
UNIVERSITAT DE BARCELONA

Final Degree Project

**Biomedical Engineering Degree**

# "Machine learning approaches for the study of AD with brain MRI data"

Barcelona, 14th June 2021

Author: Laia Borrell Araunabeña

Directors: Roser Sala Llonch,

Agnès Pérez Millan

**ACKNOWLEDGEMENTS**

First of all, I would like to express my gratitude to both my tutors Roser Sala Llonch and Agnès Pérez Millan for guiding me, helping me and supporting me throughout the development of this project. Surely this thesis would have not been the same without their advice.

I would also like to thank my friends and family for giving me all the necessary support to carry out the project.

Sincerely,

Laia Borrell Araunabeña.

**ABSTRACT**

The use of automated or semi-automated approaches based on imaging data has been suggested to support the diagnoses of some diseases. In this context, Machine Learning (ML) appears as a useful emerging tool for this purpose, allowing from feature extraction to automatic classification.

Alzheimer Disease (AD) and Frontotemporal Dementia (FTD) are two common and prevalent forms of early-onset dementia with different, but partly overlapping, symptoms and brain patterns of atrophy. Because of the similarities, there is a need to establish an accurate diagnosis and to obtain good markers for prognosis. This work combines both supervised and unsupervised ML algorithms to classify AD and FTD.

The data used consisted of gray matter volumes and cortical thicknesses (CTh) extracted from 3T-T1 MRI of 44 healthy controls (HC, age: 57.8±5.4 years), 53 Early-Onset Alzheimer Disease patients (EOAD, age: 59.4±4.4 years) and 64 FTD patients (FTD, age: 64.4±8.8 years). A principal component analysis (PCA) of all volumes and thicknesses was performed and a number of principal components (PC) that accumulated at least 80% of the data variance were entered into a Support Vector Machine (SVM). Overall performance was assessed using a 5-fold cross-validation.

The algorithm proposed achieved an accuracy of 91.7±5.76 % in the HC vs EOAD classification, 83.3±5.2 % for HC vs FTD, 83.0±5.8 % for EOAD vs FTD and 77.7±5.2 % when performing a multiclass classification between the 3 groups. The first PC was used to create disease-specific patterns.

By using a low number of features, combining information from CTh and subcortical volumes, the algorithm proposed is able to classify HC, EOAD and FTD with fairly good accuracy. Thus, this approach can be used to reduce the amount of data used in ML algorithms while providing interpretable atrophy patterns.

## LIST OF TABLES

## LIST OF FIGURES

**GLOSSARY**

*AD* – Alzheimer Disease.

*ADI* – Alzheimer Disease International.

*Aseg atlas* – shows the subcortical volumes.

*AUC* – Area under the curve.

*BFS* – Backwards feature selection.

*BOE* – Boletín oficial del estado.

*BrainSegVol* – volume of the whole brain, except brain stem.

*BrainSegVolNotVent* – volume of the whole brain except brain stem and ventricles.

*CATI* – Clinical Advanced Technologies Innovation.

*CSC measures* – Cortical and subcortical measures.

*CSF* – cerebrospinal fluid.

*CT* – Computed tomography.

*CTh* – Cortical thicknesses.

*CV* – Cross-validation.

*DK atlas* – Desikan-Killiany atlas. Shows the cortical and subcortical thicknesses.

*DTI* – Diffusion tensor imaging.

*EOAD* – Early-onset Alzheimer Disease.

*FDA* – Food and Drinks Administration.

*FFS* – Forward feature selection.

*FTD* – Frontotemporal dementia.

*GM* – Gray matter.

*ISMRM* – International Society for Magnetic Resonance in Medicine.

*LDA* – Linear discriminant analysis.

*LOO* – Leave One Out.

*LR* – Linear regression.

*MaskVol* – Volume of the count non-zero voxels of the brain mask.

*MCI* – Mild Cognitive Impairment. Can be either Alzheimer converters (*cMCI*) or non-converters (*ncMCI)*.

*ML* – Machine learning.

