Universitat Oberta de Catalunya (UOC)

Máster Universitario en Ciencia de Datos (*Data Science*)

TRABAJO FINAL DE MÁSTER

Aplicación de algoritmos de aprendizaje automático para predecir la disfunción cognitiva en pacientes de esclerosis múltiple mediante índices de conectividad estructural

Autor: Efraín Lima Miranda

Tutor: Dr. Eloy Martínez de las Heras

Co-Tutor: Dra. Sara Llufriu

Profesor: Dr. Jordi Casas Roma

Barcelona, 25 de junio de 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	Aplicación de algoritmos de aprendizaje automático para predecir la disfunción cognitiva en pacientes de esclerosis múltiple mediante índices de conectividad estructural
Nombre del autor:	Efraín Lima Miranda
Nombre de los colaboradores	Dr. Eloy Martínez de las Heras Dra. Sara Llufriu
Nombre del PRA:	Dr. Jordi Casas Roma
Fecha de entrega:	Junio de 2018
Titulación o programa:	Máster Universitario en Ciencia de Datos (*Data Science*)
Área del Trabajo Final:	Aprendizaje automático
Idioma:	Español
Palabras clave:	aprendizaje automático, conectividad estructural, esclerosis múltiple, rendimiento cognitivo

Dedicado a Laura, mi felicidad de cada día.

Agradecimientos

Deseo expresar mi más sincero agradecimiento:

A mi tutor de este proyecto, Dr. Eloy Martínez de las Heras, por su apoyo personal y humano durante todo este trabajo.

A Dra. Sara Llufriu y a todo el equipo del Institut d'investigacions Biomèdiques August Pi i Sunyer (IDIBAPS) junto al grupo de Imagen Avanzada en enfermedades Neuroinmunológicas (ImaginEM) su por su ardua labor sin la cual que este proyecto hubiera sido imposible.

A Dr. Jordi Casas y toda la comunidad educativa de la Universitat Oberta de Catalunya.

A todas las personas anónimas que a través de sus datos colaboran con la investigación médica.

A todos, mil gracias.

Resumen

Este trabajo tiene como objetivo la predicción del rendimiento cognitivo del paciente con esclerosis múltiple (EM) a través de la cuantificación del volumen lesional y la microestructura de la red cerebral empleando herramientas de aprendizaje automático. Esta enfermedad neurodegenerativa es una de las causas más importantes de discapacidad física y cognitiva en adultos jóvenes.

Para esta investigación se ha dispuesto de una muestra de 182 sujetos, donde 140 padecen de EM y 42 son controles sanos. De cada uno de ellos se dispone de cuatro medidas de tensor de difusión (DTI) (Anisotropía fraccional, Difusividad media, Difusividad axial y Difusividad radial), número de fibras y volumen lesional. Toda esta información proveniente del análisis de la conectividad estructural es presentada mediante matrices simétricas.

Tras realizar las tareas de preprocesamiento y limpieza de toda esta información, con el software NeuLoadData, se han estimado los mejores parámetros de configuración para los algoritmos "Logistic Regression", "Support Vector Machine (SVM)", "Gaussian Naive Bayes, Random Forest Classifier" y "Artificial Neural Network (ANN)". Usando únicamente las medidas de tensor de difusión todos los modelos obtenidos han sido capaces de predecir exitosamente más del 75 %. Por lo tanto, el enfoque propuesto de aprendizaje automático para la predicción el rendimiento cognitivo en pacientes con EM ha demostrado su utilidad e interés como herramienta para analizar un gran conjunto de datos satisfactoriamente en el campo sanitario.

Palabras clave: aprendizaje automático, conectividad estructural, esclerosis múltiple, rendimiento cognitivo

Abstract

The present study aims to predict the cognitive performance of patients with multiple sclerosis (MS) through quantification of lesional volume and the microstructure of the brain network by means of machine learning techniques. This neurodegenerative disease is one of the main causes of both physical and cognitive disability in young adults.

For this research, we have a sample of 140 patients with Multiple Sclerosis and 42 healthy controls. For each participant, information relating to four measures of diffusion tensor (DTI) (fractional anisotropy, medium diffusivity, axial diffusivity and radial diffusivity), number of fibers and lesional volume has been recorded. All this information coming from the analysis of structural connectivity is presented by symmetric matrices.

After carrying out preprocessing tasks on all of this information using the NeuLoadData software, the best configuration parameters have been estimated for the algorithms Logistic Regression, Support Vector Machine (SVM), Gaussian Naive Bayes, Random Forest Classifier and Artificial Neural Network (ANN). Making use of only diffusion tensor measurements, all the models have had a successful prediction rate of over 75 %. Therefore, the proposed approach of machine learning for prediction cognitive performance in patients with MS has demonstrated satisfactorily its usefulness and interest as a tool to analyze a large set of data in the health field.

Keywords: manchine learning, structural connectivity, multiple sclerosis, cognitive performance

Índice general

Índice de figuras

Índice de cuadros

Acrónimos

AD Difusividad axial. 5

ANN Red neuroal artificial. VII, 9, 29

CSP Red neuronal convolucional. 9, 10, 51

CSV Comma-Separated Values. 19

DBN Deep Belief Network. 9

dRM Difusión por resonancia magnética. 4, 18

DTI Tensor de difusión. VII, VIII, 4, 5, 18, 34, 41–46, 50

EM Esclerosis múltiple. 4–6, 8, 9, 17, 19, 49, 50

FA Anisotropía fraccional. 4, 5, 18, 19

fRM Resonancia magnética funcional. 9

ICA Independent Component Analysis. 10

K-Fold Validación cruzada de K iteraciones. VII, 31, 32, 35

K-NN Algoritmo de los k-vecinos más cercanos. 9

MD Difusividad media. 5, 18

PCA Principal Component Analysis. 10, 30

RBM Retricted Bolzmann Machines. 9

RD Difusividad radial. 5

RM Resonancia magnética. 4, 6, 9

ROI Regiones de interés. 9

SB Sustancia blanca. 4, 5

SBAN Sustancia blanca de apariencia normal. 4, 5

SG Sustancia gris. 5

SGAN Sustancia gris de apariencia normal. 4

SNC Sistema nervioso central. 4

SVM Máquina de vector de soporte. VII, 9, 28

Parte I

Prolegómeno

Capítulo 1

Introducción

La esclerosis múltiple (EM) es una enfermedad neurodegenerativa crónica, inflamatoria y desmielinizante del sistema nervioso central (SNC), de carácter autoinmune y considerada como una de las causas más importantes de discapacidad física y cognitiva en adultos jóvenes [5]. Esta enfermedad se caracteriza principalmente por la presencia de placas desmielinizadas focales, y por una afectación microestructural difusa más allá de las lesiones en la sustancia blanca de apariencia normal (SBAN) y la sustancia gris de apariencia normal (SGAN). Ambos componentes son los responsables de la atrofia cerebral y, en cierta medida, están asociados con la discapacidad cognitiva [6]. Este deterioro cognitivo está presente en el 40-70 % de los pacientes, incluso durante etapas tempranas de la enfermedad. Los déficits cognitivos más comunes son los relacionados con la atención, las funciones ejecutivas, la velocidad de procesamiento de la información y la memoria episódica [7].

La resonancia magnética (RM) convencional ha demostrado ser una técnica útil para el diagnóstico y monitorización de la enfermedad de EM. La presentación de lesiones características de la enfermedad se muestran hipointensas en secuencias potenciadas en T1 y con un aumento de señal en secuencias potenciadas en T2. Sin embargo, la asociación entre la presencia de lesiones y las manifestaciones clínicas en los pacientes es modesta [8]. Probablemente, a causa de la baja especificidad en la patología subyacente y a la baja sensibilidad del daño del tejido de apariencia normal en este tipo de secuencias. La existencia de nuevas modalidades de imagen por RM, denominadas RM avanzada o no convencional, ha aportado información más sensible más allá de las lesiones focales, permitiendo estudiar el tejido de apariencia normal. Una de las modalidades más populares actualmente es la difusión por resonancia magnética (dRM). Esta técnica de RM se basa en el estudio del movimiento de las moléculas de agua y permite caracterizar las trayectorias de sustancia blanca (SB) de forma no invasiva, dado que en la SB este movimiento de moléculas de agua predomina en la dirección paralela al axón y se encuentra limitado en su dirección perpendicular [9]. Por tanto, a partir de la dRM podemos obtener datos cuantitativos y patológicamente más específicos capaces de detectar cambios en la integridad microestructural en SBAN y SGAN. A partir de la dRM podemos obtener unas medidas que se conocen como tensor de difusión (DTI). Estas medidas cuantifican la direccionalidad y magnitud del movimiento de las moléculas de agua en el espacio tridimensional. Dado que existe una fuerte direccionalidad debido a la presencia de mielina y axones en la SB, el movimiento es anisotrópico y puede representarse como una elipsoide, donde el semieje principal indica la máxima difusividad. La elipsoide se puede parametrizar por un conjunto de vectores y valores propios que nos proporcionan unos índices sensibles a la integridad del tejido. El más común es la anisotropía fraccional (FA).

La FA es una variable numérica cuyos valores oscilan entre 0 (máxima isotropía) y 1 (máxima anisotropía). La FA es mayor en la SB que en la sustancia gris (SG), debido a que la movilidad del agua está altamente influenciada por la organización de las fibras nerviosas. Por este motivo, el valor de FA es comúnmente utilizado en los estudios como un marcador de la integridad estructural, ya que la pérdida de barreras reduce el grado de anisotropía (menor FA). Hay otros valores del tensor que se pueden cuantificar como: la difusividad media (MD), la difusividad axial (AD) y la difusividad radial (RD). Mientras que la FA y la MD se han asociado a diversos cambios patológicos en el tejido, los cambios de AD y RD se han asociado con daño axonal y desmielinización principalmente en estudios con modelos animales [10].

Mediante el tensor de difusión es posible generar una representación de las fibras de la SB o tractografía. La tractografía utiliza la dirección de máxima difusividad entre vóxeles cercanos para trazar las diferentes conexiones que componen la red cerebral. Sin embargo, esta aproximación es muy simplista dado que el modelo por DTI no es capaz de descomponer las diferentes fibras contenidas en un solo vóxel. La estructura local de la SB presenta regiones de cruce, dobleces o dispersión de fibras en más del 90 % de los vóxeles [11]. El uso de modelos avanzados de tractografía proporciona una representación de mayor resolución angular permitiendo diferenciar diferentes máximos locales dentro de un único vóxel [12]. Gracias a esto se puede descomponer las fibras de SB en diferentes direcciones en una región de cruce de fibras y reconstruir aquellos tractos que presentan una gran curvatura y una FA baja [13].

La utilización de modelos avanzados de tractografía permite generar fibras de SB biológicamente más precisas respecto a la anatomía subyacente y trazar las trayectorias que componen la red cerebral [14]. Una vez generada la tractografía es posible cuantificar la reconstrucción con diferentes índices como pueden ser el número de fibras, el volumen lesional de las conexiones o las medidas del tensor. Esto puede facilitar el uso de medidas sensibles para diferenciar entre un cerebro sano y otro con una patología. Además de la posibilidad de detectar qué conexiones del cerebro (subsistemas) guardan algún tipo de relación con variables clínicas de interés [15].

A través de la cuantificación de este tipo de medidas sensibles a la microestructura tisular se pueden relacionar cambios patológicos de la SBAN sensibles al deterioro cognitivo en pacientes con EM [16] [15]. Otra herramienta útil para la detección de anomalías de la red cerebral es la teoría de grafos [17]. Esta metodología permite caracterizar varios aspectos de la estructura de la red, designando las distintas regiones de interés (SG) como nodos de un grafo y las conexiones entre estos nodos como aristas (SB). Este planteamiento ha facilitado la comprensión de la arquitectura de la red cerebral. Basándose en estos métodos, estudios recientes han demostrado que el cerebro no puede ser considerado simplemente como una gran red interconectada, sino más bien una colección jerárquica de redes de ámbito local que cooperan paralelamente y son capaces de optimizar información por medio de diferentes estructuras modulares [18]. El uso de estas herramientas de análisis en pacientes con EM ha demostrado que la red cerebral de estos pacientes tiene una menor capacidad para intercambiar y procesar información eficazmente. Además, la existencia de mecanismos de desconexión ha sido descrita como una causa importante en la manifestación de la alteración cognitiva en la EM [19] [15] [20] [21] [22]. Sin embargo, a pesar de estos importantes hallazgos, hay que tener en cuenta otros factores que pueden relacionarse con el rendimiento cognitivo como el volumen lesional y el daño tisular local de la SB [23] [24].

