, o muy cercanos, a cero, de manera que aunque se añadan más capas, estas tan apenas participan en el resultado final. No obstante parece que a mayor número de capas, mayor puntuación sobre D2. Esto podría deberse a que el número de *transformers* de D2 es del 5 % y en D4 es del 25 %. Entonces es lógico que sobre D4 saque peores métricas. Si entramos a comparar nuestros modelos de FNN con el presentado en TADPOLE, percibimos una diferencia abismal. Desconocemos los motivos por los que SmallHeads consiguió una puntuación tan inferior. Quizá sea porque utilizaron una arquitectura un tanto enrevesada o porque no añadieron alguna característica de gran relevancia. En redes neuronales recurrentes ocurre algo análogo, la RNN y la LSTM alcanzan puntuaciones muy parejas, aquí tanto para D2 como para D4. Nuestras puntuaciones sí que guardan cierta similaridad con las que obtuvo BGU. Sin embargo, CBIL nos sobrepasa a los dos. Creemos que puede deberse a que trabajan a un nivel por debajo de nosotros, utilizan pytorch en vez de tensorflow, y eso les da más flexibilidad a la hora de implementar modelos.

El siguiente descubrimiento, ya en materia de interpretabilidad, es que las redes neuronales no sólo garantizan una puntuación adecuada sino que además son coherentes en las decisiones tomadas. Se apoyan principalmente en el diagnóstico actual, que a su vez está basado en los test cognitivos CDRSB y MMSE. Llama la atención que al retirar los features de naturaleza cognitiva los modelos pierden su capacidad de acierto.

Con respecto a trabajos futuros, se podría utilizar esta misma tecnología para la resolución de los otros problemas planteados en el challenge: la predicción de la puntuación del test cognitivo ADAS-Cog y la del volumen de ventrículos. Reconocemos que en ese terreno aún queda bastante por investigar, ya que en la predicción de la puntuación de ADAS-Cog en el challenge fue sutilmente

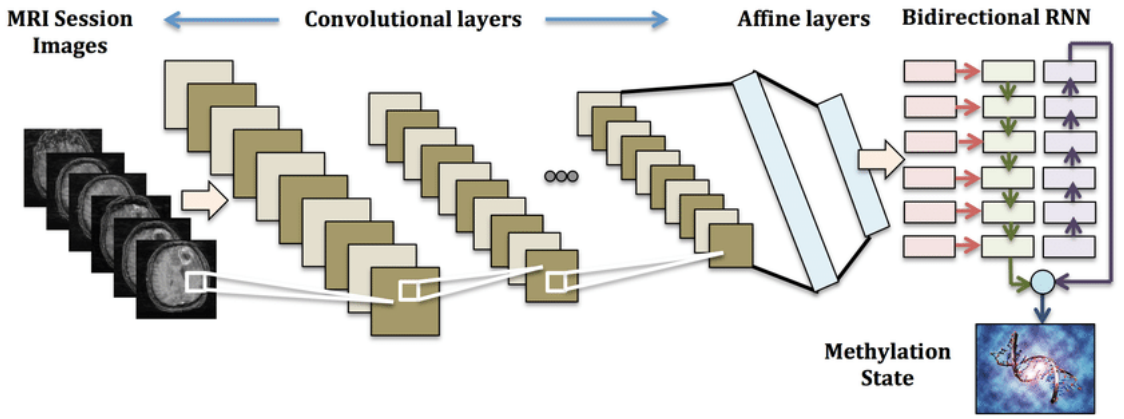


Figura 5.1: Ejemplo de arquitectura de red neuronal convolucional recurrente, extraido del paper de Lichy Han y Maulik R. Kamda [10].

superior a una asignación aleatoria. Si deseamos que los modelos de aprendizaje profundo adelanten al resto, es imprescindible aumentar la cantidad de datos. Hemos visto que en métodos complejos como la LSTM, se llega a producir sobreajuste, con un mayor volumen de información esto no sucedería. Asimismo, los modelos recurrentes ofrecen una mejor respuesta que los estáticos debido a la naturaleza temporal del problema. Sin embargo, no disponemos de demasiada información secuencial. Si contáramos con más datos por paciente se mejoraría la predicción de su evolución y podríamos aplicar técnicas cada vez más complejas, como aquellas que hacen uso de trasferencia de aprendizaje.

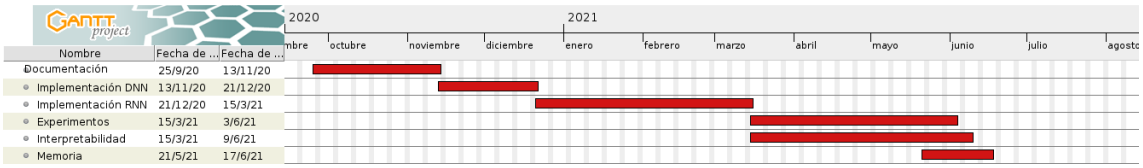


Figura 5.2: Diagrama de gantt del proyecto.

En la figura 5.2 se muestra el diagrama de GANTT de este proyecto. Como es natural no se mantuvo una misma intensidad en todos los tramos. En el periodo de documentación podemos diferenciar entre una formación básica en deep learning [1] [19] y una más específica sobre TADPOLE challenge. Desde una perspectiva personal, reconozco que este proyecto ha contribuido considerablemente a mi formación en materia de aprendizaje profundo. Por primera vez he salido de los ejemplos de los tutoriales para enfrentarme a un problema real con todo lo que eso conlleva. Bajo mi punto de vista, es de gran interés que todos los modelos aprendidos sean extrapolables a otros tipos de problemas.

A. Fundamentos del aprendizaje profundo

A.1. Arquitectura de red

Las redes neuronales están compuestas por capas de neuronas. Toda red está formada, como mínimo, por una capa de entrada (input), que se encarga de leer los datos, y una de salida. En nuestro problema, la última capa, la de salida, devolverá un vector de tamaño tres que contiene las posibilidades de pertenecer a cada una de las clases consideradas (CN, MCI, AD). La predicción del diagnóstico será aquella que la red determine como más probable. Entre el input y el output podemos encontrar todo tipo de capas ocultas. El número de capas ocultas divide estos métodos entre entre redes poco profundas (shallow) y redes profundas (deep). En la figura A.1 se muestra un ejemplo ilustrado y simplificado de la arquitectura de una red neuronal.

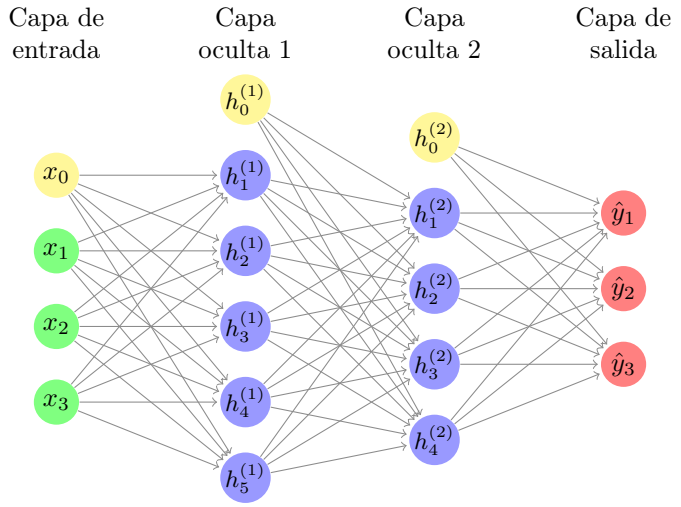


Figura A.1: Representación de una red neuronal prealimentada.

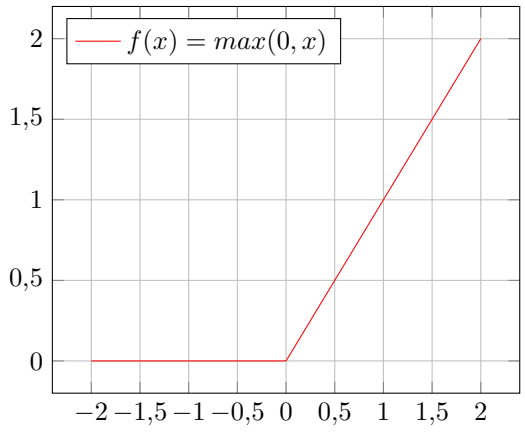
Existe un tipo de capa conocida como densa que consta de un número arbitrario de neuronas. La operación que realiza internamente con el fin de obtener el valor de salida es la siguiente:

$$f(x) = k(\sum_{i=0}^{n} w_i x_i + b) \tag{A.1}$$

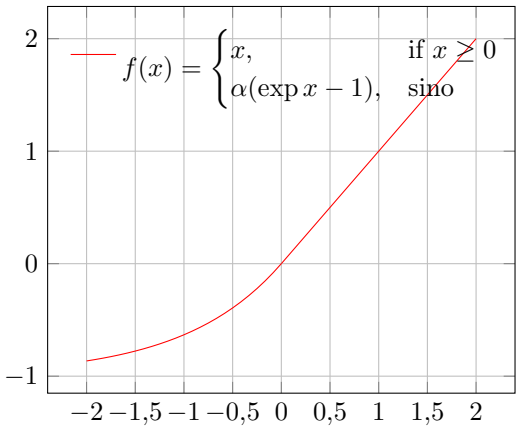
Donde x es un vector de tamaño n que contiene todos los datos de entrada, w es un vector de pesos, que van ajustándose en cada iteración del entrenamiento, y b es el sesgo que se le añade (en la figura A.1 correspondería a aquellos nodos coloreados de amarillo). La suma ponderada no deja de ser una regresión lineal al uso. Se puede demostrar matemáticamente que la concatenación de múltiples regresiones lineales es equivalente a haber calculado sólo una. De modo que no tendría sentido colocar las neuronas secuencialmente, a no ser que se distorsione la linealidad de alguna forma. Es entonces cuando entra en juego la función k, llamada de activación, que añade transformaciones no lineales y además es fácil de derivar. El tipo de transformación depende de la función utilizada, en la figura A.2 se presentan distintos ejemplos.

En este trabajo, las neuronas de las capas ocultas hacen uso del rectificador o función de activación de unidad lineal rectificada (Rectified Linear Unit, RELU). Sin embargo, en la capa de salida, al tratarse de una clasificación multiclase, utilizaremos la función softmax, la función exponencial normalizada, cuya definición es la siguiente:

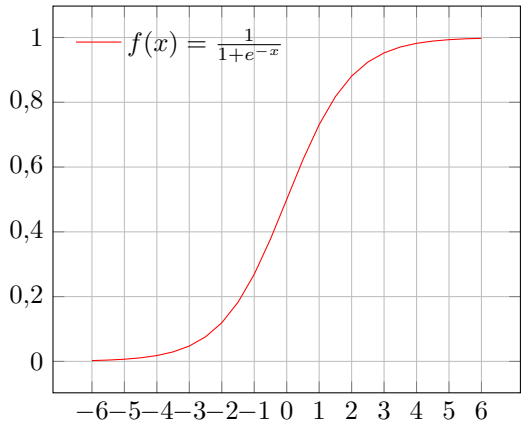
$$\begin{aligned} f(x)&\colon \mathbb{R}^K \to [0,1]^K \\ f(x)_j &= \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \forall j = 1...K \end{aligned} \tag{A.2}$$



(a) fución RELU.



(b) fución ELU.



(c) función sigmoide.

Figura A.2: Ejemplos de funciones de activación.

Como se puede apreciar el dominio y el rango tienen la misma dimensión, K, que en este proyecto tomará el valor de tres, el número de estados clínicos posibles.