*MRI* – Magnetic resonance imaging.

*NIA* – National Institute of Aging.

*NIH* – National Institutes of Health.

*PC* – Principal component. *PC1* states for first principal component.

*PCA* – Principal component analysis.

*PET* – Positron Emission Tomography.

*PET-FDG* – Positron emission tomography - fluorodeoxyglucose.

*RBF* – Radial basis function.

*RF* – Random forests.

*RFE* – Recursive feature elimination.

*ROI* – Regions of interest.

*sMRI* – structural magnetic resonance imaging.

*SPECT* – Single-photon emission computed tomography.

*SupraTentorialVolNotVent/Vox* – includes all brain volumes except cerebellum, brain stem and ventricles volumes.

*SVM* – Support Vector Machine.

*TIV* – Total intracranial volume.

*VBM* – Voxel-based morphometry.

*WM* – White matter.

.

**TABLE OF CONTENTS**

# 1. INTRODUCTION

## 1.1. Alzheimer Disease and its diagnosis

The National Institute of Aging (NIA) defines Alzheimer's Disease (AD) as an irreversible, progressive brain disorder that slowly destroys memory and thinking skills, and, eventually, the ability to carry out simple daily tasks, until patient's death [1]. AD affects people over the age of 65, but early-onset AD (EOAD) is also diagnosed in patients younger than 65. The brain of an Alzheimer patient accumulates abnormal proteins (Aβ and tau) in the form of amyloid plaques and neurofibrillary tangles [2], eventually resulting in an irreversible loss of neurons. Currently, AD is the most common neurodegenerative disease, representing the principal cause of dementia in the elderly population of developed countries. In 2020, over 50 million people worldwide suffered dementia, and these numbers are expected to reach the 152 million cases in 2050 due to population aging [3].

Although AD has no cure, there are medicines available that help in delaying the disease progress. Thus, its early diagnosis is essential.

In 2011, the National Institutes of Health (NIH) and the Alzheimer's Association developed an updated diagnostic guidelines with respect to the previous one, published in 1984. In this guidelines, three stages of AD were described: a preclinical stage, characterized by the first brain changes, including amyloid build-up, without significant clinical symptoms evident; a Mild Cognitive Impairment (MCI) stage, attributed to a loss of cognitive functions regarding memory, but that do not interfere with the patient's independence; and Alzheimer's dementia, to which some of the MCI patients evolve, marked by a progressive impairment in memory, decision making, orientation and language, significant enough to affect a person's daily life [4].

According to the same guidelines, in preclinical stages, the amyloid build-up can be found with positron emission tomography (PET) imaging and a cerebrospinal fluid (CSF) analysis, however, the future prognosis is still unknown. In a more advanced stage, in MCI, the research for diagnosis is focused on standardizing biomarkers for amyloid plaques and other signs of injury to the brain. Among these biomarkers, elevated levels of tau or decreased levels of beta-amyloid in the CSF are found, as well as reduced glucose uptake in the brain observed in PET scans. Also, atrophy of certain brain areas detected by **high-resolution structural Magnetic Resonance Imaging (MRI)** is well-correlated with both tau deposition and neuropsychological effects, so it becomes another biomarker of AD and its progression. Finally, the diagnosis criteria for Alzheimer's dementia, which applies to the last stage of the disease, includes biomarkers testing to increase the sureness of the diagnosis and to distinguish Alzheimer from other types of dementias. [5,6]

As stated above, the intensity and stage of the neurodegeneration can be identified with the help of atrophy measurement of certain brain areas with structural MRI (sMRI), such as the hippocampus, the entorhinal cortex or the amygdala. Therefore, sMRI-based feature extraction happens to play an important role in AD diagnosis and prognosis [7]. The objective of feature extraction is to retrieve a set of accurate quantified features such as size, shape or volume, from neuroimaging data that contain relevant information for the disease diagnosis.

Nonetheless, manual measurements of these structures on MRI are time-consuming and do not capture the full pattern of atrophy [8]. Therefore, many automated and semi-automatic feature

extraction techniques exist for high-resolution sMRI data analysis, which consist of several image-processing and statistical analysis steps. [9].