El presente trabajo se basa en el desarrollo de herramientas para crear un modelo de aprendizaje automático para predecir el rendimiento cognitivo a través del estudio de la conectividad estructural en pacientes con EM. El aprendizaje automático forma parte de la ciencia computacional capaz de generalizar comportamientos implícitos en base a los datos introducidos haciendo que “aprendan” automáticamente. La tarea de clasificación

es una de las más importantes entre todo el conjunto de algoritmos que engloban las técnicas de aprendizaje automático. El uso de algoritmos supervisados, se caracteriza por la búsqueda de un modelo capaz de asignar un conjunto de clases, o etiquetas, basándose en la información contenida en los datos. Estas técnicas de aprendizaje automático han ayudado a numerosos estudios a obtener resultados muy positivos, como por ejemplo, a la hora de diferenciar entre pacientes con EM y sujetos sanos basándose únicamente en la información contenida en las imágenes de RM [25].

Capítulo 2

Motivación

Durante todo el proceso de aprendizaje del Máster en Ciencias de Datos me han atraído principalmente las materias relacionadas con el aprendizaje automático y cómo estos algoritmos pueden ayudar a transformar los numerosos datos en conocimiento para nuestra sociedad.

Concretamente, la investigación médica se mueve en un ámbito global donde se generan millones de datos diariamente sobre pacientes afectados por alguna dolencia o enfermedad. Este escenario es propicio para la aplicación de técnicas de aprendizaje automático ya que su empleo puede ayudar a analizar una multitud de datos conjuntamente y ser capaz de detectar patrones imperceptibles para el ojo humano.

Gracias a la colaboración en la investigación junto con el grupo de Imagen Avanzada en enfermedades Neuroinmunológicas (ImaginEM) sobre esclerosis múltiple del Hospital Clínic tengo la posibilidad de poder integrar mi experiencia en las tecnologías de programación y en la ingeniería del software con los nuevos conocimientos adquiridos en el Máster en un marco hospitalario.

Capítulo 3

Objetivo

El propósito de este estudio es intentar predecir el rendimiento cognitivo del paciente con EM a través de la cuantificación del volumen lesional y la microestructura de la red cerebral empleando herramientas de aprendizaje automático.

Partiendo de matrices de conectividad estructural, calculadas mediante la estimación del volumen de lesión y de los índices del tensor en cada una de conexiones cerebrales, se estudiará cómo difieren los pacientes con un mayor o menor rendimiento cognitivo. Esto será útil para predecir en qué fase de afectación de la red cerebral se desarrolla una posible disfunción cognitiva por parte del paciente con EM.

Capítulo 4

Estado del arte

Las técnicas de aprendizaje automático ha aportado grandes avances en estudios relacionados con la estructura cerebral y la neuroimagen. Por ejemplo, [25] el uso de la herramienta "entropía de ondas estacionarias" (SWE) permitió extraer aquellos atributos más característicos de las imágenes cerebrales. Posteriormente la aplicación del algoritmo de los k-vecinos más cercanos (K-NN), consiguió una clasificación del 97.94 % de exactitud diferenciando sujetos sanos y pacientes con EM. Otros estudios, basándose en técnicas de aprendizaje automático, consiguieron localizar regiones de interés (ROI) partiendo de imágenes de RM [26].

Si bien la aplicación de redes neuronales artificiales (ANN) para el análisis de imágenes médicas está muy extendido, su uso en aplicaciones neurológicas está aún en desarrollo [27] [28]. A través de algoritmos más complejos de aprendizaje automático, "deep learning" o aprendizaje profundo, se han obtenido resultados muy prometedores en este campo. Por ejemplo, usando una red neuronal convolucional (CNN) con estructura LeNet-5, se ha logrado clasificar satisfactoriamente enfermos de Alzheimer con una exactitud de 96.85 % a partir de imágenes por resonancia magnética funcional (fRM). Es más, con este método también se consiguió predecir los estados de la enfermedad para los diferentes grupos de edades [29]. También otros estudios relacionados con EM han demostrado la eficacia de este tipo de algoritmos. Por ejemplo, el uso conjunto de "Retricted Bolzmann Machines" (RBM) y "Deep Belief Network" (DBN) ha demostrado ser una herramienta eficaz para la creación de características nuevas tan eficaces como las obtenidas directamente con los datos [30].

Teniendo esto presente, se han creado herramientas específicas para las aplicaciones neurológicas. Por ejemplo, ann4brains [31] proporciona redes neuronales concebidas especialmente para el uso de matrices de conectividad.

Destacar también que la mayoría de los estudios que aplican el aprendizaje automático a datos del conectoma se han centrado en predecir los resultados a través de la clasificación o regresión. Además, el uso del aprendizaje automático ha permitido identificar subredes y regiones de interés a través de un análisis no supervisado.

También está muy extendido en la actualidad el uso de algoritmos relacionados con máquinas de vector de soporte (SVM) [32] ya que ofrecen una mejor precisión de predicción y es menos sensible al ruido. Por ejemplo, [33] se ha conseguido alcanzar una exactitud del 95 % para distinguir entre una cohorte de sujetos sanos y otra de pacientes afectados por depresión a través de imágenes de fRM.

En el ámbito de la neurociencia computacional los datos utilizados como entrada suelen tener una relación entre el número de ejemplares y la dimensión de estos bastante desfavorable. Para hacer factible la tarea del aprendizaje en muchos de estos estudios, la dimensionalidad de los datos debe reducirse significativamente para poder extraer las características más relevantes. Los algoritmos no supervisados más usados para este propósito son "Principal Component Analysis" (PCA) y "Independent Component Analysis" (ICA), entre los supervisados destaca "Common Spatial Pattern" (CSP) [34].

Capítulo 5

Metodología

Teniendo presente el objetivo principal de este trabajo, la investigación se divide en tres partes principales: los datos, los algoritmos y los resultados obtenidos.

Respecto a los datos disponibles para la investigación, se ha estudiado en detalle el origen de éstos, cómo son calculados y el significado de cada uno de ellos. También fue necesario estudiar cómo están presentados, la distribución de los atributos y su calidad para su integración en el sistema de aprendizaje automático.

La metodología seguida durante la selección de algoritmos para la obtención de los modelos sigue un carácter iterativo sobre los algoritmos de aprendizaje automático. Empezando por los algoritmos obtenidos durante la fase del estudio del arte e incorporándose a posteriori nuevos algoritmos, se han aplicado los datos y optimizados los hiperparámetros que configuran a cada uno de ellos. En cada iteración se han probado fases nuevas para el preprocesado de los datos y nuevas configuraciones para la ejecución de modo que los resultados se han ido obteniendo formaran parte de la heurística aprendida para retomar nuevamente el proceso iterativo.

Finalmente, se han expuesto en un informe autogenerado durante el proceso experimentos con sus configuraciones usadas, el proceso de obtención de los modelos y resultados obtenidos durante la ejecución.

Como herramienta principal se ha usado el lenguaje de programación interpretado Python para la implementación de software necesario para la obtención y adecuación datos proveídos. Todo el trabajo realizado para la ejecución y selección de los algoritmos junto con sus hiperparámetros y elementos de preprocesado se ha realizado dentro del framework Scikit-learn [35] y con la ayuda de otros frameworks y librerías dedicados al aprendizaje automático; Keras [36], TensorFlow [37], Caffe2 [38] y, enfocado a la neurología, ann4brains [31]. También fue necesario el uso de las librerías científicas para el tratamiento de datos; Numpy [39], SciPy [40] y Pandas [41]. Con ayuda de librerías gráficas de Python como seaborn [42] y matplotlib [43] se generan las gráficas y representaciones visuales de los datos.

La documentación de las ejecuciones se exponen usando notebooks de Jupyter [44] donde se presentan los resultados y ejemplos prácticos del estudio. Como plataforma de edición del contenido se ha usado Google Drive y LaTeX[45], dentro de la plataforma Overleaf [46] como lenguaje de edición para la documentación final.

Todo el código generado durante la investigación se ha publicado bajo licencia Open Source GPLv3. Para la

documentación estática se ha usado la licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de CreativeCommons (CC BY-NC-SA 3.0 ES). Destacar que los datos proporcionados no se pueden exponer para garantizar su privacidad y confidencialidad.

Capítulo 6

Plan de investigación

La planificación de este Trabajo Fin de Máster se ha diseñado para que corresponda con el calendario de entregas propuesto por la Universidad Oberta de Catalunya.

En la gráfica de Gantt 6.1 se expone el calendario de actividades planeadas desde el comienzo de esta investigación. En color azul se han marcado las actividades relacionadas con la investigación y el estudio teórico. Las tareas relacionadas con la implementación y cuestiones técnicas aparecen en color rosado y las actividades documentales y de divulgación en amarillo.

Respecto a la planificación inicial sólo se han eliminado la tarea de "Bagging, boosting and stacking". El tiempo asignado a éste se se ha redistribuido entre las tareas "Optimización de los datos" y "Aplicación de Modelos". En la imagen 6.2 se muestra gráficamente cómo fueron reorganizadas las tareas.

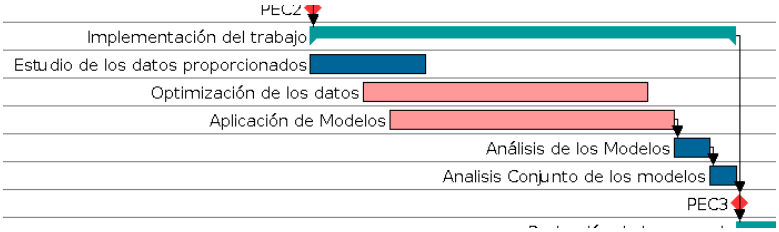

Figura 6.2: Modificación realizada en el esquema de Gantt

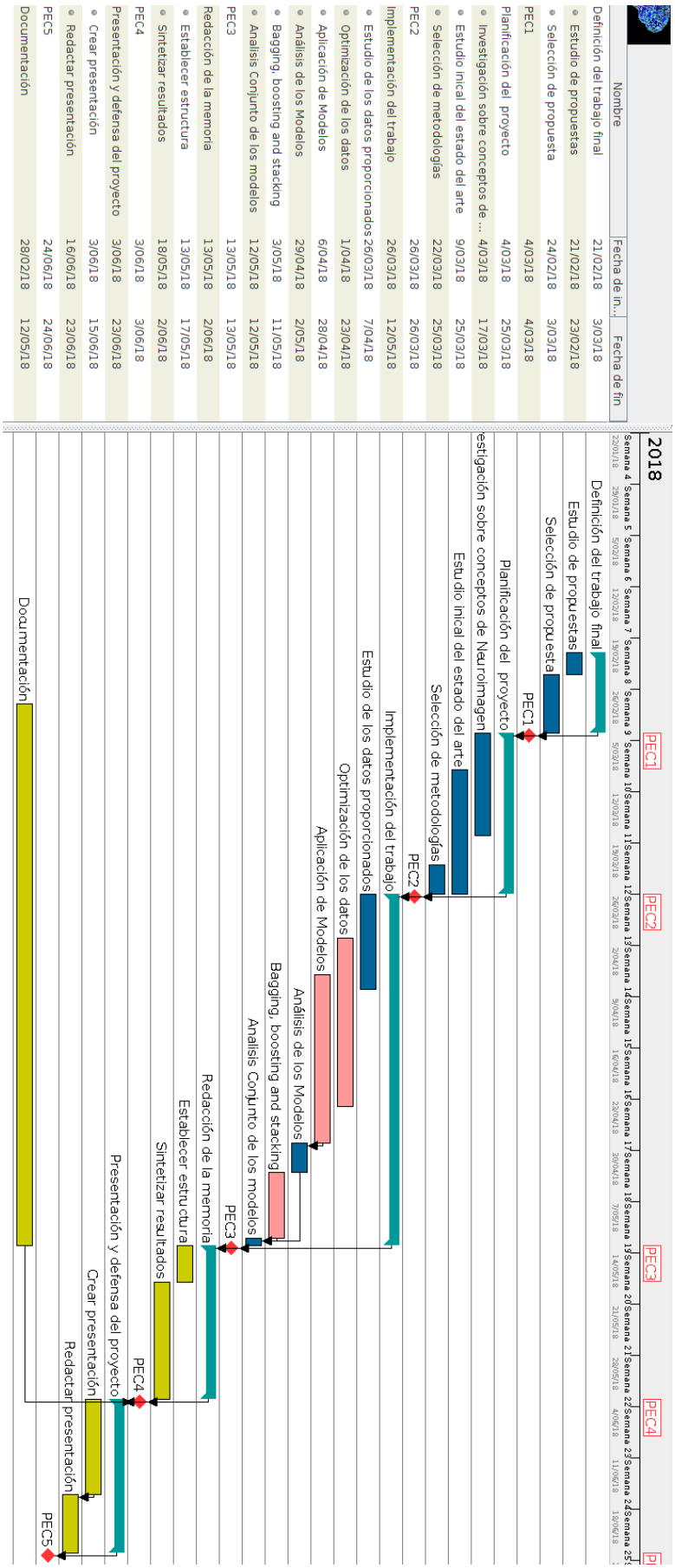

Figura 6.1: Planificación inicial en el esquema de Gantt

Parte II

Desarrollo

Capítulo 7

Desarrollo del trabajo

Los métodos del aprendizaje automático pueden ayudar a las deficiencias de la interpretación humana en relación con la complejidad del cerebro [34]. Más concretamente, usando algoritmos supervisados de clasificación predecir la disfunción cognitiva en pacientes de EM

Teniendo siempre presente esta meta, se ha llevado a cabo un estudio preliminar de los datos proporcionados (matrices de conectividad basadas en la integridad microestructural y volumen lesional). Posteriormente se implementó un sistema para la limpieza y preparación de los datos para la aplicación a los algoritmos de aprendizaje automático. Con especial énfasis en descartar modelos que sufrieran sobreajuste y usando de métodos de validación cruzada y remuestreo se han validado los resultados obtenidos siguiendo métricas de exactitud.