A.2. Entrenamiento del modelo

Con anterioridad se ha mencionado que los pesos y el sesgo se deben ajustar hasta dar con el modelo deseado. Como bien sugiere el término aprendizaje automático, la idea no es que nosotros asignemos los parámetros a la red, sino que esta misma encuentre la mejor aproximación a partir de un conjunto de datos de entrenamiento. Sabemos que al aplicar otras técnicas de machine learning, con derivar la función de coste podemos averiguar los parámetros por medio de la técnica de mínimos cuadrados. No obstante, pese a que todavía no se ha mostrado la función de coste, adelantamos que no se trata de una función convexa; dicho de otra manera, cuando la derivada es cero puede darse la situación de que no sea un mínimo global, puede ser un mínimo local, un máximo, un punto de inflexión o un punto de silla, en suma, todo aquello que se conoce en matemáticas como un punto crítico.

Por consiguiente, las redes neuronales necesitan un algoritmo más complejo que mínimos cuadrados, ha de entrar en escena el descenso del gradiente. Dado un esquema de n parámetros y una función de pérdida f, debemos calcular las derivadas parciales de cada parámetro en nuestro punto actual. El vector de derivadas se conoce como gradiente ($\nabla f(x)$). Al decrementar ese valor al punto actual y repetir este proceso una serie de veces, nos acercaremos a un mínimo local (figura A.3).Podría resumirse el algoritmo de esta manera:

function GRADIENTDESCENT
  for $i \leftarrow 0$; $i < EPOCHS$; $i++$ **do**
    $\theta \leftarrow \theta - a\nabla f(x)$
  end for
end function

Apreciamos la aparición de dos hiperparametros, el número de épocas o iteraciones y el factor que multiplica el gradiente, conocido como tasa de aprendizaje. La eficiencia y el éxito de nuestro algoritmo depende de una precisa calibración de estas variables. Una tasa muy grande impediría acercarnos al mínimo, mientras que una muy pequeña haría que avancemos muy lentamente, así que sería necesario un mayor número de iteraciones y, en definitiva, sería un algoritmo ineficaz.

Nuestro modelo, específicamente, es entrenado con un ratio de aprendizaje de 0.001 y ejecuta 250 iteraciones. Quizás en algunos ejemplos pueden parecer demasiadas, afortunadamente jugamos con la baza de que el algoritmo es de rápido procesamiento.

Únicamente queda un detalle por concretar, la técnica de descenso del gradiente precisa, indiscutiblemente, del gradiente. El error de cada salida se calcula hacia atrás con retropropagación (backpropagation), se trata de un método muy eficiente, ya que con una sola pasada obtienes el vector con las variaciones.

Dada una arquitectura de L capas, comenzaremos por el gradiente de la última. Somos conscientes de que los parámetros de la una neurona son el peso y el sesgo, w y b respectivamente, así que, para una función de pérdida C, debemos calcular estas dos derivadas parciales:

$$\frac{\partial C}{\partial w^L}, \frac{\partial C}{\partial b^L} \tag{A.3}$$

En la capa final solo hay que derivar la siguiente composición de funciones (recuerden que k es la función de activación):

$$C(k^L(Z^L)), \texttt{ con } Z^L = w^L k^{L-1} + b^L \tag{A.4}$$

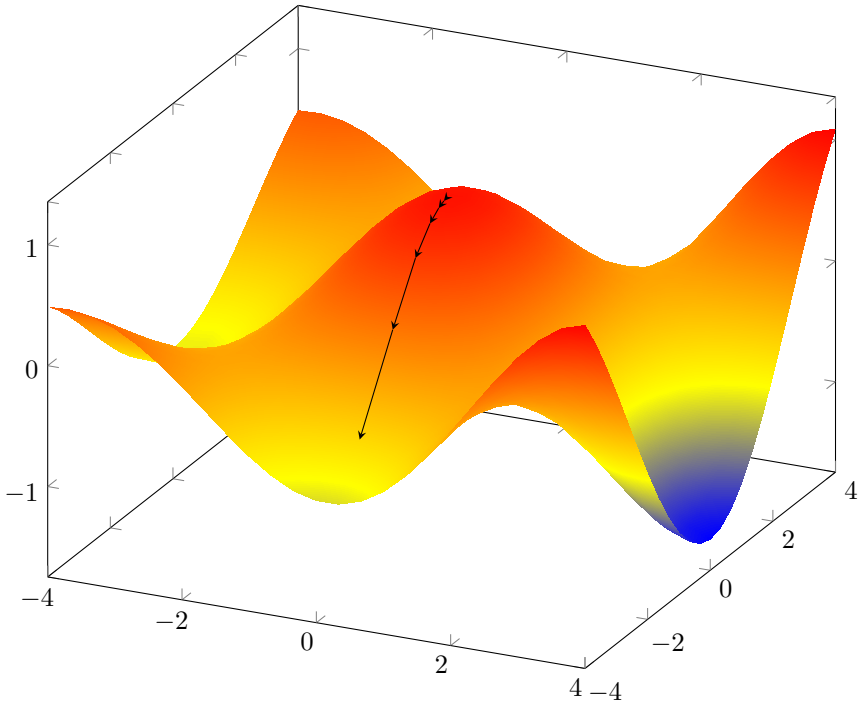


Figura A.3: Ilustración del descenso del gradiente. (En aras de la simplicidad, la representación cuenta con dos parámetros y en el eje z subyace el valor de la función de coste).

Llegados a este punto conviene indicar, conforme a la naturaleza de nuestro problema: un clasificador de clase múltiple y etiqueta única, lo aconsejable es que nuestra función de coste sea la entropía cruzada categórica (Categorical Cross Entropy):

$$C = -\sum_{i=0}^{N_{cat}} t_i \log(k^L(Z^L)) \tag{A.5}$$

Utilizando la regla de la cadena, podemos establecer las siguientes igualdades:

$$\begin{aligned} \frac{\partial C}{\partial w^L} &= \frac{\partial C}{\partial k^L} \cdot \frac{\partial k^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial w^L} \\ \frac{\partial C}{\partial b^L} &= \frac{\partial C}{\partial k^L} \cdot \frac{\partial k^L}{\partial z^L} \cdot \frac{\partial z^L}{\partial b^L} \end{aligned} \tag{A.6}$$

$\frac{\partial C}{\partial z^L} = \frac{\partial C}{\partial k^L}\frac{\partial k^L}{\partial z^L}$ es el error imputado a la neurona. A mayor valor, más responsabilidad habrá tenido esa neurona en concreto en el fallo de la ejecución. Se suele representar así δ^L.

A primera vista parece que se ha complicado el problema. En cambio ahora las derivadas son mucho más fáciles de resolver:

$$\frac{\partial z^L}{\partial b^L} = 1, \frac{\partial z^L}{\partial w^L} = k_i^L \tag{A.7}$$

Si k^L es softmax:

$$\begin{aligned} \texttt{si } i = j: \frac{\partial k_i^L}{\partial z_i^L} &= \frac{\partial \frac{e^{z_i^L}}{\sum_C}}{\partial z_i^L} = \frac{e^{z_i^L}\sum_C - e^{2z_i^L}}{\sum_C^2} = \frac{e^{z_i^L}}{\sum_C}(1 - \frac{e^{z_i^L}}{\sum_C}) = y_i(1-y_i) \\ \texttt{si } i \neq j: \frac{\partial k_i^L}{\partial z_j^L} &= \frac{\partial \frac{e^{z_i^L}}{\sum_C}}{\partial z_j^L} = \frac{0 - e^{z_i^L}e^{z_j^L}}{\sum_C^2} = -y_i y_j \end{aligned} \tag{A.8}$$

Si es RELU, será igual 0 para los negativos y 1 para los positivos y no estará definida en x=0.

Finalmente, podemos calcular δ^L suponiendo que la función de activación utilizada sea softmax y la de coste la entropía cruzada categórica, con el apoyo de las ecuaciones anteriores.

$$\begin{aligned}\frac{\partial C}{\partial z_i^L} &= -\sum_{j=0}^{N_{cat}} \frac{\partial t_j \log(y_j))}{\partial z_i^L} = -\sum_{j=0}^{N_{cat}} \frac{t_j}{y_j}\frac{y_i}{\partial z_i} = -\frac{t_i}{y_i}\frac{\partial y_i}{\partial z_i} - \sum_{j\neq i}^{N_{cat}} \frac{t_j}{y_j}\frac{\partial y_j}{\partial z_i} \\ &= -\frac{t_i}{y_i} y_i(1-y_i) - \sum_{j\neq i}^{N_{cat}} \frac{t_j}{y_j}(-y_j y_i) = -t_i + \sum_{j=0}^{N_{cat}} t_j y_i = -t_i + y_i \sum_{j=0}^{N_{cat}} t_j \\ &= y_i - t_i\end{aligned} \tag{A.9}$$

Después de todos estos cálculos, ya hemos sacado las ecuaciones de la capa L, tenemos pendiente hacer lo mismo para el resto de capas. Lo ventajoso de la backpropagation es que conociendo δ^L, se puede inferir δ^l para cualquier capa únicamente aplicando esta fórmula recursiva:

$$\delta^{l-1} = W^l \cdot \delta^l \cdot \frac{\partial k^{l-1}}{\partial Z^{l-1}} \tag{A.10}$$

A.3. Prevención de sobreajuste

Aunque la minimización del error es fundamental en el proceso de aprendizaje, si nos excedemos en la reducción, corremos el riesgo de caer en sobreajuste, es decir, que el modelo desempeñe su labor con excelencia en el conjunto de entrenamiento, pero que su actuación sea mediocre con otros datos.

Una posible manera de eludir el problema estriba en no utilizar todos los datos para el entrenamiento, sino hacer una división del conjunto en dos partes. Nosotros tenemos un 80 % de la información dedicada al entrenamiento y el otro 20 % a la validación. Naturalmente, la selección de datos de cada conjunto se ha realizado de manera aleatoria. Es una buena práctica porque en ocasiones los elementos del dataset se encuentran ordenados.

Durante el entrenamiento se evalúa el grado de generalización del modelo a través del conjunto de validación (figura A.4). Es lógico que el error de validación sea siempre ligeramente más alto, como se muestra en la gráfica. El sobreajuste se produce a partir de la mitad del diagrama de la figura A.4, cuando la pérdida no deja de crecer.

Otro elemento a tener en cuenta es el dropout (traducido a veces como abandono o disolución). Su misión consiste en desactivar neuronas de manera aleatoria, en otras palabras, hace que el input sea cero. Esta medida viene definida por el ratio de neuronas que anula. Si es 0, ninguna se verá afectada. Si es 1, todas. En los experimentos, se probó a variar la tasa entre 0,15 y 0,3.