When applying feature extraction methods, neuroimaging modalities generate an extreme huge amount of data. Thus, to analyse it and observe inherent patterns in the data, there is a need for multivariate data analysis and Artificial Intelligence (AI) techniques, such as Machine Learning (ML) [10].

### 1.2. Artificial Intelligence: machine learning

In 2019, the High-Level Expert Group on Artificial Intelligence (HLEG) defined AI systems as "*software (and possibly also hardware) systems designed by humans that, given a complex goal, act in the physical or digital dimensions by perceiving their environment through data acquisition, interpreting the collected structured or unstructured data, reasoning on the knowledge, or processing the information, derived from this data and deciding the best action(s) to take to achieve the given goals.*" These systems can use symbolic rules or learn a numeric model [11]. Also, AI encompasses many techniques, such as Machine Learning (ML); machine reasoning, which refers to scheduling, search and optimization; and robotics, which includes sensors and actuators [12].

**Machine Learning** is a subfield of AI which consists of building models or applications that learn from data, called *training data*, in order to improve its accuracy when predicting the outcome of new data without being explicitly programmed to do so. There are different ways in which ML algorithms learn from the training data:

Supervised learning. In this type of learning, expert humans feed the algorithm with labelled data, that is, pairs of inputs with known outputs. Then, the algorithm looks for the function which best maps the input data to the output in a generalizable from. Supervised learning is useful in either classification or regression problems. In classification problems, the algorithm predicts a discrete value for the input data, identifying each data point with a specific class. On the other hand, regression problems work with continuous numeric data.

Unsupervised learning. Sometimes, well-labelled data sets are not easy to obtain. In these cases, algorithms learn completely by themselves by finding unknown patterns in the data without any labels. Unsupervised learning aims to find structure in the data by extracting useful features and analyzing them. The way the model organizes data is different depending on the problem. One of the ways is clustering, where the model looks for training data with similar features and groups them together in what is called a cluster. Another common way is association, in which the model looks for some key attributes in a data point and predicts the other ones by checking them in those data points with which the point is correlated [13].

Semi-supervised learning. These are hybrid algorithms lying between supervised and unsupervised models. The training dataset is filled with both labelled and unlabelled data, which is a common situation in those cases where labelling data requires from expert human skills.

Reinforced learning. This kind of learning is common in robotics, since it allows machines to determine in an automatic way the best behaviour in a specific context which maximizes the model performance. Trial and error search, together with a reward feedback, are required by the system to learn.

### 1.3. Machine Learning and neurodegenerative diseases diagnosis with sMRI

As mentioned, clinical studies have demonstrated that measures of structural atrophy using sMRI data are a promising biomarker for an accurate diagnosis of AD patients. Thus, automatic volumetric measures of cortical and subcortical volumes, together with measures of cortical thicknesses (CTh), both extracted from sMRI images, happen to be essential biomarkers for early detection of the AD [14]. In addition to providing certainty to the diagnosis of the disease, its non-invasive nature and the consequent lack of pain for the patient, make structural MRI a comfortable and effective diagnostic test.

Nevertheless, the utility of these sMRI-derived data is not limited to differentiate controls from affected patients, but is also useful in differentiating AD from other clinical pathologies, such as **frontotemporal dementia (FTD),** which, together with AD, is one of the leading causes of early-onset dementia. However, FTD differs from Alzheimer in the sense that FTD patients usually develop early noncognitive behavioural changes together with frontal atrophy in sMRI scans, while AD patients have early cognitive changes and relatively reserved behaviour together with hippocampal, entorhinal cortex and medial-temporal lobe atrophy on sMRI scans [15].

Moreover, brain tissue loss from the whole brain or specific regions, as well as the rate of structural atrophy measures, highly correlates with cognitive deficits changes [16], so they are also great biomarkers for the diseases progression.