Capítulo 8

Estudio de los datos

A través de un convenio de cooperación entre la Universidad Oberta de Catalunya y el Institut d'investigacions Biomèdiques August Pi i Sunyer (IDIBAPS), contamos con datos clínicos y de neuroimagen avanzada de personas que padecen esclerosis múltiple junto con un grupo control.

Como información clave para este estudio destacan las matrices simétricas de conectividad estructural. Estas matrices son el resultado del procesado de imágenes de dRM. Los índices obtenidos a través de estas técnicas, tales como las medidas del tensor, el número de fibras y el volumen de lesión pueden facilitar la obtención de medidas sensibles para diferenciar entre un paciente con un mejor o peor rendimiento cognitivo:

DTI-FA Anisotropía fraccional

DTI-MD Difusividad media

DTI-L1 Difusividad axial

DTI-RX Difusividad radial

RAW Número de fibras

LS Volumen lesional

Los pacientes presentan un peor rendimiento cognitivo principalmente en atención y memoria. Para detectar una posible discapacidad cognitiva de los pacientes hemos realizado una evaluación neuropsicológica. Esta evaluación incluye una batería BRBN (Brief Repeatable Battery, [47]) junto con otras prueba cognitivas. Los dominios estudiados han sido los relacionados con aprendizaje y memoria. En la siguiente tabla podemos observar el número de pacientes cuya discapacidad está por debajo de puntuaciones -1 y -2 desviaciones estándares de una población de referencia (ver tabla 8).

Matriz/ Grupo	Todos	HV (Sanos)	EM sim problema	Atención	Atención y memoria	Memoria
DTI_FA	140	42	49	14	16	19
DTI_L1	140	42	49	14	16	19
DTI_MD	140	42	49	14	16	19
DTI_RX	140	42	49	14	16	19
RAW	211	42	90	29	21	29
LS	169	0	90	29	21	29
TOTAL	255	42	101	29	21	32

Cuadro 8.1: Tabla con la distribución según clases cognitivas

8.1. Distribución de los datos

La información proveída por el grupo de Imagen Avanzada en enfermedades Neuroinmunológicas (ImaginEM) [48] se encuentra organizada en una hoja de cálculo donde se presentan los datos clínicos de cada paciente anonimizado y una estructura en forma de distintos directorios donde se presentan cada matriz de adyacencia.

8.1.1. Hoja de cálculo (Indice principal)

La hoja del cálculo con los datos de cada paciente consta de 255 filas, una por paciente y 33 columnas. Entre ellas se encuentra la columna *profile* con los siguientes valores:

none Persona con EM sin deficiencia cognitiva detectada.

HV Persona sin EM

memory Persona con EM y afectación en memoria

attention Persona con EM y afectación en atención.

mem_att Persona con EM y deterioro cognitivo en memoria y atención.

8.1.2. Directorios con matrices

Estos directorios siguen un formato establecido donde cada fichero indica en su prefijo a qué paciente corresponde y, en función del directorio donde se encuentra, la medición de dónde proviene la matriz. Por ejemplo, el fichero situado en

```
FIS/ADJACENCY_MATRICES_GRAPH/DTI_indices/FA/FIS_001_FA.csv
```

corresponde con la medida DTI-FA del paciente de cuyo identificador es FIS_001. En la tabla de la imagen 8.1 se describe detalladamente esta estructura.

Cada fichero que contiene los valores de una matriz está almacenado en formato CSV conteniendo todos ellos 76 filas y 76 columnas sin ningún valor nulo o vacío.

Rutas	Posfijo	Medida
CONTROLS/ADJACENCY_MATRICES_GRAPH/DTI_indices/FA FIS/ADJACENCY_MATRICES_GRAPH/DTI_indices/FA PREDICTORS/ADJACENCY_MATRICES_GRAPH/DTI_indices/FA REESCAN/ADJACENCY_MATRICES_GRAPH/DTI_indices/FA	_FA_factor.csv	DTI_FA
CONTROLS/ADJACENCY_MATRICES_GRAPH/DTI_indices/L1 FIS/ADJACENCY_MATRICES_GRAPH/DTI_indices/L1 PREDICTORS/ADJACENCY_MATRICES_GRAPH/DTI_indices/L1 REESCAN/ADJACENCY_MATRICES_GRAPH/DTI_indices/L1	_L1_factor.csv	DTI_L1
CONTROLS/ADJACENCY_MATRICES_GRAPH/DTI_indices/MD FIS/ADJACENCY_MATRICES_GRAPH/DTI_indices/MD PREDICTORS/ADJACENCY_MATRICES_GRAPH/DTI_indices/MD REESCAN/ADJACENCY_MATRICES_GRAPH/DTI_indices/MD	_MD_factor.csv	DTI_MD
CONTROLS/ADJACENCY_MATRICES_GRAPH/DTI_indices/RX FIS/ADJACENCY_MATRICES_GRAPH/DTI_indices/RX PREDICTORS/ADJACENCY_MATRICES_GRAPH/DTI_indices/RX REESCAN/ADJACENCY_MATRICES_GRAPH/DTI_indices/RX	_RX_factor.csv	DTI_RX
CONTROLS/ADJACENCY_MATRICES_GRAPH/RAW FIS/ADJACENCY_MATRICES_GRAPH/RAW PREDICTORS/ADJACENCY_MATRICES_GRAPH/RAW REESCAN/ADJACENCY_MATRICES_GRAPH/RAW	_weight_factor.csv	RAW
FIS/ADJACENCY_MATRICES_GRAPH/LS PREDICTORS/ADJACENCY_MATRICES_GRAPH/LS REESCAN/ADJACENCY_MATRICES_GRAPH/LS	_matrix_LS.csv	LS

Figura 8.1: Imagen con la tabla de los directorios de matrices y sus medidas correspondientes

8.2. Características de los datos

Para cada instancia que disponemos de los datos, contamos con seis matrices de adyacencia de tamaño $76x76$. Aparte de estas matrices los valores de estas matrices, de la hoja de cálculo se presentan 33 columnas. Es decir, disponemos ejemplares de la muestra con hasta $76x76x6 + 33 = 34689$ atributos cada uno.

Los valores de las matrices son completos, es decir, si una instancia tiene una matriz, todos sus valores están presentes. Como se ve en la tabla 8, no todas las instancias tienen todas sus matrices. Por otra parte, también se encuentran 40 matrices "perdidas", sin correspondencia con ninguna instancia de la hoja de cálculo y, por lo tanto, descartadas para el estudio.

Capítulo 9

Software para el preprocesado

El preprocesamiento de los datos es la etapa encargada de la limpieza de los datos, la integración en el sistema de aprendizaje automático, la transformación y reducción para los algoritmos de aprendizaje automático. Después de esta fase, los datos deben de ofrecerse a través de una fuente consistente y adecuada para la aplicación de los diferentes algoritmos de aprendizaje automático.

Con esta finalidad se ha desarrollado el software NeuLoadData. Usando como origen los archivos contenidos en los directorios anteriormente descritos y su correspondiente hoja de cálculo es capaz de limpiar, validar e integrar los datos para su posterior uso.

9.1. NeuLoadData

Este software ha sido concebido especialmente para los datos en el formato proporcionado por IDIBAPS para este estudio. Aún así, su implementación se ha centrado en un enfoque genérico para su posible adaptación a nuevos orígenes de datos o necesidades futuras.

Usando Python como lenguaje de programación se ha implementado esta herramienta y publicado para la comunidad bajo licencia Open Source en la plataforma de desarrollo colaborativo Github [49] bajo la dirección `https://github.com/efrain70/NeuDataLoad`.

9.2. Metodología de desarrollo

La implementación se ha ejecutado siguiendo una metodología iterativa basada en los principios del Manifiesto por el Desarrollo Ágil de Software [50]. Principalmente, a medida que se aparecían las necesidades sobre el conjunto de datos, se han añadido nuevas funcionalidades completas y dispuesto para su utilización.

9.3. Aseguramiento de la calidad

La calidad de este software es clave para una correcta utilización de los datos en todo el proceso de investigación. Cualquier error en el preprocesado podría dar a valores inesperados en los resultados o condicionar la investigación erróneamente.

Con este propósito, se ha desarrollado todo este software validando todo su funcionamiento a través de pruebas unitarias y de integración. Para ello se ha usado la herramienta py.test [51] junto con la plataforma Coveralls [52] (dirección del proyecto `https://coveralls.io/github/efrain70/NeuDataLoad` imagen 9.1) para controlar que la cobertura del código validado con estas pruebas siempre es total, 100 %.

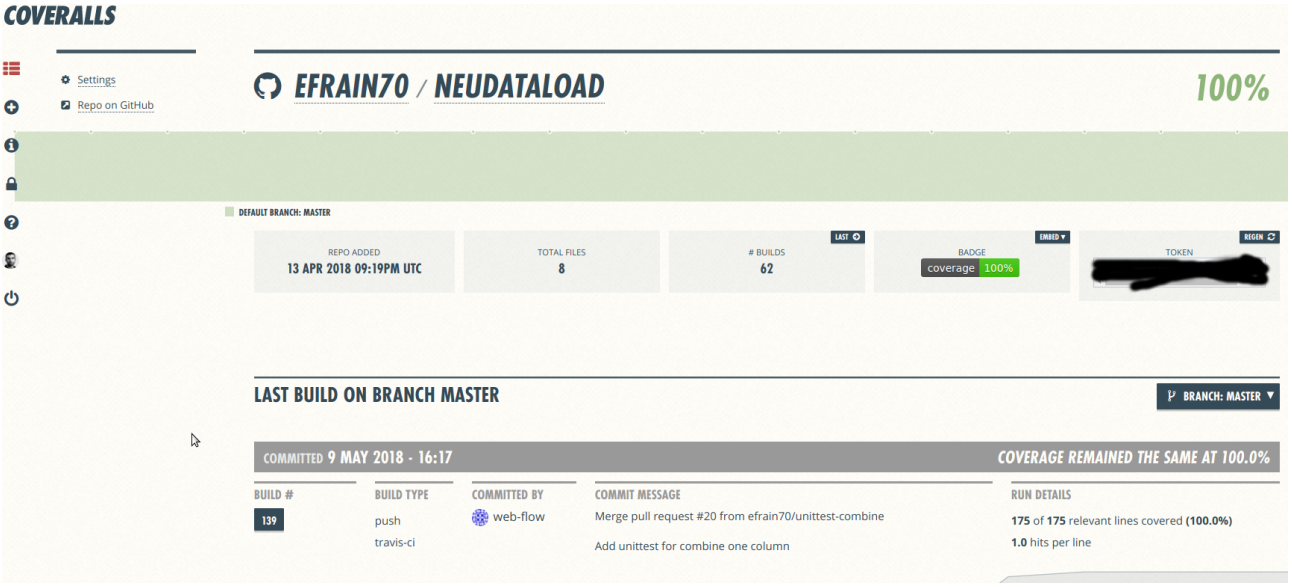

Figura 9.1: Captura de pantalla del proyecto NeuLoadData en Coveralls

No solamente se ha comprobado que el código funciona correctamente, sino que también su sintaxis se ajusta con los estándares del lenguaje. Se ha validado el estilo de todo el código bajo la guía especial para Python PEP8 [53] y para la documentación en código PEP257 [54] y controlado la complejidad del mismo.

Todas estas validaciones de la calidad del software se ejecutan en un entorno ágil, configurado el proceso siguiendo un enfoque de integración continua, usando los "pipelines" de la plataforma Travis CI [55] (Dirección del proyecto `https://travis-ci.org/efrain70/NeuDataLoad`, imagen 9.2) como plataforma coordinada con el repositorio Github. Es decir, cada vez que se publique alguna modificación en el código, Github se coordina con Travis CI para la ejecución de todas las pruebas anteriormente descritas reportando los posibles errores automáticamente.