En última instancia se añade un término de regularización. Se ha probado con dos tipos de regularizadores y se han ajustado el valor de sendas normas. Por un lado, en los primeros ensayos, se añadía el regularizador LASSO (least absolute shrinkage and selection operator) o norma L1 a la función de coste.

$$\hat{C}(X) = C(x) + \alpha \left\|W\right\|_1 = C(x) + \alpha \sum_i^n |w_i| \tag{A.11}$$

Por otro lado, también se hicieron pruebas con la norma L2 o Ridge. En la sección 3 se muestran todos los tests y se termina concluyendo cuál es el regularizador más adecuado y el valor de alfa indicado para cada modelo para cada modelo.

$$\hat{C}(X) = C(x) + \alpha \left\|W\right\|_2^2 = C(x) + \alpha \sum_i^n w_i^2 \tag{A.12}$$

El concepto de normalización de lotes no ha sido introducido hasta ahora. Es una manera de aumentar el aprendizaje sin caer en sobreajuste [14]. Comienza estandarizando la salida de la capa

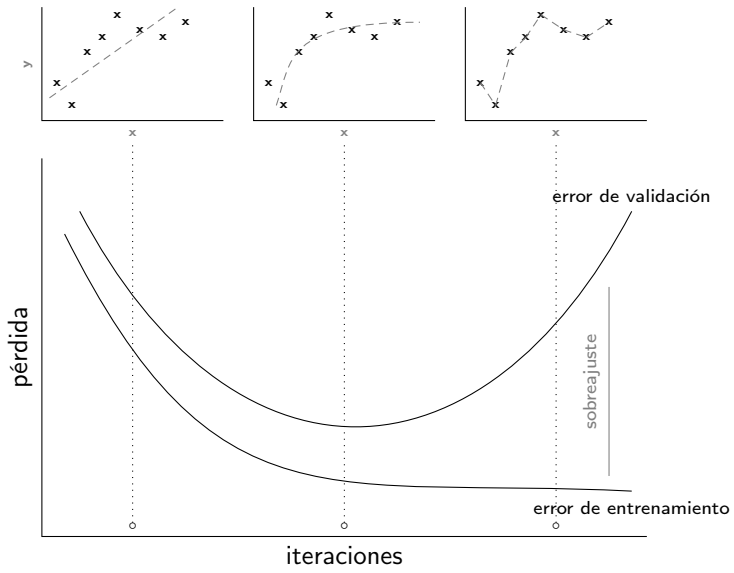


Figura A.4: Evolución del error durante el entrenamiento. Se observan casos de subajuste (izq.), ajuste adecuado (centro) y sobreajuste (dcha.)

anterior.

$$\hat{x} = \frac{x - \mu}{\sigma} = \frac{x - \frac{1}{N}\sum_{i=1}^{N} x_i}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}} \qquad \text{(2.4 duplicada)}$$

El resultado de servirnos de esta capa es el siguiente. Donde γ y β son parámetros que asimila la red durante su entrenamiento.

$$y = \gamma \cdot \hat{x} + \beta \qquad \text{(A.13)}$$

B. Fundamentos de SHAP

En juego cooperativo una coalición de jugadores busca justicia a la hora de adjudicar sus ganancias. Partimos de la base de que no todos han contribuido de la misma manera en la generación de la riqueza y entonces se tienen que tener en cuenta los aportes individuales a la hora de repartir.

Examinemos brevemente la definición de valores de Shapley. Debemos comenzar con la función característica v, que dada una coalición devuelve el valor que engendra dicha coalición. Sea la contribución marginal el beneficio (o pérdida) que aporta un miembro a la coalición. Que la podemos designar de la siguiente manera:

$$v(S \cup \{i\}) - v(S)$$

De acuerdo con el principio de justicia, cada jugador exigirá el promedio de sus contribuciones, es decir, valor de Shapley de i es:

$$\varphi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!}(v(S \cup \{i\}) - v(S)) \tag{B.1}$$

Donde n es la cardinalidad del conjunto y S una de las posibles coaliciones a las que no pertenece i. Vean como el promedio queda como las posibles permutaciones que se pueden formar dentro de la coalición partido por el número total de posibles permutaciones.

Esta distribución se considera ecuánime debido a que se satisfacen las siguientes propiedades:

1. **Eficiencia:** la suma de las contribuciones individuales es equivalente a la ganancia total $\sum_{i \in N} \varphi_i(v) = v(N)$.
2. **Anonimato:** si dos jugadores aportan lo mismo al unirse a la coalición, se les debe asignar la misma contribución. En suma, el etiquetado no debe influir.
3. **Linealidad:** en los juegos en los que haya más de una ganancia se debe cumplir que: $\varphi_i(v + w) = \varphi_i(v) + \varphi_i(w)$
4. **Jugador Nulo:** $v(S \cup \{i\}) = v(S), \forall S \implies \varphi_i(v) = 0$

En XAI cada predictor se comporta como un jugador y φ_i es la forma en la que ha colaborado en el resultado final. SHAP, más concretamente el Kernel Explainer [18] [16], trata de explicar la puntuación como la suma de las contribuciones marginales de variables mediante de un modelo lineal $g(z)$

$$g(z) = \varphi_0 + \sum_{j=1}^{M} \varphi_j z_j \tag{B.2}$$

Donde φ_0 es la esperanza del output sobre los datos de entrenamiento, z es un vector binario que representa la membresía de cada variable a la coalición (z_i vale 1 si la i-esima variable pertenece a la coalición, 0 en caso contrario), φ_j es el valor de Shapley de característica j-esima.

El cálculo de los valores de Shapley se obtienen midiendo la puntuación de distintas coaliciones. Sin embargo el modelo ha sido entrenado con todos los predictores, no podemos eliminar los que queramos a nuestro antojo. Del mismo modo no basta con asignar un valor especifico a esas variables, como NaN o cero. La solución pasa por definir una función $h_x(z') = z, h_x : \{0,1\}^M \to R^p$, que para los elementos con $z'_i = 1$ devuelva el valor correspondiente de esa variable en la instancia x y para el resto un valor que tome ese mismo atributo en otra instancia. Dentro de una misma coalición tendríamos distintos registros, donde los miembros de esta se mantendrían constantes y el resto de features variarían. De ese modo reduciríamos la dependencia de las variables ajenas a la coalición (figura B.1).

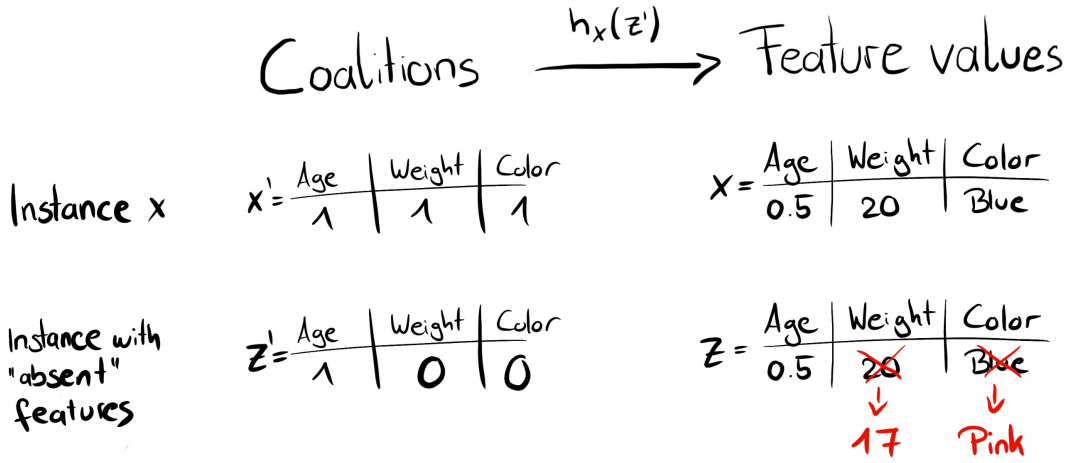


Figura B.1: Explicación grafica del mapeo de h_x [18]

Las instancias del modelo lineal se terminan ponderando por el siguiente kernel:

$$\pi_x(z') = \frac{(M-1}{\binom{M}{|z'|}|z'|(M-|z'|)} \tag{B.3}$$

Donde M es el número total de características y $|z'|$ la cantidad de miembros de la coalición. La finalidad es otorgarle más peso a las coaliciones con cardinalidad más cercana a 1 o $M-1$, ya que son las que mejor explican el comportamiento de una variable en concreto. Si la coalición consiste en justo la mitad de features posibles, aprendemos muy poco del impacto individual de cada una.

Ahora solo quedaría realizar una regresión lineal ponderada a nuestro modelo y averiguaremos la influencia que tiene cada variable en la predicción final. En otras palabras, habría que optimizar la siguiente función.

$$L(f, g, \pi_x) = \sum_{z' \in Z} [\, f(h_x(z')) - g(z')]^{\,2} \pi_x(z') \tag{B.4}$$

Como habrán podido apreciar, el coste temporal del algoritmo trasciende a niveles exponenciales, dado que existen un total de 2^M combinaciones de coaliciones. Si bien es cierto que los pesos de la mayoría de combinaciones son tan pequeños que la omisión de esos cálculos no tiene ninguna repercusión en el resultado final, continúa siendo un modelo poco eficiente y en la práctica ha conllevado alguna que otra traba.

Otra apreciación es que se trata de un explainer agnóstico al modelo, dicho de otra forma, en teoría se le puede pasar cualquier tipo de función. Existen explainer implementados y optimizados expresamente para técnicas concretas, como es el caso del Tree Explainer que es compatible con la mayoría modelos basados en árboles de scikit-learn. De manera análoga, hay otro para redes neuronales, que desgraciadamente no es compatible con nuestros algoritmos.

C. Análisis de interpretabilidad

Después de haber concedido tanta importancia a la interpretabilidad en los capítulos principales de la memoria, en especial en el cuarto, y de haber comentado el funcionamiento de SHAP en el anexo B, es turno de poner a prueba esta herramienta sobre nuestro sistema y realizar un exhaustivo análisis de la interpretabilidad.

Dado que contamos con dos categorías de modelos bien diferenciadas, que no tienen por qué comportarse igual a la hora de llevar a cabo sus predicciones, vamos a dividir esta sección en dos partes, basadas en los análisis de las dos arquitecturas de redes neuronales. Ambos empiezan con una ejecución del explainer sobre un algoritmo entrenado con todas las características. A medida que se desarrollan los experimentos, se retiran predictores y vamos llegando a una serie de conclusiones.

C.1. Interpretabilidad en modelos estáticos

En relación con los pasos previos a los ensayos, recordar que se utilizan un explainer de kernel, que sirve para cualquier tipo de técnica de aprendizaje automático, y que solo necesitamos pasar como parámetro la función de predicción de la red neuronal después de la etapa de aprendizaje.

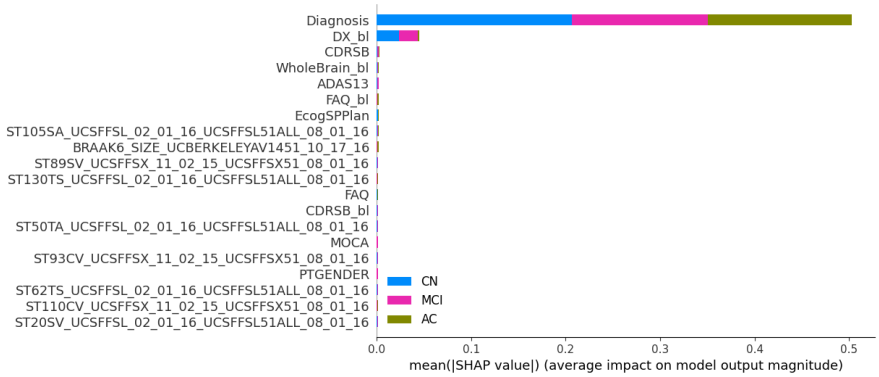
En un primer estudio se aprecia una importancia colosal al diagnóstico (figura C.1). Si prestamos atención a las gráficas violín, vemos que son de lo más coherentes: a menor valor de diagnóstico más posibilidad de CN, si toma valores medios más posibilidad de MCI y los ejemplos de Diagnosis más alto se asocian a AD. Podemos inferir que el estado clínico actual tiene un gran peso en la predicción a futuro.

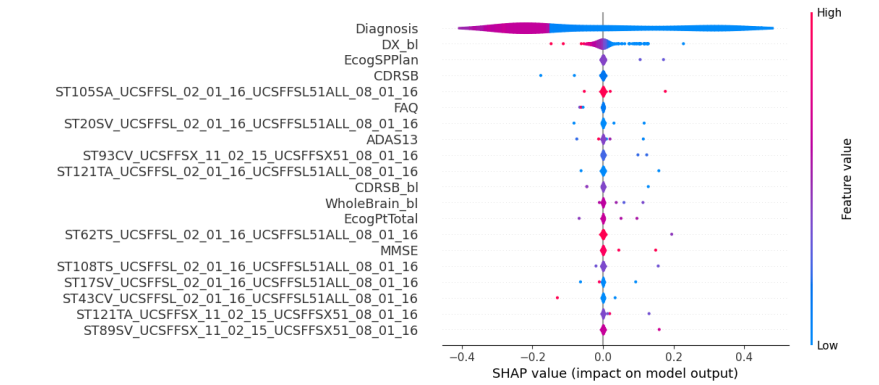
El diagnóstico no deja de ser una valoración que hacen los médicos en función de otros biomarcadores. Creemos que es de interés investigar si el rigor científico prevalece cuando se elaboran las predicciones sin conocer este estado clínico.