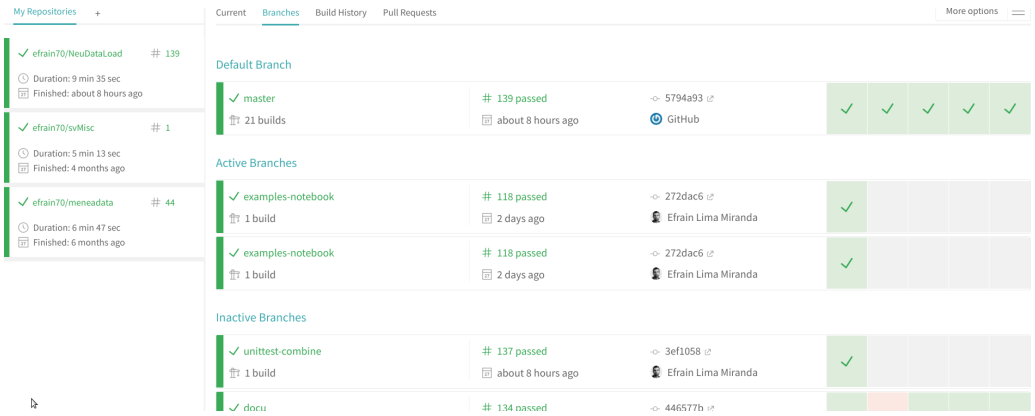

Figura 9.2: Captura de pantalla de la ejecución en Travis CI del proyecto NeuLoadData

9.4. Documentación

Toda la documentación de este software se ha descrito usando la tecnología propia del lenguaje de Python. Gracias a la utilidad de Sphinx [56] se han generado automáticamente la documentación con el contenido descrito en el código. Toda esta documentación del software se ha incluido en el código y generado en formato web para su publicación en Github Pages [57]. Bajo la dirección `https://efrain70.github.io/NeuDataLoad/`, imagen 9.3, se encuentra la interfaz de la aplicación detallada.

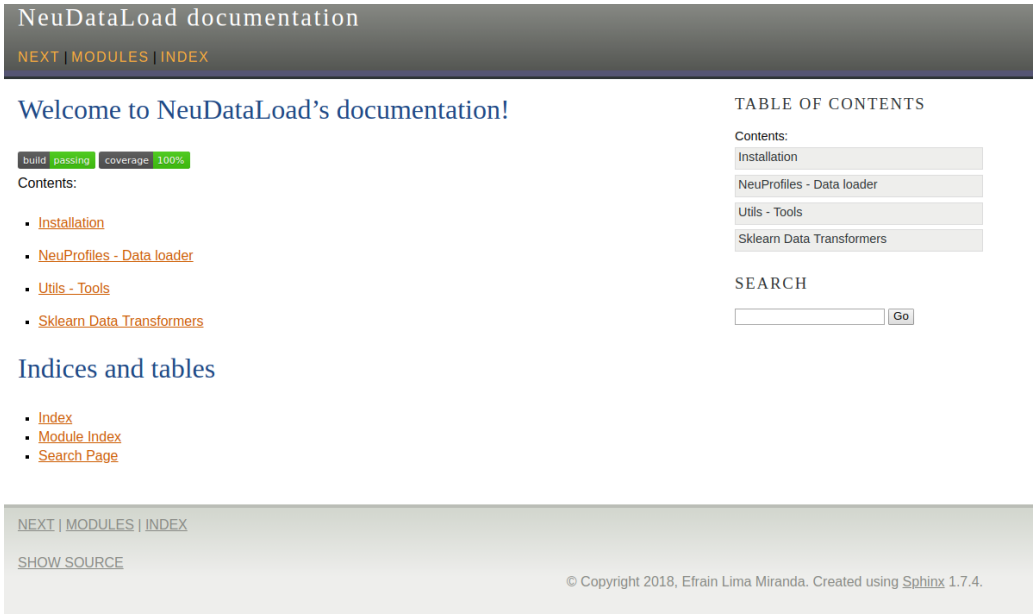

Figura 9.3: Captura de pantalla de la documentación en linea de NeuLoadData

9.5. Funcionalidades principales

9.5.1. Aplicación de transformaciones a las matrices

Para adaptarse a las necesidades de los algoritmos de aprendizaje automático y para mejorar su rendimiento se incluyen un conjunto de herramientas para la transformación de las matrices:

Combinación de las matrices: Teniendo como entrada un conjunto de matrices del mismo tamaño NxM, se le aplica una función a las celdas de todas las matrices situadas en las mismas coordenadas i,j dando como resultado una nueva de tamaño NxM con los resultados de la aplicación de la función.

Aplanamiento o remodelación: A partir de una matriz de tamaño NxM, se generan una nueva de una única dimensión con los valores extendidos. Es decir, obtendremos $N\Delta M$ atributos nuevos.

Binarización: A partir de un valor de frontera, la matriz convierte todos sus valores a 1 y 0 dependiendo si éstos superan el valor de frontera. Se obtiene una matriz con valores binarios de tamaño NxM.

Selección de atributos: Selección de los atributos del conjunto de datos y de los posibles nuevos atributos generados durante la transformación aplanamiento.

Todas estas funcionalidades se implementan principalmente en el módulo 'utils', pero puede ser usada a través de la clase NeuProfiles directamente o por medio de transformadores de pipelines de scikit-learn.

Todas estas funcionalidades se implementan principalmente en el módulo 'utils', pero puede ser usada a través de la clase NeuProfiles directamente o por medio de transformadores de pipelines de scikit-learn [35]. En el módulo transformer están las clases transformadoras (https://efrain70.github.io/NeuDataLoad/transformer.html) que facilitan la integración en el framefork scikit-learn.

9.5.2. Carga automática y validación de las matrices

Usando la ruta local o remota definida por el usuario, NeuLoadData valida las matrices leídas de los ficheros y las incluye en un único dataframe, contenedor de toda la información de la hoja de cálculos y las matrices de adyacencia.

NeuProfiles es clase que implementa esta funcionalidad a través de su método load, dando como resultado objeto contenedor dataframe de Pandas [41] conjuntamente con las utilidades para la transformaciones de matrices. (https://efrain70.github.io/NeuDataLoad/profiles.html)

Capítulo 10

Algoritmos de aprendizaje automático

El aprendizaje automático es una rama de la inteligencia artificial cuyo objetivo es proporcionar técnicas para hacer que los sistemas "aprendan". Este aprendizaje se basa en algoritmos que, a partir de un conjunto de datos, son capaces de crear un modelo capaz de generalizar comportamientos y reconocer patrones. Estos modelos se caracterizan por su finalidad y se clasifican en dos grupos diferenciados: Algoritmos supervisados y algoritmos no supervisados

10.1. Tipos de tareas

En función de la tarea que queramos resolver con el aprendizaje automático podemos diferenciar en tres grupos: clasificación, regresión y agrupamiento. Todas siguen un paradigma inductivo donde a partir de los datos y un modelo se puede obtener un nuevo conocimiento que puede ser aplicado posteriormente a nuevos datos. A su vez, éstos se diferencian en dos grandes grupos, algoritmos supervisados y no supervisados.

10.1.1. Clasificación

La tarea de clasificación se centra en asignar un conjunto de clases a instancias de un dominio compuesto por atributos discretos o continuos. Estas clases, o etiquetas, tiene que ser conocidas para un subconjunto para la obtención del modelo.

10.1.2. Regresión

La tarea de regresión la podemos describir como una clasificación con clases continuas. Es decir, asigna un valor numérico a instancias de un dominio compuesto por atributos discretos o continuos. Este modelo está definido por una función, generalmente desconocida, que establece un valor numérico para los nuevos datos.

10.1.3. Agrupamiento

El agrupamiento, al igual que la clasificación y la regresión, es una tarea inductiva. En cambio, se considera un algoritmo no supervisado ya que no se dispone de un conjunto de clases para predecir. El resultado del agrupamiento es un conjunto de clases y la asignación de cada elemento del conjunto de datos a una de estas clases basándose en la similitud entre las instancias. El modelo que se obtiene como resultado del agrupamiento también asigna a cada nueva instancia una clase de las anteriormente obtenidas.

10.2. Algoritmos supervisados y no supervisados

10.2.1. Algoritmos supervisados

Estos algoritmos requieren un conjunto de datos etiquetados. Si las etiquetas de los datos son categóricos se denominan algoritmos de clasificación. En cambio, si son etiquetas numéricas corresponden a algoritmos de regresión. Por lo tanto, son usados tanto en tareas de clasificación como de regresión.

10.2.2. Algoritmos no supervisados

Estos algoritmos no necesitan que los datos dispongan de ninguna etiqueta o clasificación previa. Se centran en el agrupamiento o segmentación con el fin de encontrar características similares. Podemos diferenciar entre dos grupos; los métodos jerárquicos donde se obtiene una organización con varios niveles de agrupación, y los métodos particionales o no jerárquicos.

Capítulo 11

Experimentos

En este trabajo nos hemos centrado principalmente en la aplicación de algoritmos de clasificación de aprendizaje supervisado para las etiquetas relacionadas con las disfunciones cognitivas presentes en los datos proporcionados. Los experimentos que se han llevado a cabo persiguen el objetivo de obtener el mejor modelo para cada algoritmos a partir de un conjunto de datos usados para su entrenamiento.

11.1. Obtención de los modelos

Los modelos de aprendizaje automático se generan a través de la implementación de un algoritmo junto a sus parámetros que lo configuran aplicado a un subconjunto de atributos. Por lo tanto, hay que elegir el algoritmo, sus parámetros y los atributos de los datos para poder describir cada modelo. A su vez, un modelo puede estar compuesto de varios pasos configurables que transforman los datos organizados en fases consecutivas.

11.2. Algoritmos

Dentro de la amplia familia de los algoritmos de clasificación de aprendizaje supervisado nos encontramos multitud de algoritmos con sus variantes donde ninguno es, a priori, más adecuado que otro: "no free lunch theorem" [58]. Centrándonos en las implementaciones proporcionadas por los frameworks de aprendizaje automático scikit-learn [35] y Keras [36], se han realizado una valoración inicial de los algoritmos seleccionando finalmente "Logistic Regression", "Support Vector Machine", "Gaussian Naive Bayes", "Random Forest Classifier" y "Artificial Neural Network".

11.2.1. Logistic Regression

Usado extensamente en las ciencias médicas y sociales, este algoritmo se basa en una clasificación lineal donde las probabilidades que describen los posibles resultados de un único ensayo se modelan utilizando una

función logística. En la imagen 11.1 se puede ver un ejemplo gráfico de su aplicación.

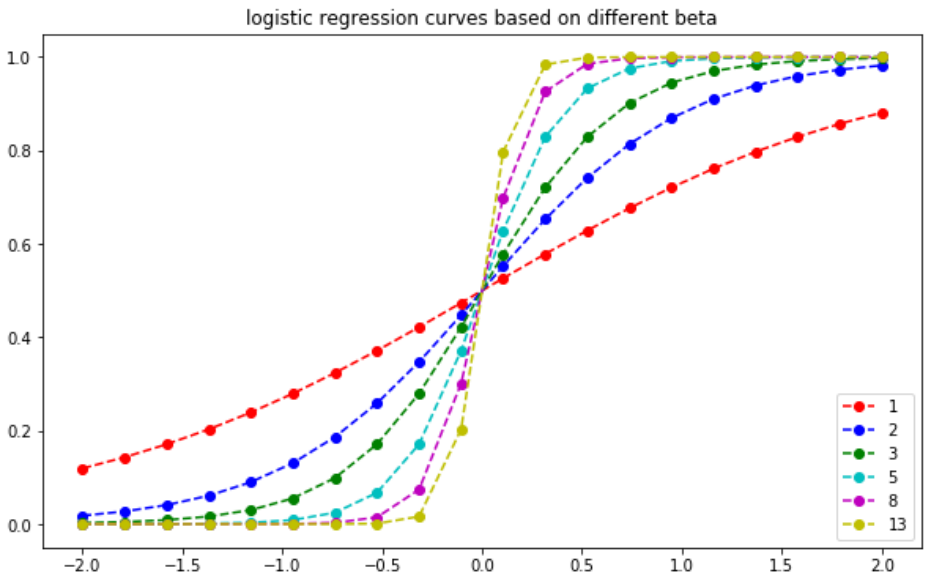

Figura 11.1: Algoritmo Logistic Regression [1]

11.2.2. Support Vector Machine

Como se puede consultar en [59] SVM es un algoritmo de aprendizaje supervisado muy utilizado para investigaciones relacionadas con la neuroimagen. Tiene la característica de ser muy eficiente cuando el número de atributos es alto ya que está basado en la minimización del riesgo estructural. Este algoritmo representa las instancias de la muestra como puntos en el espacio el cual es dividido por un hiperplano. En base a esta división se establecen las etiquetas a los valores (figura 11.2).