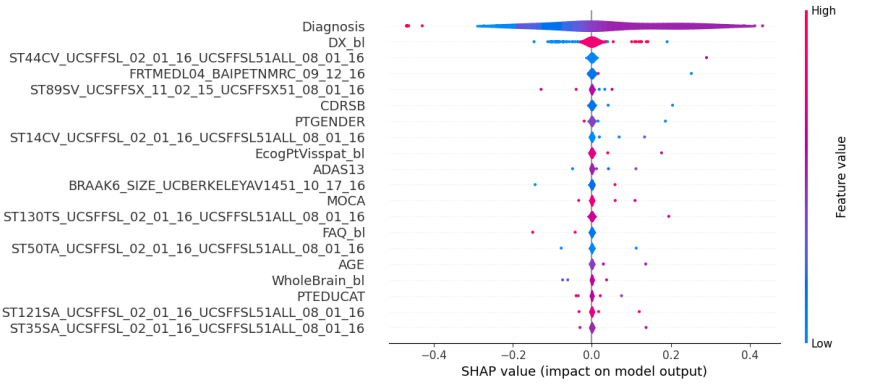
Al extraer Diagnosis y DX_bl (figura C.2) la importancia recae sobre CDRSB y de manera subalterna sobre otros test cognitivos. CDRSB (Clinical Dementia Rating - Sum of Boxes) [21] es un test cognitivo que evalúa la memoria, la orientación, la resolución de problemas, la vida social, aficiones y otros parámetros que afectan al día a día del paciente. A menor puntuación mejor estado de salud. Las gráficas vuelven a mostrar algo similar, los pacientes con menor calificación en este test se distribuyen en los que tienen un mayor valor de Shapley para CN y los que tienen una puntuación alta ostentan un Shapley negativo en CN. Evidentemente para AD ocurre justo lo contrario.

Esto es una señal de que la red funciona correctamente debido a que en ADNI se establece el diagnóstico en base a los resultados de los tests cognitivos MMSE, CDRSB y Wechsler Memory Scale–Revised [20]. Aclarar que este último no se encuentra disponible en el conjunto de datos de TADPOLE. En MMSE los criterios son opuestos a los de CDRSB; una nota alta implica una menor probabilidad de padecimiento de la enfermedad de Alzheimer. Es por eso que la información que muestra la figura tiene sentido. Nótese como MMSE influye más en la clasificación de AD.

Cabría preguntarse qué ocurre cuando se retiran esos biomarcadores relativos a la elaboración del diagnóstico. La figura C.3 muestra cómo entonces adquieren trascendencia otros test cognitivos, especialmente ADAS13 y FAQ. Era esperable que se terminara decantando por otros features cognitivos, a fin de cuentas, el diagnóstico (la clase a predecir) se apoya en ese tipo de características. Siendo que son test rigurosos y utilizados en la práctica clínica, no debería haber mucha diferencias entre los resultados de unos u otros. En ADAS y FAQ el resultado es inversamente proporcional a la salud, como en CDRSB, mientras que en RAVLT es al revés, como en MMSE.







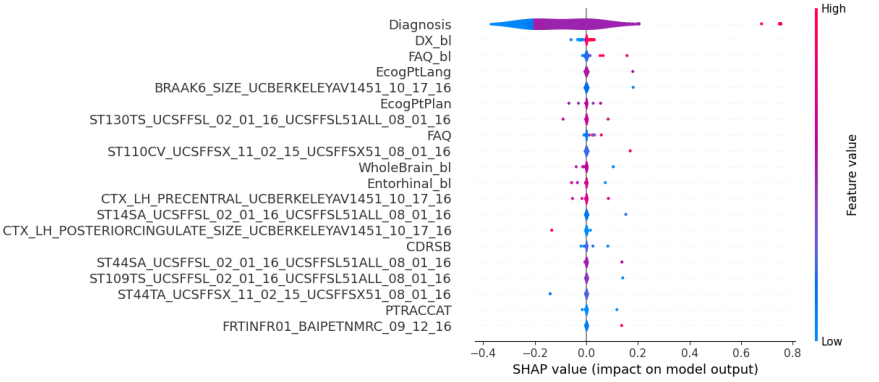


Figura C.1: Análisis de interpretabilidad de las DNN con todos los predictores. (Diagrama de barras, diagramas de violín sobre la predicción de CN, MCI y AD, respectivamente).

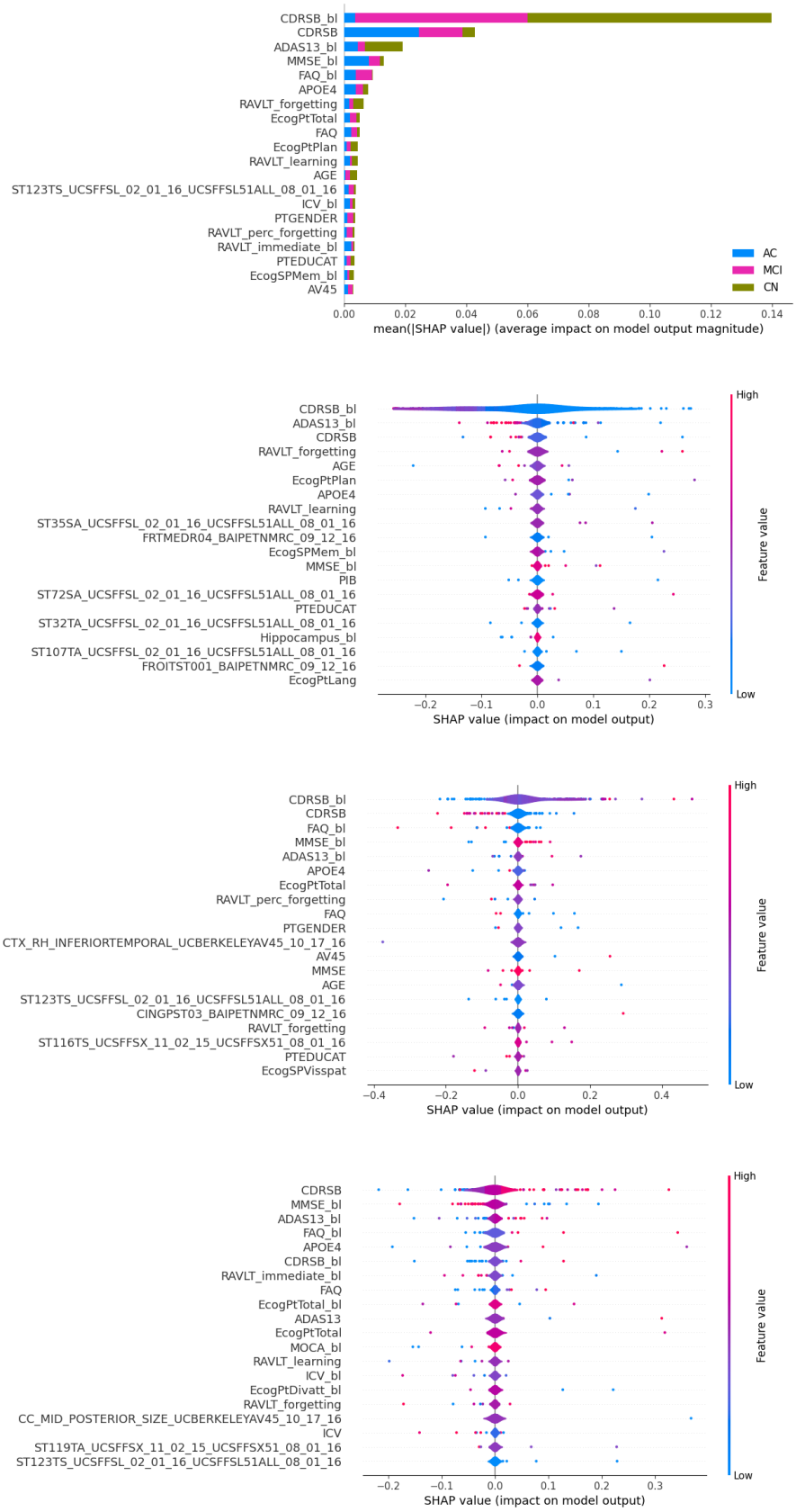


Figura C.2: Análisis de interpretabilidad de las DNN con todos los predictores salvo el diagnostico. (Diagrama de barras, diagramas de violín sobre la predicción de CN, MCI y AD, respectivamente).

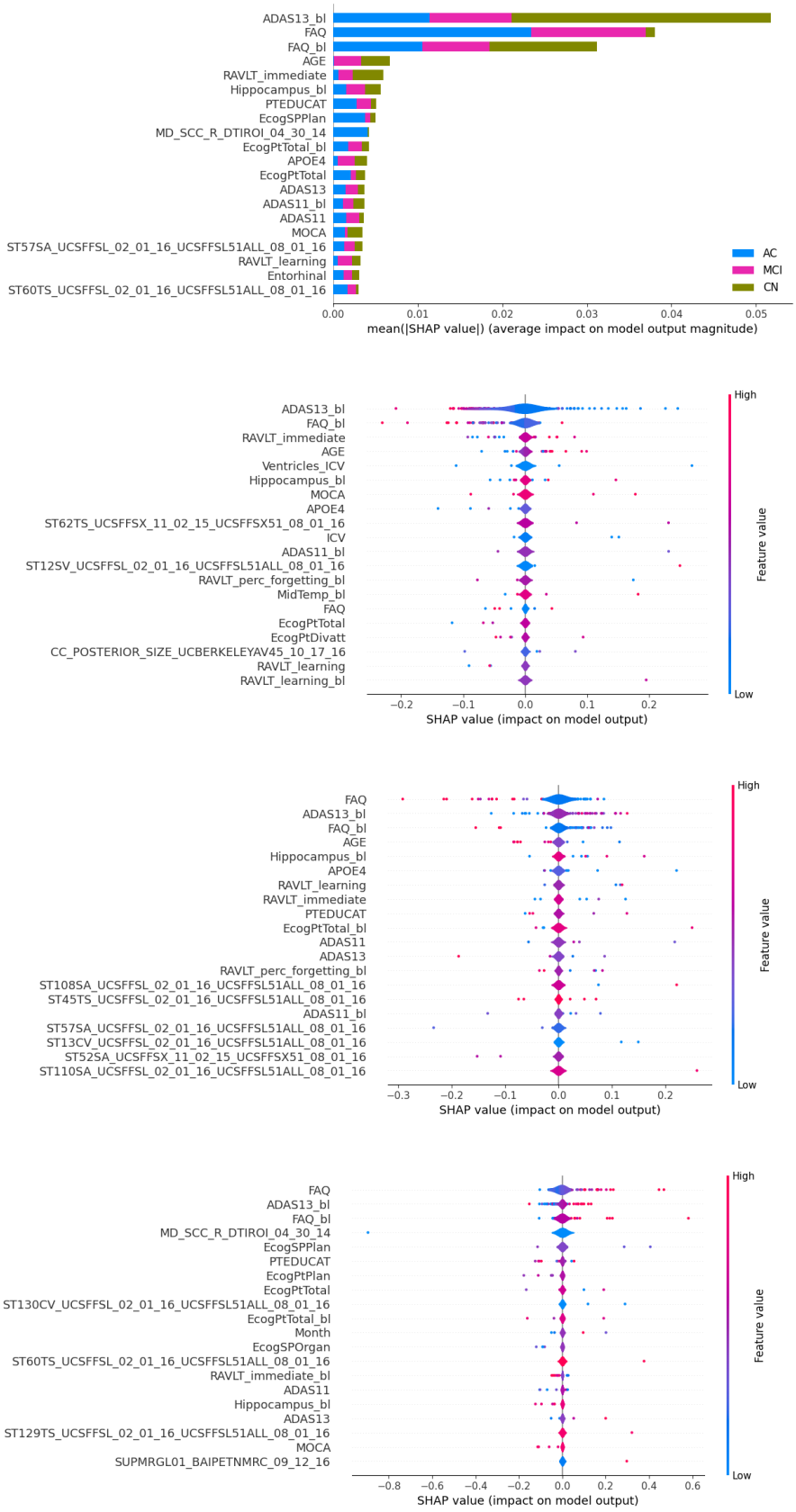


Figura C.3: Análisis de interpretabilidad de las DNN con todos los predictores salvo los relativos a la elaboración del diagnóstico. (Diagrama de barras, diagramas de violín sobre la predicción de CN, MCI y AD, respectivamente).