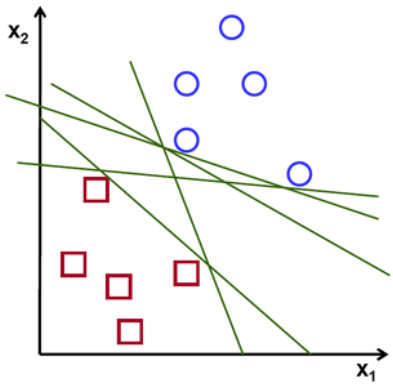

Figura 11.2: Ejemplo algoritmo SVM [2]

11.2.3. Gaussian Naive Bayes

Este modelo es clasificador probabilístico fundamentado en el teorema de Bayes, supone que la presencia o ausencia de una característica particular no está relacionada con la presencia o ausencia de cualquier otra característica dada la clase variable [60].

11.2.4. Random Forest Classifier

Modelo compuesto por la combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos [61].

11.2.5. Artificial Neural Network

Este modelo se caracteriza por su organización en capas de neuronas donde cada una propaga los resultados a la siguiente en función a unas reglas configuradas. Dependiendo del la configuración de estas reglas y capas define la complejidad del modelo. En la imagen 11.3 podemos ver un ejemplo con tres capas.

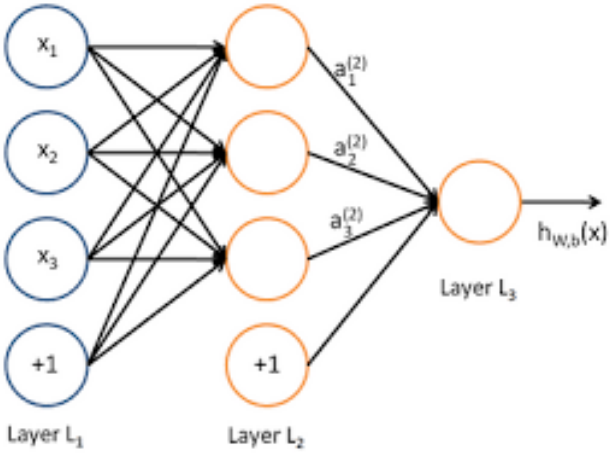

Figura 11.3: Ejemplo ANN con tres capas

11.3. Reducción de la dimensionalidad

En este trabajo nos encontramos con unos datos con muy alta dimensionalidad. Sin contar otros atributos disponibles en la hoja de cálculo, disponemos de 6 matrices con $76x76$ celdas cada una, es decir, más de 35.000 atributos por cada ejemplar. Para reducir esta dimensionalidad se han seguido los siguientes pasos durante los experimentos:

1. Selección conjunto de matrices para ser usadas como datos para los modelos a través de combinaciones, esto implica el descarte de matrices en su conjunto con todos sus valores.

2. Combinación de los valores de las matrices en una única. Por ejemplo, usar todas las matrices disponibles para las instancias para la creación de una única matriz con los valores de la media ponderada de las 6. Esta matriz sería la usada en el modelo.

3. Reducción de la dimensionalidad a través del algoritmo PCA.

11.4. Selección de los parámetros y atributos

Cada algoritmo tiene su propio conjunto de parámetros que lo configuran. A su vez, a cada uno se le pueden aplicar un conjunto de atributos pudiéndose obtener resultados muy diversos. Igualmente, cada paso en el preprocesado de los datos explicado con anterioridad tiene sus propios parámetros. Todo esto hace que la elección del modelo óptimo para un algoritmo establecido no es un problema trivial.

Dado el número de combinaciones posibles entre los parámetros de cada modelo, la búsqueda exhaustiva de todos ellos tiene un coste inasumible para este trabajo. A través de una selección aleatoria de combinaciones usando la utilidad de scikit-learn "Randomized Parameter Optimization" obtenemos una aproximación del modelo óptimo y de la configuración de los pasos previos.

Aunque los rangos para estas combinaciones no siempre son fáciles de estimar, el uso de herramientas gráficas ha proporcionado la respuesta de la mejor combinación que el algoritmo pueda utilizar.

Aunque los rangos para estas combinaciones no siempre son fáciles de estimar, el uso de herramientas gráficas ha proporcionado la respuesta de la mejor combinación que el algoritmo pueda utilizar. Las imágenes 11.4, y 11.5 muestran tres gráficas usadas para conocer las zonas donde las combinaciones de dos parámetros devuelven mejores resultados para los algoritmos Random Forest Classifier y Artificial Neural Network respectivamente.

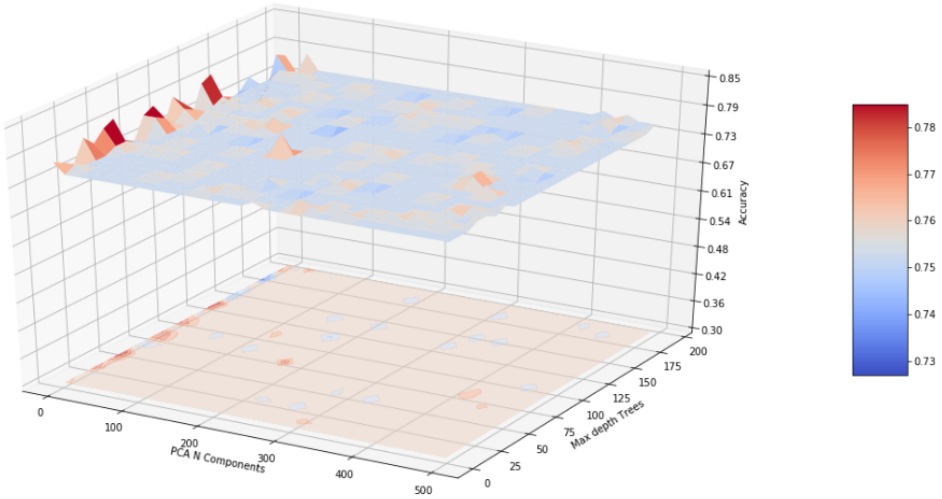

Figura 11.4: Gráfica para la selección de parámetros en Random Forest Classifier

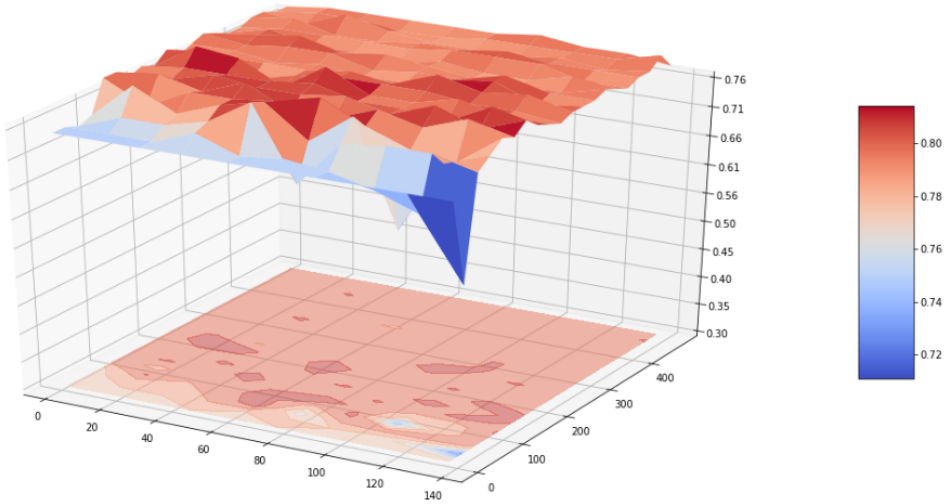

Figura 11.5: Gráfica para la selección de parámetros en Artificial Neural Network

11.5. Validación de los modelos

Tanto para comparar los posibles modelos obtenidos durante la selección de parámetros y atributos y para validar el modelo seleccionado para compararlo con otros se usa la implementación en scikit-learn de la técnica validación "K-Cross-Vaidation" (K-Fold) estratificada (Stratified K-Fold). La estratificación es muy importante para los algoritmos de clasificación con muestras con pocos ejemplares para conservar el porcentaje de elementos de cada clase en los conjuntos.

Esta técnica de validación descompone el conjunto inicial en dos, entrenamiento y test. El primero es usado para seleccionar el modelo. El conjunto de test para validar el modelo seleccionado, medir su exactitud y comprobar que no se produce sobreajuste (overfitting).

Durante la selección de los parámetros, el algoritmo "Randomized Parameter Optimization" realiza igualmente K-Fold estratificada sobre el conjunto de entrenamiento para comparar los candidatos entre ellos. Es decir, vuelve a dividir en cada combinación de parámetros establecida el conjunto de entrenamiento previamente obtenido. Como resultado se obtiene un conjunto de entrenamiento para el modelo y otro de test, que en este caso se suele llamar conjunto de validación. Este conjunto es usado para comprobar la bondad de la combinación de atributos que se está probando (ver imagen 11.6).

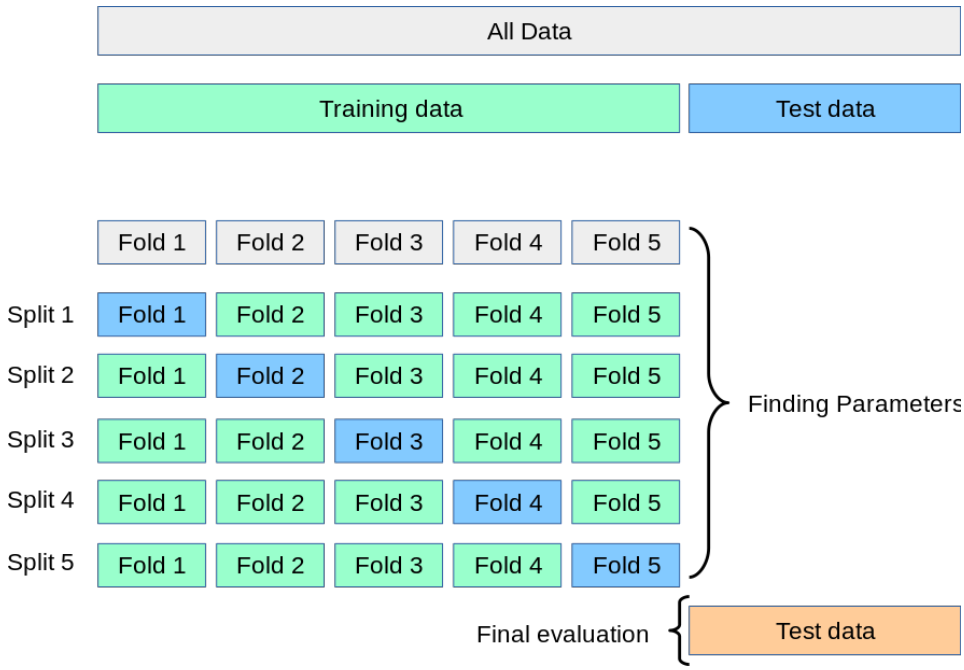

Figura 11.6: Ejemplo K-Fold para seleccionar el modelo y la evaluación final [3]

Capítulo 12

Ejecución de los experimentos

A continuación se describen detalladamente cómo se han ejecutado los experimentos para cada algoritmo, empleando el mismo esquema de ejecución.

Las ejecuciones realizadas en este trabajo de investigación han sido realizadas usando la herramienta Jypyter notebook [44]. Gracias a ésta se han publicado los resultados obtenidos durante de las ejecuciones de los experimentos junto con el código implementado para cada modelo.

En el repositorio Github de este trabajo bajo directorio 'examples' (https://github.com/efrain70/NeuDataLoad/tree/master/examples/notebooks) se ecuentran todas las ejecuciones realizadas para los modelos seleccionados.

12.1. Sistema para la ejecución

La ejecución de los experimentos se ha ejecutado en un ordenador personal, con Ubuntu 16.04.2 como sistema operativo con CUDA [62] instalado y la siguiente configuración de hardware:

- Tarjeta gráfica NVIDIA® GeForce® GTX 1050 (4 GB GDDR5 dedicados)
- Procesador Intel® Core™ i7-7700HQ (2.8, 3.8GHz, 6 MB Cache, 4 núcleos, 8 hijos de procesamiento)
- Memoria DDR4-2133 SDRAM, 16 GB (2 x 8 GB)
- Almacenamiento 1 TB HDD, 256 GB SSD

12.2. Carga de los datos

Este es el momento cuando se cargan todos los datos provenientes de los ficheros y la hoja de cálculo. En la imagen 12.1 podemos ver como la la salida de la ejecución muestra los errores al realizar la validación de los

datos y el código de su ejecución. En este caso concreto, se encuentran las 40 matrices que no disponen de su elemento correspondiente en el índice principal.

```
import os

import numpy as np
from neudataload import NeuProfiles

path = os.environ.get('NEUPATH')
filename = 'cognitive-profiles-FINAL.xlsx'

profiles = NeuProfiles(profiles_path=path, profiles_filename=filename)
profiles.load()

df = profiles.data_frame
```

```
WARNING:root:Index FIS_003, FIS_024, FIS_039, FIS_041, FIS_047, FIS_056, FIS_068, FIS_083, FIS_087, FIS_101, FIS_11
4, FIS_129, FIS_006, FIS_032, FIS_040, FIS_048, FIS_072, FIS_125, TTO_23, TTO_25, TTO_37, TTO_66, TTO_67, TTO_74, M
BP_05, ANM_05, AVM_05, AZR_05, CIL_05, FGC_05, JAG_05, JBB_05, JUI_05, MLM_05, MNN_05, SMS_05, TCR_05, TMS_05, VNS_
05, YGB_05 couldn't be found in main file (total 40 index(es)).
```

Figura 12.1: Código y resultado de ejecución para la carga de los datos.

12.3. Selección de los ejemplares

Como se aprecia en la tabla 8 hay ejemplares que no tienen algunas matrices de adyacencia. Por este motivo, se han repetido las ejecuciones de cada algoritmo dos veces; para los que tienen todas las matrices y para los que tienen todas las matrices relacionadas con DTI (FA, L1, MD y RX). El código para este caso se expone en la imagen 12.2.

```
column_matrixes = ['DTI_FA', 'DTI_L1', 'DTI_MD', 'DTI_RX', ]

# Matrixes witout NAN values
for column in column_matrixes:
    df = df[df[column].notna()]
```

Figura 12.2: Código para la selección de las matrices DTI.

12.4. Configuración de las etiquetas

Como se ha comentado en la sección 8.1, la clase binaria objetivo, memoria, se encuentra descrita en el atributo “profile” de la hoja del cálculo. En base a los valores, se establece su valor correspondiente usando el código de la imagen 12.3. Un ejemplo de cómo quedan estos valores se aprecia en la imagen 12.4.

```
from neudataload import get_multilabel

# Binarizing profile feature
groups = {'none': [],
          'HV': [],
          'memory': ['memory'],
          'attention': ['attention'],
          'mem_att': ['memory', 'attention']
          }

y_values = get_multilabel(df, 'profile', groups)
display(y_values.head())
```

Figura 12.3: Código para obtención de las etiquetas.

ID	attention	memory
FIS_001	0	0
FIS_002	0	0
FIS_004	0	0
FIS_005	0	0
FIS_007	1	0

(98, 2)

Figura 12.4: Resultado de las etiquetas de las primeras instancias.

12.5. Conjunto de entrenamiento y test

Con los valores de las instancias con sus respectivas etiquetas se aplica el algoritmo para la separación de los conjuntos de entrenamiento y test (ver 11.5). Se ha separado un 20 % de la muestra para el test y el resto para el entrenamiento (imagen 12.5).

```
# Training and test subsamples

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df, y, test_size=0.2, stratify=y)
```

Figura 12.5: Código para la aplicación de K-Fold.

12.6. Selección de la configuración

Cada algoritmo tiene su parámetros propios configurables y sus configuraciones para el preprocesado pueden ser distintas. Para definir cada una de ellas se han implementado "pipelines" que definen el flujo de ejecución. Estos son ejecutados usando nuevamente una separación de los conjuntos de entrenamiento y validación (ver sección 11.5). Otro punto importante es cómo se comparan los resultados de las configuraciones, en nuestro caso se ha optado por la más general, nivel de acierto de las etiquetas.

12.6.1. Logistic Regression

Para este algoritmo se han configurado los parámetros "penalty" y "C", imagen 12.6:

Penalty tipo de penalización.

C coeficiente de regulación.

```
# Parametrized values
N_FEATURES_OPTIONS = range(1, 50, 3)
PENALTY = ['l1', 'l2']
REGULATION_C = np.linspace(1, 100, 90)
MATRIXES = list(all_combinations(['DTI_FA', 'DTI_L1', 'DTI_MD', 'DTI_RX',]))
EXTRA_1D_COLUMS = [None] # + list(all_combinations(available_columns))



param_grid = {
        'combining__columns': MATRIXES,
        'filter__columns': EXTRA_1D_COLUMS,

        'reduce_dim__n_components': N_FEATURES_OPTIONS,
        'classify__penalty': PENALTY,
        'classify__C' : REGULATION_C,
}
```

Figura 12.6: Código para la selección de parámetros de Logistic Regression.

12.6.2. Support Vector Machine

Los parámetros para este algoritmo son, "degree" y "C", imagen 12.7:

Degree Grado del hiperplano.

C Grado de penalización por error.

```
# Parametrized values
N_FEATURES_OPTIONS = range(1, 500, 10)
PENALTY = ['l1', 'l2']
REGULATION_C = np.linspace(1, 500, 50)
MATRIXES = list(all_combinations(['DTI_FA', 'DTI_L1', 'DTI_MD', 'DTI_RX', ]))
EXTRA_1D_COLUMS = [None] # + list(all_combinations(available_columns))

DEGREE = range(2, 10)

param_grid = {
        'combining__columns': MATRIXES,
        'filter__columns': EXTRA_1D_COLUMS,

        'reduce_dim__n_components': N_FEATURES_OPTIONS,
        'classify__C' : REGULATION_C,
        'classify__degree': DEGREE,
}
```

Figura 12.7: Código para la selección de parámetros de Support Vector Machine.

12.6.3. Gaussian Naive Bayes

Para este algoritmo no se ha configurado ningún parámetro durante la búsqueda, imagen 12.8.

```
# Parametrized values
N_FEATURES_OPTIONS = range(1, 500, 10)
MATRIXES = list(all_combinations(column_matrixes))
EXTRA_1D_COLUMS = [None] # + list(all_combinations(available_columns))


param_grid = {
        'combining__columns': MATRIXES,
        'filter__columns': EXTRA_1D_COLUMS,

        'reduce_dim__n_components': N_FEATURES_OPTIONS,
}
```

Figura 12.8: Código para la selección de parámetros de Gaussian Naive Bayes.

12.6.4. Random Forest Classifier

El parámetro que configura este algoritmo es “max_depth”, imagen 12.9:

max_depth Profundidad máxima de los árboles.

```
# Parametrized values
N_FEATURES_OPTIONS = range(1, 500, 10)
MATRIXES = list(all_combinations(column_matrixes))
EXTRA_1D_COLUMS = [None] # + list(all_combinations(available_columns))

MAX_DEPTH = range(1, 100, 10)

param_grid = {
        'combining__columns': MATRIXES,
        'filter__columns': EXTRA_1D_COLUMS,

        'reduce_dim__n_components': N_FEATURES_OPTIONS,
        'classify__max_depth' : MAX_DEPTH,
}
```

Figura 12.9: Código para la selección de parámetros de Random Forest Classifier.

12.6.5. Artificial Neural Network

Este algoritmo puede ser el más difícil de configurar al ser el más complejo. Usando una red neuronal bastante simple con una única capa oculta. En la imagen 12.10 se puede ver cómo se ha diseñado y en 12.11 los parámetros:

n_hidden Número de células en la capa oculta.

learning_rate Grado de aprendizaje de la red neuronal.

epochs Épocas de entrenamiento de la red neuronal.

```
# Keras model

from keras.models import Sequential
from keras.layers import Dense, Activation, InputLayer
from keras.optimizers import SGD
from keras.wrappers.scikit_learn import KerasClassifier
from scipy import random
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from keras.wrappers.scikit_learn import KerasClassifier

n_classes = 2


def create_model(n_hidden, learning_rate):
    model = Sequential()
    model.add(InputLayer(input_shape=(2850, )))
    model.add(Dense(units=n_hidden))
    model.add(Activation("sigmoid"))
    model.add(Dense(n_classes))
    model.add(Activation("softmax"))
    sgd = SGD(lr=learning_rate)
    model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=["accuracy"])
    return model
```

Figura 12.10: Código para el diseño del modelo de Artificial Neural Network.

```
# Parametrized values
N_FEATURES_OPTIONS = range(1, 500, 10)
MATRIXES = list(all_combinations(column_matrixes))
EXTRA_1D_COLUMS = [None]


param_grid = {
        'combining__columns': MATRIXES,
        # 'filter__columns': EXTRA_1D_COLUMS,

        'classify__n_hidden': random.randint(20, 200, 1000),
        'classify__learning_rate': random.uniform(0.001, 0.2, 1000),
        'classify__epochs': random.randint(10, 50, 1000),
    }
```

Figura 12.11: Código para la selección de parámetros de Artificial Neural Network.

Capítulo 13

Resultados

Una vez que obtenemos para cada algoritmos la mejor combinación de resultados, se ejecutan los modelos calculados con el conjunto de test y de entrenamiento inicial. De esta forma se pretende estimar el sobreajuste, "overfitting", del modelo.

13.1. Concepto de sobreajuste

El sobreajuste u "overfitting" se produce cuando un modelo cuando obtiene muy buenos resultados con los datos de entrenamiento, pero su precisión es notablemente más baja con el conjunto de test. Esto se produce porque el modelo se ha adaptado a los valores del conjunto de entrenamiento y no es capaz de generalizar para datos que no ha procesado. En la imagen 13.1 podemos observar gráficamente en que consiste este problema.

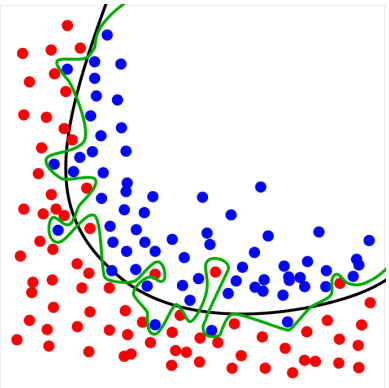

Figura 13.1: Ejemplo de sobreajuste, línea verde [4]

Este sobreajuste está directamente relacionado con la complejidad del modelo, cuanto más complejo sea más tendencia tendrá a sobreajustarse. Además, si el conjunto de datos del que disponemos es reducido, como en nuestro caso, este problema de sobreajuste estará muy presente.

13.2. Resumen de los resultados

En la tabla 13.1 podemos ver para cada algoritmo los resultados obtenidos cuando se ejecuta la mejor configuración obtenida en los pasos anteriores. Entre los resultados se distinguen los valores para el conjunto de entrenamiento y para el conjunto de test. De esta forma controlamos el sobreajuste anteriormente descrito.

	Todas las matrices		**Matrices de DTI**	
	Test	**Train**	**Test**	**Train**
Logistic Regression	72.73 %	67.69 %	78.57 %	84.42 %
Support Vector Machine	72.73 %	100 %	78.72 %	100 %
Gaussian Naive Bayes	60.61 %	70.77 %	78.72 %	86.02 %
Random Forest Classifier	72.73 %	96.92 %	74.47	93.55 %
Artificial Neural Network	65.00 %	70.51 %	82.14	82.14 %

Cuadro 13.1: Tabla resumen de los resultados de los experimentos por algoritmo

13.3. Precisión, exhaustividad y medida-F1

En las imágenes 13.2, 13.3, 13.4, 13.5, 13.6, 13.7, 13.8, 13.9, 13.10 y 13.11, extraídas de los resultados podemos ver para cada algoritmo los resultados de las dos ejecuciones: considerando todas las matrices o solamente las DTI. Se diferencian para cada clase (1, 0) los valores de precisión (precision), exhaustividad (recall) y la medida-F1 (f1-score) de cada ejecución. Estas tres medidas están relacionadas con los errores de tipo I y tipo II.

```
reduce_dim__n_components = 1
filter__columns = None
combining__columns = ('DTI_FA', 'DTI_L1', 'DTI_RX', 'LS')
classify__penalty = l1
classify__C = 37.70786516853933
Test
Accuracy in test: 72.73%
             precision    recall  f1-score   support

          0       0.70      1.00      0.82        21
          1       1.00      0.25      0.40        12

avg / total       0.81      0.73      0.67        33

Train
Accuracy in train: 67.69%
             precision    recall  f1-score   support

          0       0.68      0.95      0.79        42
          1       0.67      0.17      0.28        23

avg / total       0.67      0.68      0.61        65
```

Figura 13.2: Resultado ejecución algoritmo Logistic Regression. Considerando todas las matrices.

```
reduce_dim__n_components = 25
filter__columns = None
combining__columns = ('DTI_FA', 'DTI_L1', 'DTI_RX')
classify__penalty = l1
classify__C = 54.3932584269663
Test
Accuracy in test: 78.57%
             precision    recall  f1-score   support

          0       0.78      1.00      0.88        21
          1       1.00      0.14      0.25         7

avg / total       0.83      0.79      0.72        28

Train
Accuracy in train: 84.82%
             precision    recall  f1-score   support

          0       0.85      0.96      0.91        84
          1       0.82      0.50      0.62        28

avg / total       0.85      0.85      0.83       112
```