Finalmente si suprimimos también los test cognitivos (figuraC.4), los biomarcadores PET pasan a ser las característica más determinante, aunque la edad y el factor genético también poseen gran relevancia. A medida que se extraen predictores importantes, la calidad de los resultados baja. Ahora la mAUC y el BCA están entre 0,5 y 0,6; como ven con las mismas garantías que nos podría dar un modelo aleatorio. Por esa razón nos mostramos un tanto escépticos ante cualquier análisis que se efectúe sobre los datos que nos concede SHAP en este caso. Baste como ejemplo que según la gráfica de violín de CN a mayor edad más posibilidades de no padecer la enfermedad. A simple vista sugiere que en el dataset los pacientes sanos tienen una mayor esperanza de vida. Aunque con solo mirar el resumen de datos (tabla 2.11) se demuestra lo contrario.

En definitiva, podríamos asegurar que las predicciones de nuestros modelos estáticos se asientan sobre los mismos indicadores con los que los expertos clínicos elaboran sus diagnósticos. Lo que significa que nada se deja al azar. Es remarcable el gran peso que tienen los test cognitivos en los resultados, incluso aquellos que no utiliza ADNI.

C.2. Interpretabilidad en modelos recurrentes

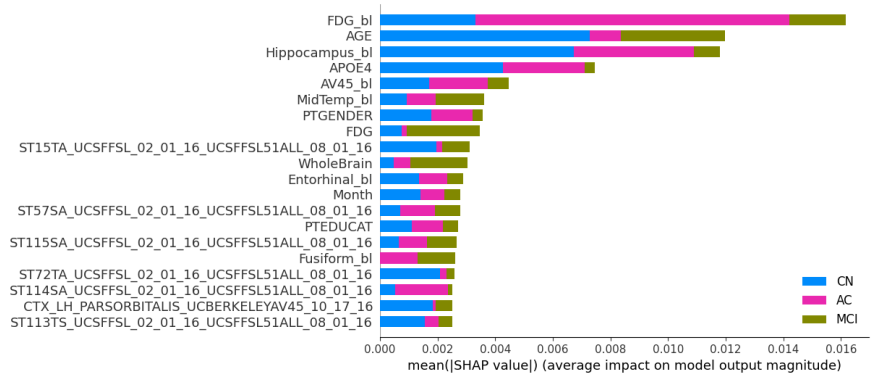
Debemos realizar la fase de interpretabilidad con técnicas recurrentes de manera semejante a la introducida en la subsección anterior, empezando con todas las features y retirándolas progresivamente. Ahora bien, se ha de tener en cuenta todas las limitaciones comentadas en la cuarta sección, es decir, no podemos transmitir al explainer de kernel la función de predicción tal cual, como hacíamos con los modelos estáticos, sino que hay que adaptar la forma del input a la demandada. De hecho, una vez adaptada la dimensionalidad, la función que se le utilice dentro del explainer deberá volver a cambiar la forma para poder invocar a la función de predicción de la RNN.

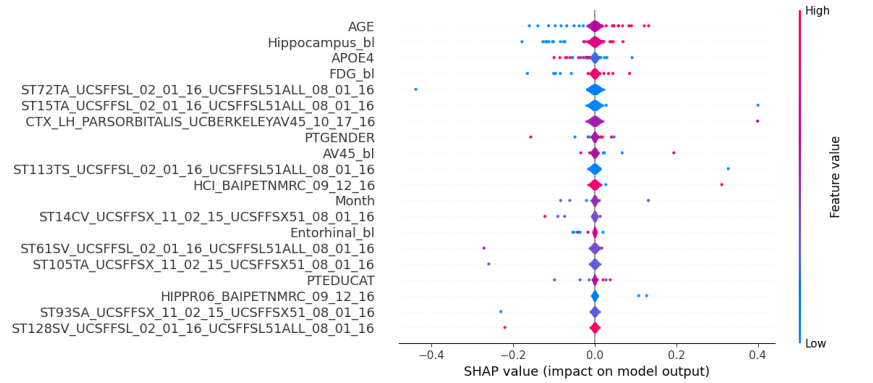
De nuevo queda constancia de que la característica de mayor peso es el estado clínico actual (figura C.5), al igual que ocurría con las DNN. Así que, en principio, comienza siendo coherente. Las gráficas de violín demuestran que los pacientes con diagnósticos más bajos son potencialmente clasificados como CN, los que adquieren valores intermedios como MCI y la gente con los más altos como AD. En segundo lugar, ostentan una buena posición los tests cognitivos CDR y MMSE, era esperable, al fin y al cabo, son los que se utilizan en la elaboración del diagnóstico. A continuación se realizará la prueba de retirar las características del diagnóstico, con el fin de comprobar que las pruebas cognitivas consiguen el papel protagonista.

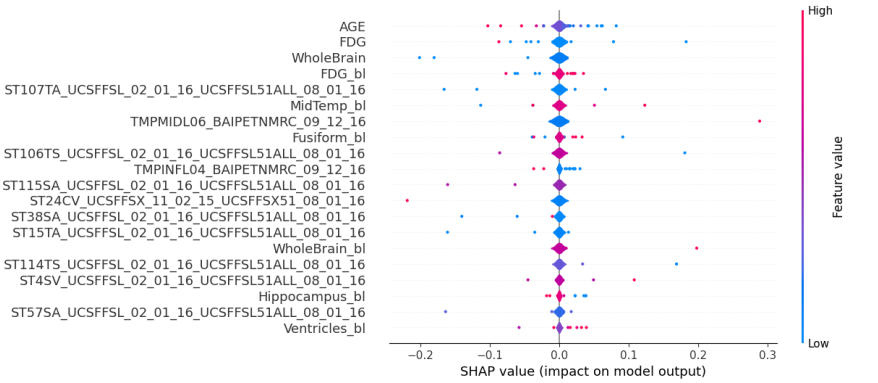
Efectivamente, sin el diagnóstico, el ranking sale según lo previsible (figura C.6), salvo por MMSE, que se esperaba una posición más alta. En la red neuronal profunda los test CDR eran los responsables de las predicciones (figura C.2), y el resto de predictores hacían una aportación mínima. En este caso FAQ también tiene bastante influencia en el resultado final, aunque solo sea porque los valores de SHAP de los CDR son bastante inferiores a los de la DNN. Los gráficos de violín nos muestran como la puntuación de los tests es acorde al valor de SHAP que se le asigna para cada predicción. E. g. en CN las puntuaciones menores de CDRSB tienen un mayor valor de Shapley.

Si continuamos con el análisis, ahora sería el turno de apartar las características referentes a la elaboración del diagnóstico (figura C.7). En esta ocasión, FAQ sobresale frente a otros test y en general los biomarcadores con mayor repercusión son test cognitivos. ADAS13 ya no le caracteriza una puntuación media tan alta como en la DNN, aunque, aún así, alcanza la segunda posición. Cuando, finalmente, extraemos de nuestras pruebas todos los tests cognitivos (figura C.8), las características dominantes son FDG, la edad, el hipocampo, el factor genético. No hay un grupo manifiesto que destaque sobre los demás. Además nos volvemos a mover en puntuaciones muy bajas y se detectan ciertas incongruencias, como la de la edad, comentada en la subsección anterior.

Concluimos con unas ideas similares a las del análisis anterior, señal de que este modelo también







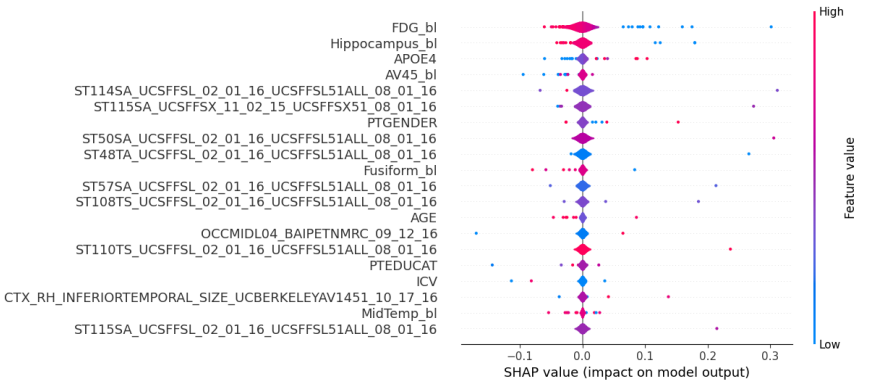


Figura C.4: Análisis de interpretabilidad de las DNN con todos los predictores salvo los relativos a la elaboración del diagnóstico y test cognitivos. (Diagrama de barras, diagramas de violín sobre la predicción de CN, MCI y AD, respectivamente).

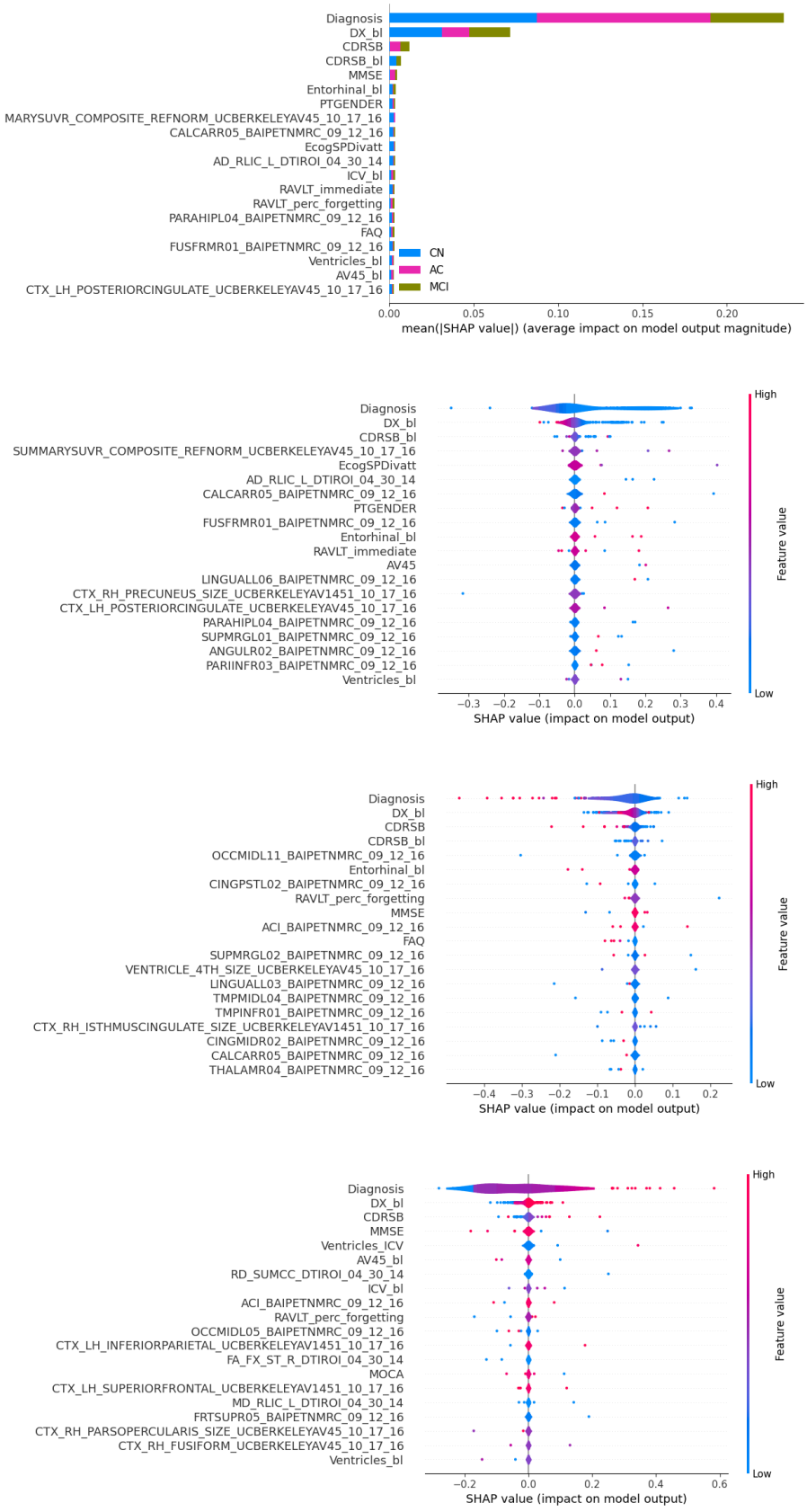
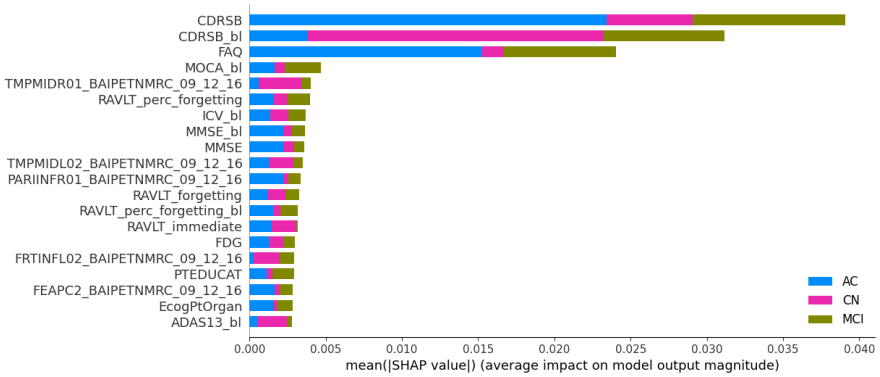
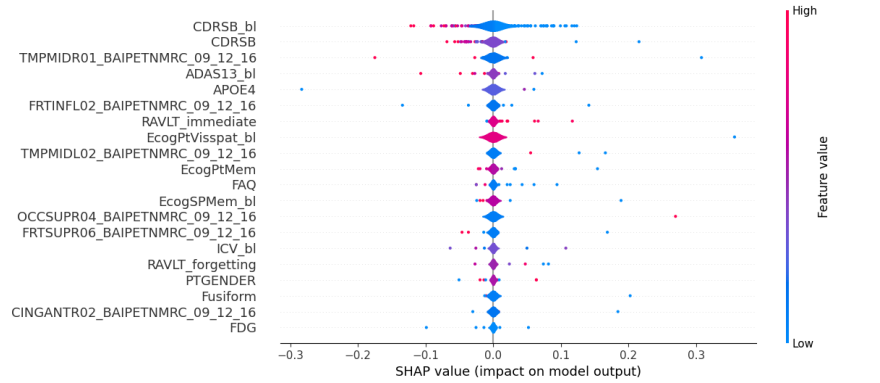
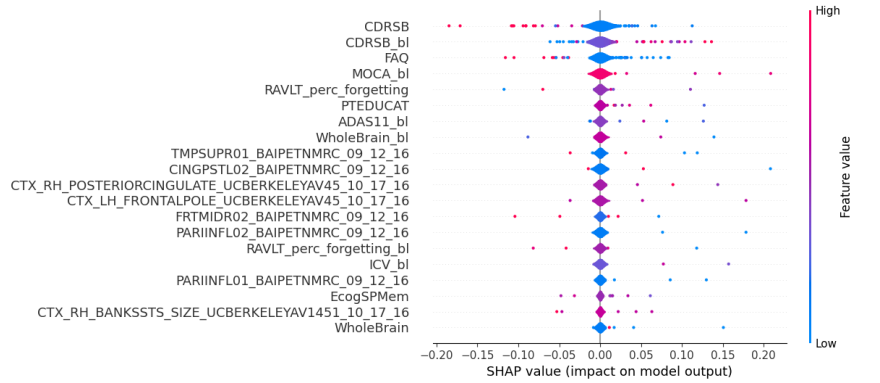


Figura C.5: Análisis de interpretabilidad de las RNN con todos los predictores. (Diagrama de barras, diagramas de violín sobre la predicción de CN, MCI y AD, respectivamente).







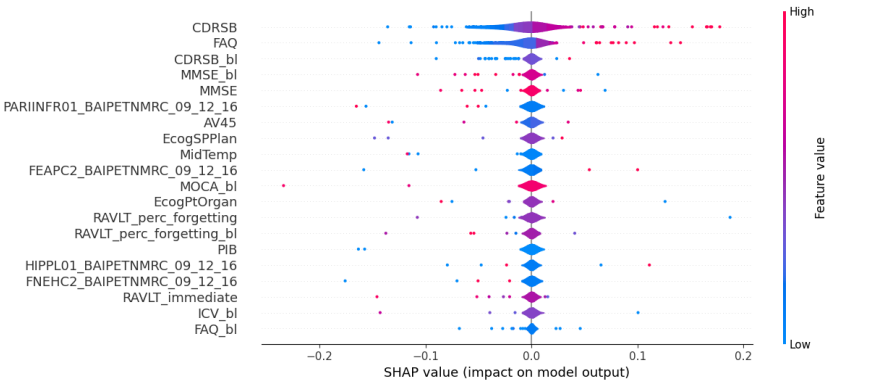
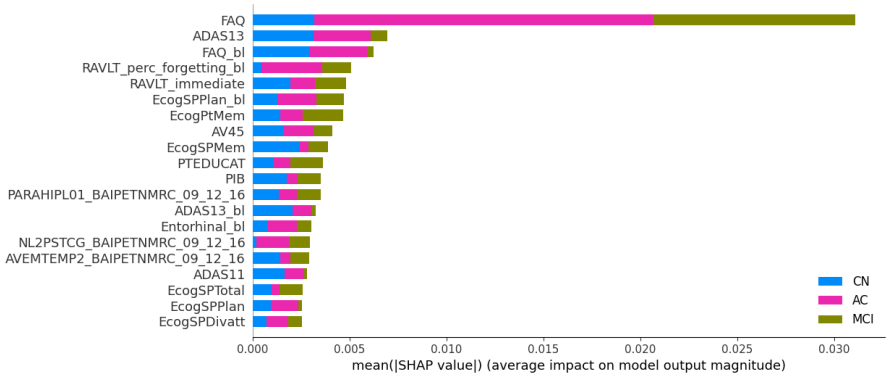
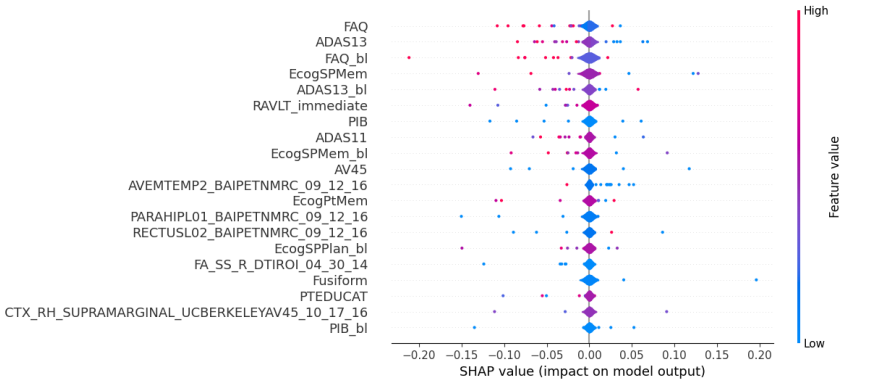
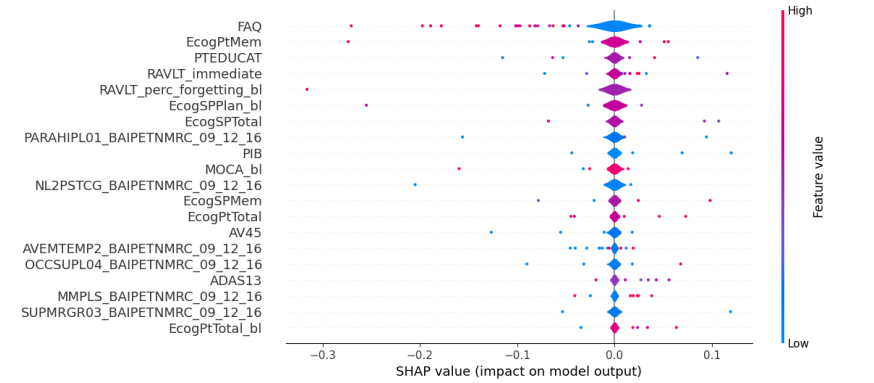


Figura C.6: Análisis de interpretabilidad de las RNN con todos los predictores salvo el diagnostico. (Diagrama de barras, diagramas de violín sobre la predicción de CN, MCI y AD, respectivamente).







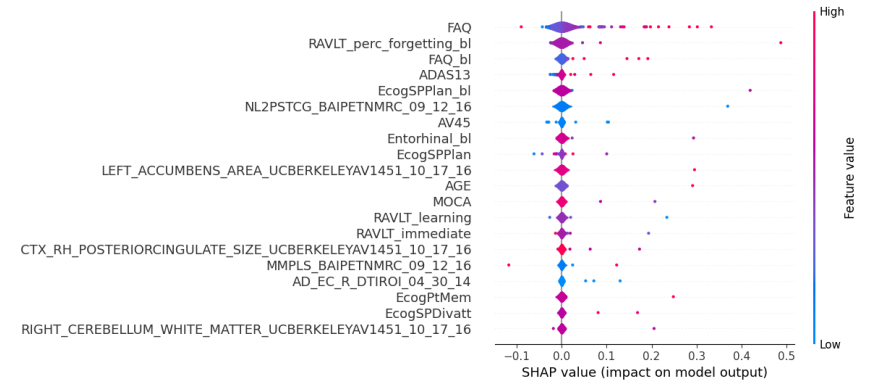
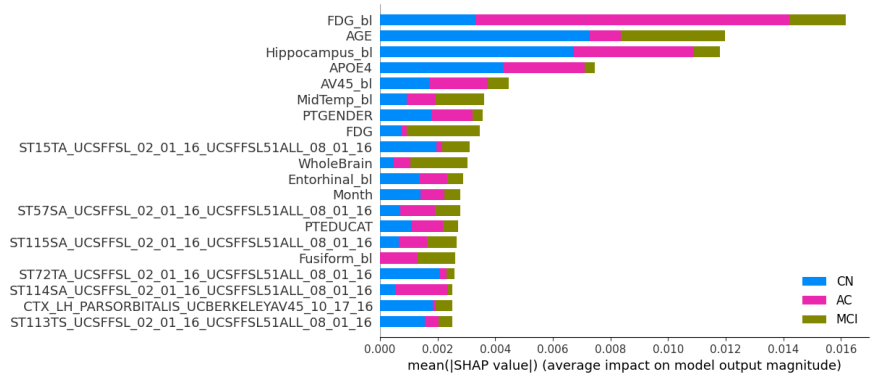
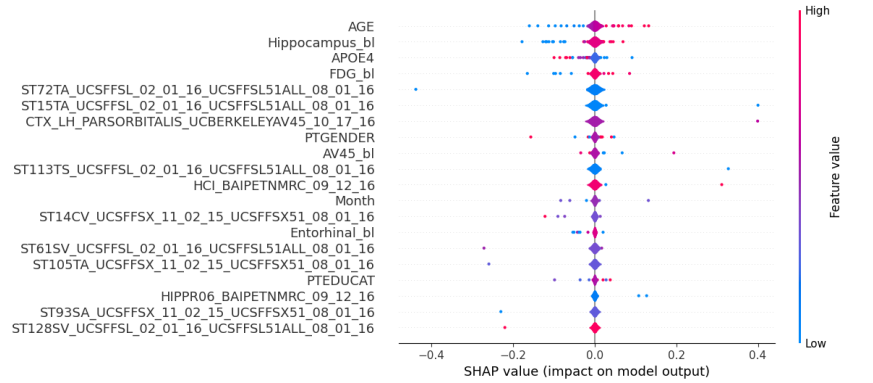
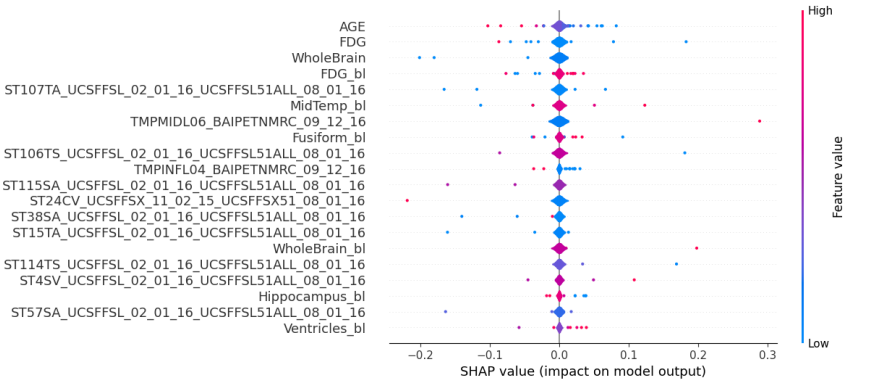