Figura 13.3: Resultado ejecución algoritmo Logistic Regression. Considerando sólo matrices DTI.

```
reduce_dim__n_components = 471
filter__columns = None
combining__columns = ('DTI_FA', 'DTI_L1', 'DTI_RX')
classify__degree = 2
classify__C = 306.51020408163265
Test
Accuracy in test: 72.73%
             precision    recall  f1-score   support

          0       0.73      0.90      0.81        21
          1       0.71      0.42      0.53        12

avg / total       0.72      0.73      0.71        33

Train
Accuracy in train: 100.00%
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        42
          1       1.00      1.00      1.00        23

avg / total       1.00      1.00      1.00        65
```

Figura 13.4: Resultado ejecución algoritmo Support Vector Machine. Considerando todas las matrices.

```
reduce_dim__n_components = 461
filter__columns = None
combining__columns = ('DTI_FA',)
classify__degree = 2
classify__C = 367.6122448979592
Test
Accuracy in test: 78.72%
             precision    recall  f1-score   support

          0       0.78      1.00      0.88        35
          1       1.00      0.17      0.29        12

avg / total       0.83      0.79      0.72        47

Train
Accuracy in train: 100.00%
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        70
          1       1.00      1.00      1.00        23

avg / total       1.00      1.00      1.00        93
```

Figura 13.5: Resultado ejecución algoritmo Support Vector Machine. Considerando sólo matrices DTI.

```
reduce_dim__n_components = 1
filter__columns = None
combining__columns = ('DTI_FA', 'DTI_MD', 'DTI_RX', 'RAW')
Test
Accuracy in test: 60.61%
             precision    recall  f1-score   support

          0       0.68      0.71      0.70        21
          1       0.45      0.42      0.43        12

avg / total       0.60      0.61      0.60        33

Train
Accuracy in train: 70.77%
             precision    recall  f1-score   support

          0       0.73      0.88      0.80        42
          1       0.64      0.39      0.49        23

avg / total       0.70      0.71      0.69        65
```

Figura 13.6: Resultado ejecución algoritmo Gaussian Naive Bayes. Considerando todas las matrices.

```
reduce_dim__n_components = 11
filter__columns = None
combining__columns = ('DTI_L1',)
Test
Accuracy in test: 78.72%
             precision    recall  f1-score   support

          0       0.78      1.00      0.88        35
          1       1.00      0.17      0.29        12

avg / total       0.83      0.79      0.72        47

Train
Accuracy in train: 86.02%
             precision    recall  f1-score   support

          0       0.87      0.96      0.91        70
          1       0.81      0.57      0.67        23

avg / total       0.86      0.86      0.85        93
```

Figura 13.7: Resultado ejecución algoritmo Gaussian Naive Bayes. Considerando sólo matrices DTI.

```
reduce_dim__n_components = 351
filter__columns = None
combining__columns = ('DTI_L1', 'DTI_RX', 'RAW', 'LS')
classify__max_depth = 91
Test
Accuracy in test: 72.73%
             precision    recall  f1-score   support

          0       0.77      0.81      0.79        21
          1       0.64      0.58      0.61        12

avg / total       0.72      0.73      0.72        33

Train
Accuracy in train: 96.92%
             precision    recall  f1-score   support

          0       0.95      1.00      0.98        42
          1       1.00      0.91      0.95        23

avg / total       0.97      0.97      0.97        65
```

Figura 13.8: Resultado ejecución algoritmo Random Forest Classifier. Considerando todas las matrices.

```
reduce_dim__n_components = 121
filter__columns = None
combining__columns = ('DTI_MD', 'DTI_RX')
classify__max_depth = 81
Test
Accuracy in test: 74.47%
             precision    recall  f1-score   support

          0       0.74      1.00      0.85        35
          1       0.00      0.00      0.00        12

avg / total       0.55      0.74      0.64        47

Train
Accuracy in train: 93.55%
             precision    recall  f1-score   support

          0       0.92      1.00      0.96        70
          1       1.00      0.74      0.85        23

avg / total       0.94      0.94      0.93        93
```

Figura 13.9: Resultado ejecución algoritmo Random Forest Classifier. Considerando sólo matrices DTI.

```
combining__columns = ('DTI_FA', 'DTI_MD')
classify__n_hidden = 72
classify__learning_rate = 0.11955873107296836
classify__epochs = 29
Test
Accuracy in test: 65.00%
             precision    recall  f1-score   support

          0       0.65      1.00      0.79        13
          1       0.00      0.00      0.00         7

avg / total       0.42      0.65      0.51        20

Train
Accuracy in train: 70.51%
             precision    recall  f1-score   support

          0       0.68      1.00      0.81        50
          1       1.00      0.18      0.30        28

avg / total       0.80      0.71      0.63        78
```

Figura 13.10: Resultado ejecución algoritmo Artificial Neural Network. Considerando todas las matrices.

```
combining__columns = ('DTI_FA',)
classify__n_hidden = 88
classify__learning_rate = 0.03291384334119732
classify__epochs = 48
Test
Accuracy in test: 82.14%
             precision    recall  f1-score   support

          0       0.81      1.00      0.89        21
          1       1.00      0.29      0.44         7

avg / total       0.86      0.82      0.78        28

Train
Accuracy in train: 82.14%
             precision    recall  f1-score   support

          0       0.81      0.99      0.89        84
          1       0.90      0.32      0.47        28

avg / total       0.84      0.82      0.79       112
```

Figura 13.11: Resultado ejecución algoritmo Artificial Neural Network. Considerando sólo matrices DTI.

La precisión es la habilidad del clasificador etiquetar correctamente los valores dentro de su grupo. La exhaustividad en cambio es la habilidad para acertar con las etiquetas en todo el conjunto. Por ejemplo, si disponemos de 10 ejemplares de donde 3 son del tipo A y 7 son del tipo B. A través de nuestro modelo obtenemos 5 elementos del tipo A pero sólo 2 han sido correctamente etiquetados, entonces nuestra precisión sería 2/3 y la exhaustividad 2/10. La medida-F1 engloba las otras dos siendo un valor único ponderado de la precisión y la exhaustividad. Su valor se calcula con la siguiente fórmula:

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Aunque en este trabajo nos hemos centrado en la medida de exactitud ("accuracy"). Dependiendo de la finalidad de los experimentos se pueden ajustar para que la selección de los parámetros de la configuración tuviese como meta alguno de éstos.

Parte III

Epílogo

Capítulo 14

Conclusiones

14.1. Experimentos

La obtención de unos algoritmos que lograron unos resultados satisfactorios ha sido una ardua labor. Los problemas con el sobreajuste de los modelos (ver 13.1) siempre han estado presentes, en gran parte, debido al poco número de ejemplares con los que se dispone para el entrenamiento y la estrecha relación de los índices del tensor. Además, también ha ocurrido que no todos los algoritmos respondieran como pudiésemos esperar, incluso buscando intensamente una configuración apropiada.

Los algoritmos más simples han proporcionado unos mejores resultados y apenas eran propicios al sobreajuste. En cambio, cuando se aumentó la complejidad de los algoritmos, los resultados ya no eran tan satisfactorios. Por ejemplo, se intentó incrementar el número de capas ocultas en la "Artificial Neural Network" pero todos los resultados empeoraron notablemente. Por este motivo, se han descartado un gran número de algoritmos que lograban estimar las diferentes clases que representan las disfunciones cognitivas pero presentaban indicios claros de sobreajuste.

Finalmente se han seleccionado cinco modelos que han dado una respuesta muy esperanzadora. Como se puede ver en la tabla resumen 13.1, tres de los cinco han superado un acierto del 70 % al introducir todas las matrices de conectividad en el modelo de predicción de la variable clínica (rendimiento cognitivo). Cuando se han usado solamente las matrices relacionadas con el tensor de difusión (índices sensibles a la microestructura del tejido subyacente), los resultados han mejorado considerablemente, llegando a una predicción del 75 % e incluso superando el 80 % en algunos casos. Esto ocurre también cuando buscamos una mejor configuración, como se ven en las imágenes 13.2, 13.4, 13.6, 13.8 y 13.10, donde se muestran las configuraciones de los modelos que dan mejores resultados. A pesar de contar con todas las matrices de conectividad para la predicción del rendimiento cognitivo, el mejor modelo de predicción incluye únicamente las matrices del tensor. Señalando que estos datos son quizás los más sensibles para la manifestación de la disfunción cognitiva en los pacientes con EM.

Estos resultados, a pesar de ser convincentes, no pueden darse como definitivos ya que la búsqueda de los parámetros no ha sido posible ejecutarla completamente con todas las combinaciones posibles. Dado el elevado

número de posibles combinaciones de los hiperparámentros, es inviable ejecutar todas ellas en el sistema. Por ello se ha seleccionado la configuración entre 150 combinaciones aleatorias. Por otra parte, se aprecia claramente la importancia de los índices DTI sobre el resto y cómo éstas pueden son una fuente de datos válida para la estimación de las disfunciones cognitivas.

14.2. Proyecto de investigación

La realización de este trabajo ha sido un gran reto personal. Es la primera vez que me enfrento a un trabajo de investigación y, a la vez, a un ejercicio relacionado con el aprendizaje automático en el ámbito clínico. Todo esto conjuntamente con las lecciones aprendidas durante la realización del trabajo y los resultados obtenidos me han proporcionado experiencia muy positiva y enriquecedora.

La primera dificultad encontrada durante este trabajo fue la "compresión del negocio". Los datos usados en este trabajo tienen tras de sí numerosas investigaciones, teorías y técnicas que han sido necesarias para poder comprender y trabajar con los datos. El conocimiento adquirido durante esta fase inicial ha sido de gran utilidad a lo largo de la ejecución de todo el proyecto.

Uno de los mayores retos durante el proceso de investigación ha sido la búsqueda de algoritmos capaces de dar respuesta a las objetivos planteados dado el gran número de opciones disponibles y todas las variantes que el aprendizaje automático provee. Además, siempre hay que considerar los requerimientos y el tiempo de cómputo que los algoritmos necesitan. A pesar de las implementaciones óptimas ofrecidas por los "frameworks", el hardware usado se ha visto a menudo incapaz de soportar la carga de trabajo en un tiempo razonable. Para lidiar en parte con este problema se optó por la búsqueda aleatoria de la configuración de los algoritmos seleccionados.

En definitiva, el objetivo inicial planteado para la superación del trabajo final de Máster de Ciencia de Datos ha sido cumplido. Los resultados obtenidos demuestran cómo los algoritmos de aprendizaje automático pueden ayudar a la predicción del rendimiento cognitivo del paciente con EM a través de la cuantificación del volumen lesional y la microestructura de la red cerebral.

Capítulo 15

Trabajo Futuro

Este trabajo de investigación se puede considerar como un primer paso para una investigación más profunda. La obtención de una cohorte mayor y la introducción de nuevos datos relacionados con la disfunción cognitiva, como por ejemplo medidas de atrofia, podrían mejorar la eficiencia de las técnicas de aprendizaje automático.

En primer lugar, sería muy interesante comprobar los resultados obtenidos con una fuente de datos más amplia y poder investigar la configuración óptima de los algoritmos con más profundidad.

Por falta de tiempo se han quedado sin ejecutar algunas propuestas para la reducción de la dimensionalidad, por ejemplo CSP, y también técnicas que combinan clasificadores sencillos para obtener uno más complejo (bagging y stacking). Experimentos con estas técnicas también podrían producir resultados interesantes.

Bibliografía

[1] "Machine Learning in Python: Logistic Regression - HEURISTICS." [Online]. Available: http://blog.welcomege.com/machine-learning-python-logistic-regression/

[2] "Introduction to Support Vector Machines — OpenCV 2.4.13.6 documentation." [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[3] "GitHub - pagutierrez/tutorial-sklearn: Tutorial sobre scikit-learn completo." [Online]. Available: https://github.com/pagutierrez/tutorial-sklearn/

[4] "Sobreajuste - Wikipedia, la enciclopedia libre." [Online]. Available: https://es.wikipedia.org/wiki/Sobreajuste

[5] M. A. Rocca, M. P. Amato, N. De Stefano, C. Enzinger, J. J. Geurts, I.-K. Penner, A. Rovira, J. F. Sumowski, P. Valsasina, and M. Filippi, "Clinical and imaging assessment of cognitive dysfunction in multiple sclerosis," *The Lancet Neurology*, vol. 14, no. 3, pp. 302–317, 3 2015. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1474442214702509

[6] A. Kutzelnigg and H. Lassmann, "Pathology of multiple sclerosis and related inflammatory demyelinating diseases," 2014, pp. 15–58. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/B9780444520012000029

[7] N. D. Chiaravalloti and J. DeLuca, "Cognitive impairment in multiple sclerosis," *The Lancet Neurology*, vol. 7, no. 12, pp. 1139–1151, 12 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S147444220870259X

[8] F. Barkhof, "The clinico-radiological paradox in multiple sclerosis revisited," *Current Opinion in Neurology*, vol. 15, no. 3, pp. 239–245, 6 2002. [Online]. Available: https://insights.ovid.com/crossref?an=00019052-200206000-00003

[9] P. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi, "In vivo fiber tractography using DT-MRI data," *Magnetic resonance in medicine: official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, vol. 44, no. 4, 2000.