Figura C.7: Análisis de interpretabilidad de las RNN con todos los predictores salvo los relativos a la elaboración del diagnóstico. (Diagrama de barras, diagramas de violín sobre la predicción de CN, MCI y AD, respectivamente).







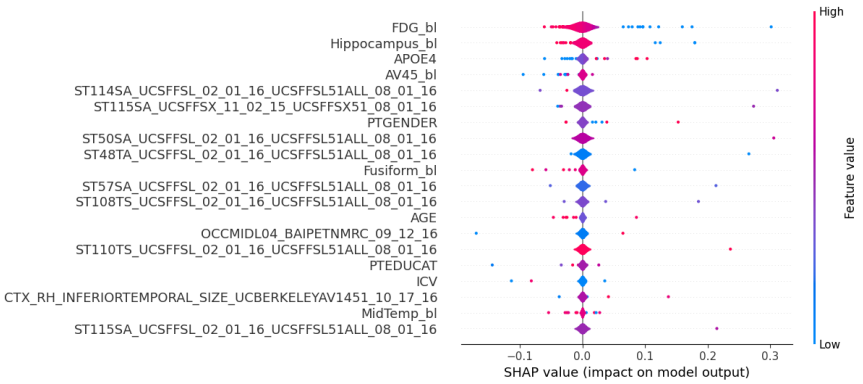


Figura C.8: Análisis de interpretabilidad de las RNN con todos los predictores salvo los relativos a la elaboración del diagnóstico y test cognitivos. (Diagrama de barras, diagramas de violín sobre la predicción de CN, MCI y AD, respectivamente).

es racional. En suma, para determinar el diagnóstico se apoya esencialmente en test cognitivos, como hacen en ADNI.

D. Análisis de la configuración interna de las redes

No podíamos poner fin a este trabajo sin adentrarnos en los parámetros internos de nuestras redes para satisfacer nuestra curiosidad. Se ha decidido utilizar como herramienta de visualización TensorBoard, la misma que nos trazó las curvas de aprendizaje. Este kit nos permite mirar la distribución y los histogramas de pesos y sesgos de cada capa. Como a veces los gráficos no son suficientes, plasmaremos en un csv los pesos que asigna la red en la primera capa a cada feature, por si en alguna prueba precisamos de información complementaria.

Por una cuestión de simplicidad, vamos a empezar ahondando en los atributos de la red neuronal poco profunda. Se trata de una sola capa, así que solo se puede prestar atención al valor de los pesos y el sesgo que se asignan a ese único grupo de neuronas. Sin lugar a dudas, lo que llama la atención es la enorme cantidad de características que tienen un peso cercano a cero. Si nos fijamos en la distribución (figura D.1), vemos como el pico se sitúa entorno a los 3500 predictores. Al diagnóstico se le da un peso de 0,139, teniendo en cuenta que la media de los valores absolutos está en 0,00039 y la suma es de 0,712, hay un desmedido número de predictores que no aportan nada a la predicción final y la responsabilidad de las decisiones tomadas recae mayormente en el estado clínico. En general, si ordenamos las características por peso, el ranking se asemeja al obtenido por SHAP.

Conviene hacer un pequeño paréntesis, para no caer en lecturas erróneas. El peso no es cómo el valor de Shapley. Si la suma de pesos es 100 y el de una característica es de 10, no quiere decir que tenga un 10 % de importancia, ya que las predicciones dependen de otros factores.

Cuando hacemos una selección de características, la distribución no es tan uniforme en cero, sino que este subconjunto de características tiende a adoptar valores más altos (figura D.2). Contamos ahora con un peso en el diagnóstico de 0,0093, más cercano a la media de 0.0016. De esta manera, corroboramos que la selección de características cubre el subconjunto de las más trascendentes, porque todas contribuyen en gran medida al resultado final, no es como antes que a la mayoría se les asignaba un peso igual, o muy cercano, a cero.

Conforme añadimos más capas observamos como en el pico del cero se va aplanando (figura D.3). Eso se traduce a que la cantidad de valores que es próxima a cero se va reduciendo. Lo cual no significa que ahora haya más biomarcadores condicionando los resultados finales, ya que en las capas siguientes vuelve a manifestarse ese pico. Es por eso que el incrementar las capas no supone una mejora del rendimiento, una holgada mayoría de neuronas no aportan nada al modelo. En general, el aumento de capas implica un incremento de precisión en modelos en los que la información está jerarquizada y la red puede ir abstrayendo por niveles más información. Aquí no se presenta esa jerarquía bien definida.

Figura D.1: Histograma de pesos de la primera capa (y única) de la red Shallow tras entrenarla con todos los features.

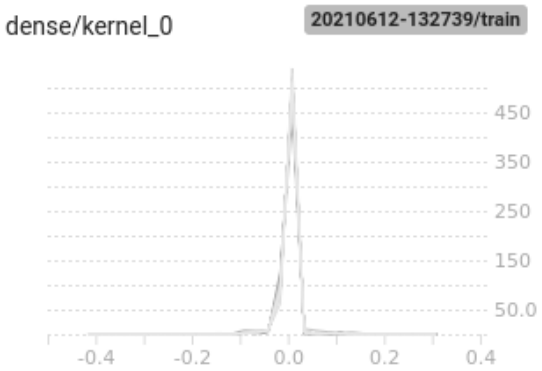


Figura D.2: Histograma de pesos de la primera capa (y única) de la red Shallow tras entrenarla con un conjunto reducido de features.

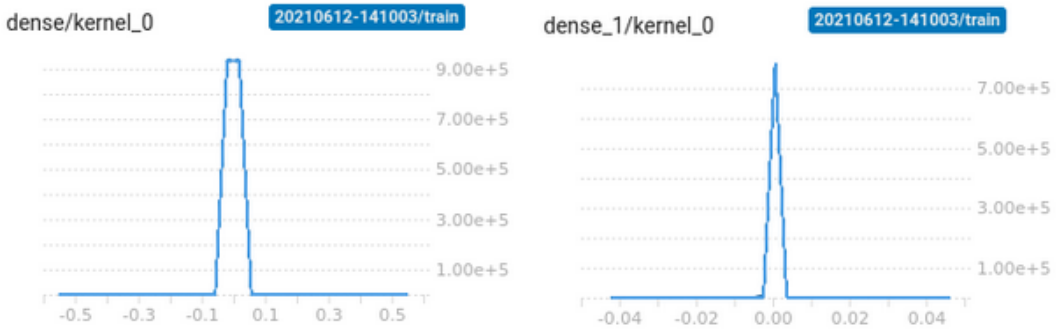


Figura D.3: Histograma de pesos de la primera capa (izq) y segunda capa (dcha) de la DNN tras entrenarla con todas las features.

Esa meseta aparece solo cuando se ejecuta la DNN con todas las features, con un conjunto reducido, adopta su forma puntiaguda original. Es decir, a menor número de predictores, más neuronas inactivas y, por tanto, menos provecho se saca del potencial de la red.

En general se puede percibir que estamos utilizando un método demasiado complejo y que hay características que no aportan tanto como nos gustaría, ya hemos visto en el anexo anterior que sin test cognitivos los resultados son un tanto mediocres. Posiblemente con un mayor número de muestras la red podría dar todo su potencial y desbancar a los métodos que ganaron el concurso.

En las redes neuronales recurrentes existen dos tipos de pesos: el que se aplica sobre la activación y el de entrada. En la entrada se aprecia una cima, como en casos anteriores pero con una base más amplia (figura D.4). Al examinarlos en detalle se observa que ya no existe una brecha tan acentuada entre los pesos asignados a cada característica de entrada, como ocurría anteriormente. Sin embargo, SHAP sí que seguía constatando que la variable dominante con diferencai era el diagnostico. Así que podemos concluir que los pesos de la primera capa no ofrecen una visión suficiente para interpretar la importancia de cada feature. Ahora no ocurre como en la Shallow, sino que la predicción final depende de más pesos y variables. Si prestamos atención al modelo entrenado con más features, el comportamiento es similar al de la DNN, la primera capa asigna mucho peso a unas pocas features y al resto casi nada. De esa forma, hace un primer cribado, se queda con las características que piensa que puede contribuir en mayor medida. En la segunda capa hace otro cribado, en las que selecciona ese *top* que tomaba directamente en los modelos de una sola capa. Nuevamente vemos que este fenómeno no se da en el modelo entrenado con unas pocas características porque esa separación ya la hemos hecho nosotros mismos previamente. De ahí que la red dé resultados prácticamente iguales entre ambas versiones.

En lo que respecta a la distribución los pesos de la activación, también tienen la media en cero, pero en esa ocasión la forma es más bien gausiana, en especial en la segunda capa. Tiene sentido que se le dé más peso aquí que a la entrada, hay que tener en cuenta que en la entrada intervienen mucha características que no aportan nada al modelo, mientras que aquí nos llega información de la marca anterior justo después de haber realizado ese cribado.

En LSTM se aprecia un pico más pronunciado en todas las gráficas (figura D.5). Volvemos a notar cómo la mayoría de los componentes de la red tienen un peso muy cercano a cero, es decir, contribuyen mínimamente al resultado final. Seguramente con secuencias temporales más largas, o mejor dicho, con más información, sin tantas marcas temporales vacías entremedio, los pesos adoptarían valores menos uniformes y más alejados a cero y la red sacaría partido de todo su potencial.

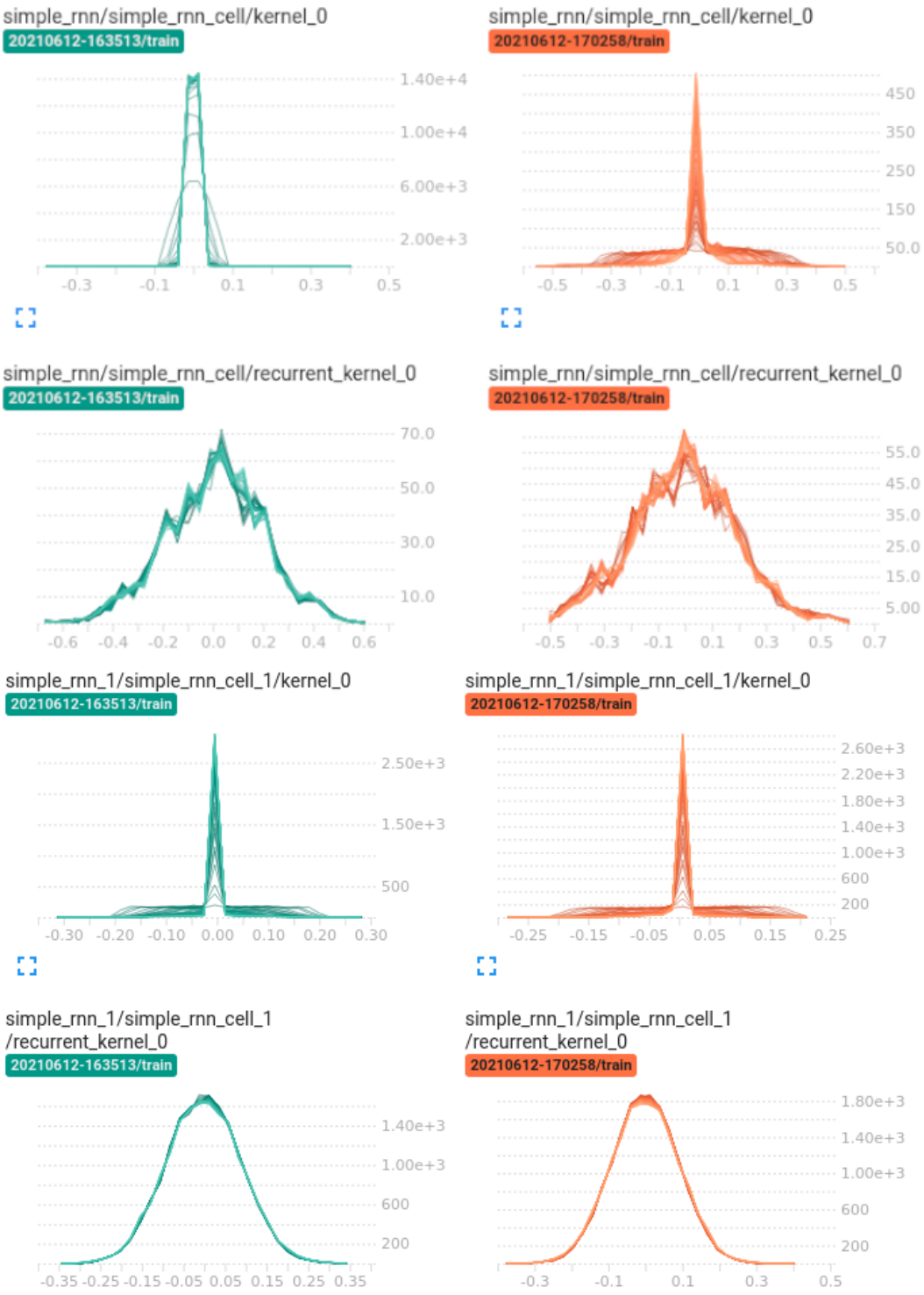


Figura D.4: Histograma de pesos de la RNN. La primera columna es el resultado de entrenar la red con todas las features menos las UCS, la de la derecha con una selección de 32 features. La primera fila son los pesos de entrada de la primera capa, la segunda los pesos de activación, la tercera los pesos de entrada de la segunda capa y la última los pesos de activación de la segunda capa.

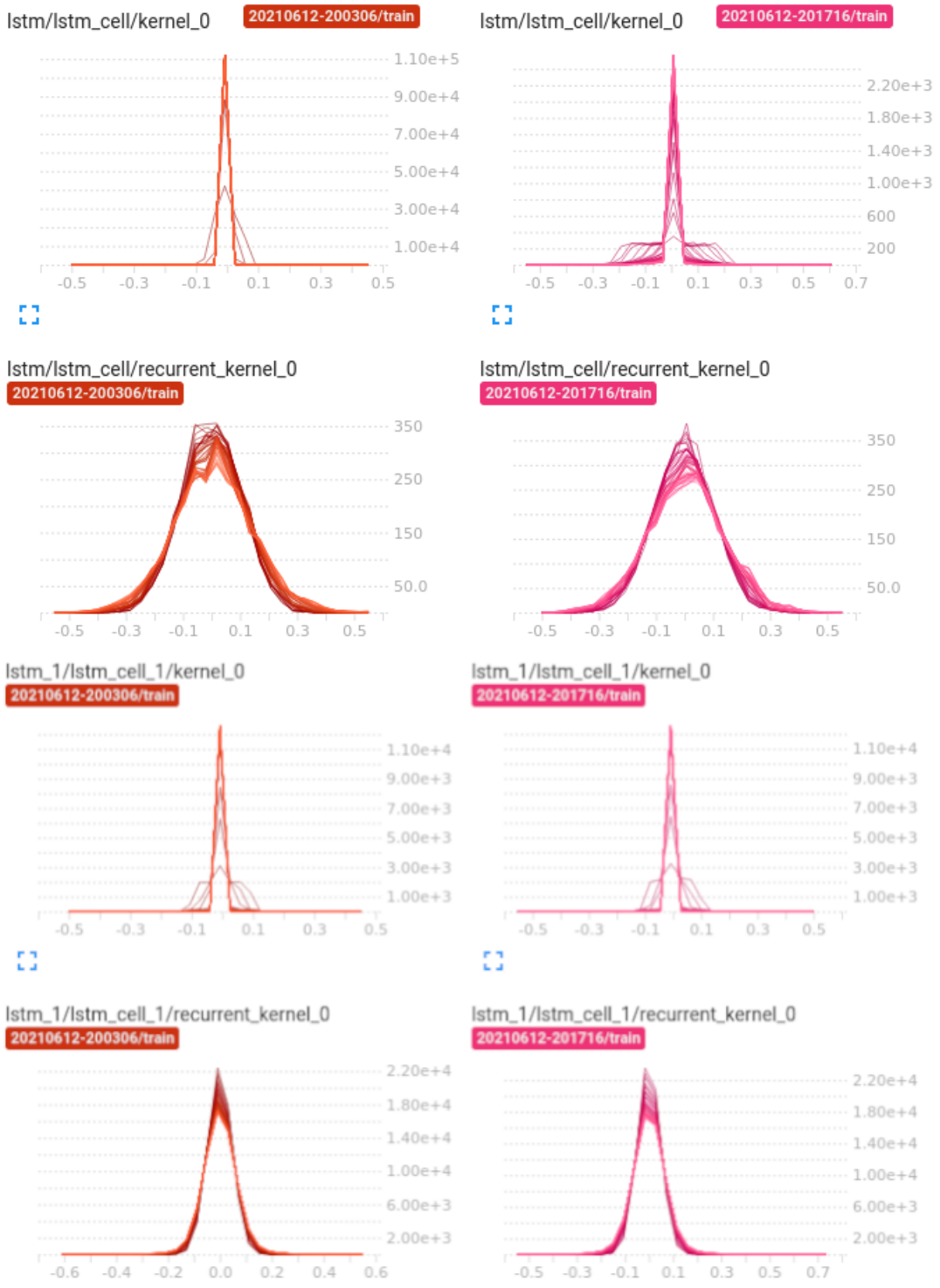


Figura D.5: Histograma de pesos de la LSTM. La primera columna es el resultado de entrenar la red con todas las features menos las UCS, la de la derecha con una selección de 32 features. La primera fila son los pesos de entrada de la primera capa, la segunda los pesos de activación, la tercera los pesos de entrada de la segunda capa y la última los pesos de activación de la segunda capa.

Referencias

[1] Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer, Cham, 2018.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: a review and new perspectives. *IEEE Trans Pattern Anal Mach Intell*, 35(8):1798–1828, Aug 2013.

[3] Edward Choi and Mohammad Taha Bahadori et al. Doctor ai: Predicting clinical events via recurrent neural networks. In Finale Doshi-Velez, Jim Fackler, David Kale, Byron Wallace, and Jenna Wiens, editors, *Proceedings of the 1st Machine Learning for Healthcare Conference*, volume 56 of *Proceedings of Machine Learning Research*, pages 301–318, Northeastern University, Boston, MA, USA, 18–19 Aug 2016. PMLR. `http://proceedings.mlr.press/v56/Choi16.html` [Último acceso 13 de Junio de 2021].

[4] Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization, 2017.

[5] E. H. Corder and A. M. et al. Saunders. Gene dose of apolipoprotein E type 4 allele and the risk of Alzheimer's disease in late onset families. *Science*, 261(5123):921–923, Aug 1993.

[6] Organización Mundial de la Salud. Dementia fact sheet. Sep 2020. `https://www.who.int/es/news-room/fact-sheets/detail/dementia` [Último acceso 13 de Junio de 2021].

[7] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. `https://www.sciencedirect.com/science/article/pii/036402139090002E` [Último acceso 13 de Junio de 2021].

[8] André Ferreira. Interpreting recurrent neural networks on multivariate time series. *Towards data science*, Aug 2019.

[9] Francisco Ferraz García. Estudio de la reproducibilidad e interpretabilidad de los métodos más precisos del TADPOLE Challenge para el diagnóstico y pronóstico de la enfermedad de Alzheimer. *Universidad de Zaragoza*, Sep 2020.

[10] Lichy Han and Maulik Kamdar. Mri to mgmt: predicting methylation status in glioblastoma patients using convolutional recurrent neural networks. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, 23:331–342, 01 2018.

[11] David Hand and Robert Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Hand, The*, 45:171–186, 11 2001.

[12] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991.

[13] Sepp Hochreiter, Martin Heusel, and Klaus Obermayer. Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14):1728–1736, 05 2007. `https://doi.org/10.1093/bioinformatics/btm247` [Último acceso 13 de Junio de 2021].

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. `http://arxiv.org/abs/1502.03167` [Último acceso 13 de Junio de 2021].

[15] Octavio Loyola-González. Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE Access*, 7:154096–154113, 2019.

[16] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874, 2017. `http://arxiv.org/abs/1705.07874/` [Último acceso 13 de Junio de 2021].

[17] Razvan V. Marinescu and Neil P. et al. Oxtoby. The Alzheimer's Disease Prediction Of Longitudinal Evolution (TADPOLE) Challenge: Results after 1 Year Follow-up. *arXiv e-prints*, page arXiv:2002.03419, February 2020.

[18] Christoph Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/ [Último acceso 13 de Junio de 2021].

[19] A. Ng. Machine learning, 2011. https://www.coursera.org/learn/machine-learning [Último acceso 13 de Junio de 2021].

[20] R. C. Petersen and P. S. et al Aisen. Alzheimer's Disease Neuroimaging Initiative (ADNI): clinical characterization. *Neurology*, 74(3):201–209, Jan 2010.

[21] M. N. Samtani and N. et al. Raghavan. Disease progression model for Clinical Dementia Rating-Sum of Boxes in mild cognitive impairment and Alzheimer's subjects from the Alzheimer's Disease Neuroimaging Initiative. *Neuropsychiatr Dis Treat*, 10:929–952, 2014.

[22] M. W. Weiner and Aisen et al. The Alzheimer's disease neuroimaging initiative: progress report and future plans. *Alzheimers Dement*, 6(3):202–211, May 2010.

[23] M. W. Weiner and Veitch et al. 2014 Update of the Alzheimer's Disease Neuroimaging Initiative: A review of papers published since its inception. *Alzheimers Dement*, 11(6):1–120, Jun 2015.

[24] Junhao Wen, Elina Thibeau-Sutre, Jorge Samper-González, Alexandre Routier, Simona Bottani, Stanley Durrleman, Ninon Burgos, and Olivier Colliot. Convolutional neural networks for classification of alzheimer's disease: Overview and reproducible evaluation. *CoRR*, abs/1904.07773, 2019. http://arxiv.org/abs/1904.07773 [Último acceso 13 de Junio de 2021].

[25] A. Zell. *Simulation neuronaler Netze*. 1994.