[10] S.-K. Song, J. Yoshino, T. Q. Le, S.-J. Lin, S.-W. Sun, A. H. Cross, and R. C. Armstrong, "Demyelination increases radial diffusivity in corpus callosum of mouse brain," *NeuroImage*, vol. 26, no. 1, pp. 132–140, 5 2005. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/15862213http://linkinghub.elsevier.com/retrieve/pii/S1053811905000224

[11] B. Jeurissen, A. Leemans, J.-D. Tournier, D. K. Jones, and J. Sijbers, "Investigating the prevalence of complex fiber configurations in white matter tissue with diffusion magnetic resonance imaging," *Human Brain Mapping*, vol. 34, no. 11, pp. 2747–2766, 11 2013. [Online]. Available: http://doi.wiley.com/10.1002/hbm.22099

[12] D. Tuch, T. Reese, M. Wiegell, N. Makris, J. Belliveau, and V. Wedeen, "High angular resolution diffusion imaging reveals intravoxel white matter fiber heterogeneity," *Magnetic resonance in medicine: official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, vol. 48, no. 4, 2002.

[13] E. Martínez-Heras, F. Varriano, V. Prčkovska, C. Laredo, M. Andorrà, E. H. Martínez-Lapiscina, A. Calvo, E. Lampert, P. Villoslada, A. Saiz, A. Prats-Galino, and S. Llufriu, "Improved Framework for Tractography Reconstruction of the Optic Radiation," *PLOS ONE*, vol. 10, no. 9, p. e0137064, 9 2015. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/26376179http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4573981http://dx.plos.org/10.1371/journal.pone.0137064

[14] M. Rubinov and O. Sporns, "Complex network measures of brain connectivity: Uses and interpretations," *NeuroImage*, vol. 52, no. 3, pp. 1059–1069, 9 2010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S105381190901074X

[15] S. Llufriu, E. Martinez-Heras, E. Solana, N. Sola-Valls, M. Sepulveda, Y. Blanco, E. H. Martinez-Lapiscina, M. Andorra, P. Villoslada, A. Prats-Galino, and A. Saiz, "Structural networks involved in attention and executive functions in multiple sclerosis," *NeuroImage: Clinical*, vol. 13, pp. 288–296, 1 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2213158216302339?via%3Dihub

[16] I. Gabilondo, E. Martínez-Lapiscina, E. Martínez-Heras, E. Fraga-Pumar, S. Llufriu, and S. Ortiz, "Trans-synaptic axonal degeneration in the visual pathway in multiple sclerosis," *Annals of neurology*, vol. 75, no. 1, 2014.

[17] E. Bullmore and O. Sporns, "Complex brain networks: graph theoretical analysis of structural and functional systems," *Nature Reviews Neuroscience*, vol. 10, no. 3, pp. 186–198, 3 2009. [Online]. Available: http://www.nature.com/articles/nrn2575

[18] O. Sporns, *Networks of the brain.* MIT Press, 2011.

[19] N. Shu, Y. Liu, K. Li, Y. Duan, J. Wang, C. Yu, H. Dong, J. Ye, and Y. He, "Diffusion Tensor Tractography Reveals Disrupted Topological Efficiency in White Matter Structural Networks in Multiple Sclerosis," *Cerebral Cortex*, vol. 21, no. 11, pp. 2565–2577, 11 2011. [Online]. Available: https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/bhr039

[20] M. Bozzali, B. Spanò, G. Parker, G. Giulietti, M. Castelli, B. Basile, S. Rossi, L. Serra, G. Magnani, U. Nocentini, C. Caltagirone, D. Centonze, and M. Cercignani, "Anatomical brain connectivity can assess cognitive dysfunction in multiple sclerosis," *Multiple Sclerosis Journal*, vol. 19, no. 9, pp. 1161–1168, 8 2013. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1352458512474088

[21] C. Louapre, V. Perlbarg, D. García-Lorenzo, M. Urbanski, H. Benali, R. Assouad, D. Galanaud, L. Freeman, B. Bodini, C. Papeix, A. Tourbah, C. Lubetzki, S. Lehéricy, and B. Stankoff, "Brain

networks disconnection in early multiple sclerosis cognitive deficits: An anatomofunctional study," *Human Brain Mapping*, vol. 35, no. 9, pp. 4706–4717, 9 2014. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/24687771http://doi.wiley.com/10.1002/hbm.22505

[22] R. A. Dineen, J. Vilisaar, J. Hlinka, C. M. Bradshaw, P. S. Morgan, C. S. Constantinescu, and D. P. Auer, "Disconnection as a mechanism for cognitive dysfunction in multiple sclerosis," *Brain*, vol. 132, no. 1, pp. 239–249, 1 2009. [Online]. Available: https://academic.oup.com/brain/article-lookup/doi/10.1093/brain/awn275

[23] J.-P. Stellmann, S. Hodecker, B. Cheng, N. Wanke, K. L. Young, C. Hilgetag, C. Gerloff, C. Heesen, G. Thomalla, and S. Siemonsen, "Reduced rich-club connectivity is related to disability in primary progressive MS," *Neurology - Neuroimmunology Neuroinflammation*, vol. 4, no. 5, p. e375, 9 2017. [Online]. Available: http://nn.neurology.org/lookup/doi/10.1212/NXI.0000000000000375

[24] R. Ouellette, Bergendal, S. Shams, J. Martola, C. Mainero, M. Kristoffersen Wiberg, S. Fredrikson, and T. Granberg, "Lesion accumulation is predictive of long-term cognitive decline in multiple sclerosis." *Multiple sclerosis and related disorders*, vol. 21, pp. 110–116, 3 2018. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/29550717

[25] Y. Zhang, S. Lu, X. Zhou, M. Yang, L. Wu, B. Liu, P. Phillips, and S. Wang, "Comparison of machine learning methods for stationary wavelet entropy-based multiple sclerosis detection: decision tree, k -nearest neighbors, and support vector machine," *SIMULATION*, vol. 92, no. 9, pp. 861–871, 9 2016. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0037549716666962

[26] R. S. Desikan, F. Ségonne, B. Fischl, B. T. Quinn, B. C. Dickerson, D. Blacker, R. L. Buckner, A. M. Dale, R. P. Maguire, B. T. Hyman, M. S. Albert, and R. J. Killiany, "An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest," *NeuroImage*, vol. 31, no. 3, pp. 968–980, 7 2006. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1053811906000437

[27] D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks." Springer, Berlin, Heidelberg, 2013, pp. 411–418. [Online]. Available: http://link.springer.com/10.1007/978-3-642-40763-5_51

[28] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images," pp. 2843–2851, 2012. [Online]. Available: https://papers.nips.cc/paper/4741-deep-neural-networks-segment-neuronal-membranes-in-electron-microscopy-images

[29] S. Sarraf and G. Tofighi, "Classification of Alzheimer's Disease using fMRI Data and Deep Learning Convolutional Neural Networks," 3 2016. [Online]. Available: http://arxiv.org/abs/1603.08631

[30] Y. Yoo, T. Brosch, A. Traboulsee, D. K. B. Li, and R. Tam, "Deep Learning of Image Features from Unlabeled Data for Multiple Sclerosis Lesion Segmentation." Springer, Cham, 2014, pp. 117–124. [Online]. Available: http://link.springer.com/10.1007/978-3-319-10581-9_15

[31] J. Kawahara, C. J. Brown, S. P. Miller, B. G. Booth, V. Chau, R. E. Grunau, J. G. Zwicker, and G. Hamarneh, "BrainNetCNN: Convolutional neural networks for brain networks; towards predicting

neurodevelopment," *NeuroImage*, vol. 146, pp. 1038–1049, 2 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1053811916305237?via%3Dihub

[32] "Hamarneh's MIA Research Group." [Online]. Available: https://sites.google.com/view/hamarneh-research/home

[33] R. C. Craddock, P. E. Holtzheimer, X. P. Hu, and H. S. Mayberg, "Disease state prediction from resting state functional connectivity," *Magnetic Resonance in Medicine*, vol. 62, no. 6, pp. 1619–1628, 12 2009. [Online]. Available: http://doi.wiley.com/10.1002/mrm.22159

[34] S. Lemm, B. Blankertz, T. Dickhaus, and K.-R. Müller, "Introduction to machine learning for brain imaging," *NeuroImage*, vol. 56, no. 2, pp. 387–399, 2011.

[35] "scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation." [Online]. Available: http://scikit-learn.org/stable/

[36] "Keras Documentation." [Online]. Available: https://keras.io/

[37] "TensorFlow." [Online]. Available: https://www.tensorflow.org/

[38] "Caffe2 — A New Lightweight, Modular, and Scalable Deep Learning Framework." [Online]. Available: https://caffe2.ai/

[39] "NumPy — NumPy." [Online]. Available: http://www.numpy.org/

[40] "SciPy.org — SciPy.org." [Online]. Available: https://www.scipy.org/

[41] "Python Data Analysis Library — pandas: Python Data Analysis Library." [Online]. Available: https://pandas.pydata.org/

[42] "seaborn: statistical data visualization — seaborn 0.8.1 documentation." [Online]. Available: https://seaborn.pydata.org/

[43] "Installation — Matplotlib 2.2.2 documentation." [Online]. Available: https://matplotlib.org/

[44] "Project Jupyter — Home." [Online]. Available: https://jupyter.org/

[45] "LaTeX - A document preparation system." [Online]. Available: https://www.latex-project.org/

[46] "Overleaf: Real-time Collaborative Writing and Publishing Tools with Integrated PDF Preview." [Online]. Available: https://www.overleaf.com/

[47] J. B. Boringa, R. H. Lazeron, I. E. Reuling, H. J. Adèr, L. E. Pfennings, J. Lindeboom, L. M. de Sonneville, N. F. Kalkers, and C. H. Polman, "The Brief Repeatable Battery of Neuropsychological Tests: normative values allow application in multiple sclerosis clinical practice," *Multiple Sclerosis Journal*, vol. 7, no. 4, pp. 263–267, 8 2001. [Online]. Available: http://journals.sagepub.com/doi/10.1177/135245850100700409

[48] "¿Qué es el grupo de Imagen Avanzada en enfermedades Neuroinmunológicas (ImaginEM)? — Programa de Neuroinmunología." [Online]. Available: http://www.neuroimmunologybcn.org/es/investigacion/imaginem/

[49] "What is a Data Frame? · mobileink/data.frame Wiki · GitHub." [Online]. Available: https://github.com/mobileink/data.frame/wiki/What-is-a-Data-Frame%3F

[50] "Manifiesto por el Desarrollo Ágil de Software." [Online]. Available: http://agilemanifesto.org/iso/es/manifesto.html

[51] "pytest: helps you write better programs — pytest documentation." [Online]. Available: https://docs.pytest.org/en/latest/

[52] "Coveralls - Test Coverage History & Statistics." [Online]. Available: https://coveralls.io/

[53] "PEP 8 – Style Guide for Python Code — Python.org." [Online]. Available: https://www.python.org/dev/peps/pep-0008/

[54] "PEP 257 – Docstring Conventions — Python.org." [Online]. Available: https://www.python.org/dev/peps/pep-0257/

[55] "Travis CI - Test and Deploy Your Code with Confidence." [Online]. Available: https://travis-ci.org/

[56] "Overview — Sphinx 1.8.0+ documentation." [Online]. Available: http://www.sphinx-doc.org/en/master/

[57] "GitHub Pages — Websites for you and your projects, hosted directly from your GitHub repository. Just edit, push, and your changes are live." [Online]. Available: https://pages.github.com/

[58] D. H. Wolpert, "The Lack of A Priori Distinctions Between Learning Algorithms," *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, 10 1996. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.1996.8.7.1341

[59] C. J. Brown and G. Hamarneh, "Machine Learning on Human Connectome Data from MRI," 11 2016. [Online]. Available: http://arxiv.org/abs/1611.08699

[60] T. de: and T. de:, *Naive Bayes classifier*. [Online]. Available: https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[61] L. Breiman and L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://link.springer.com/10.1023/A:1010933404324

[62] "Procesamiento paralelo CUDA — Qué es CUDA —NVIDIA." [Online]. Available: http://www.nvidia.es/object/cuda-parallel-computing-es.html