Recent advances in neuroimage and its analysis have led to new useful automated tools to extract valuable neuroimaging information. These techniques generate large datasets encompassing whole brain regions instead of single regions of interest (ROI), which must be manually segmented and rely on previous knowledge. Then, if all the measurements extracted from the MRI scans aim to be useful for AD diagnosis, this triggers the need for automatic classification methods that perform equally or even better than experts in the high-consuming clinical task of recognising patterns in the data for further diagnosis. Furthermore, the fact that those methods are able to use whole brain information makes them more capable of distinguishing between healthy, AD patients and other types of neurodegenerations, since a single region, like the temporal lobe, can be also damaged in other types of diseases.

For this purpose, many automatic classification algorithms have already been implemented, providing useful tools when analyzing the data and finding inherent disease-related patterns in it. The models used include from Support Vector Machines (SVM) to K Nearest neighbours (KNN) or Decision Trees, all of them supervised learning algorithms. Many other studies as [17] also use neural networks.

Nonetheless, feature extraction directly from sMRI leads to very high dimensional data. Thus, when dealing with such large datasets, a reduction of data dimensionality is usually needed, particularly when the number of labelled subjects is small. Some ways of achieving this goal are discarding some of the features extracted and keeping others or projecting the data to a reduced dimensional space with some dimensionality reduction techniques such as Principal Component Analysis (PCA).

It is worth mentioning that the application of these algorithms is not limited to subject classification between controls, EOAD or other neurodegenerative diseases such as FTD, but they are also a

great tool to study which brain regions are more atrophied in each disease and, therefore, more significant for the posterior classification. Thus, when a model is applied, it is not only its interpretability that matters, understanding interpretability as the ability of the algorithm to determine a cause and its effect, such as an atrophied region and the consequent disease, respectively. Instead, attention should also be paid to explainability, which can be defined as understanding which factors from the data are causing the differentiation between subjects from different groups and what is the reason.

### 1.4. Objectives

All the previous concepts comprise the needed context to appropriately follow the presented project, which **main goal** is to research on an automatic machine learning classification tool for early-onset Alzheimer Disease diagnosis. The model aims to predict if a patient is either healthy, suffering from EOAD or suffering from FTD, thus helping doctors in the disease diagnosis and differentiation from other neurodegenerative diseases. All of this, by taking as inputs the cortical and subcortical volumes, together with the CTh measurements extracted from the patient's structural MRI scan.

Furthermore, some secondary objectives for the project were defined:

- Conduct an **in-depth research on the state of the art of automated classification systems** applied to MRI extracted data that provide certainty in the diagnosis of Alzheimer Disease. Thus, reviewing the different algorithms employed and the results obtained, taking into account the data being used.
- To **implement a pipeline able to classify between patients and healthy controls** including a dimensionality reduction step and a classification algorithm, which must be chosen considering the performance obtained but also the computational cost and the capability of the models to be generalized.
- **Explore which brain regions appear more atrophied** in AD or, in other words, which regions result more significant for the classification and subsequent diagnosis, and check if they match with the ones reviewed in the literature.
- To add patients suffering from frontotemporal dementia to the original dataset in order to perform a multiclass classification pipeline. Therefore, a second **automated classification tool capable of distinguishing between three groups (controls, EOAD patients and FTD patents)** aims to be developed.
- Finally, to **assess if there are actual differences in the brain atrophy pattern between AD and FTD** and, if positive, to explore the significant regions differentiating both neurodegenerative diseases.

### 1.5. Spatial and temporal limitations of the project

The main limitations of the project were temporal and due to the global pandemic situation.

Firstly, the thesis was developed in a relatively tight time window, from February to June 2021. This fact, in addition to cause a more intensive work and less spaced, limiting the realization of any

changes or correction to the framework, restricted even more the time available for obtaining controls and patients samples. Furthermore, the data with which the work was developed comes from neurologists of the hospital, which had to stop their usual activities during the pandemic to be dedicated full time to Covid19. This fact, added to the temporal limitation, presented an extra limitation to the size of the sample used to train the algorithm.

However, it was not only a sample's size limitation which was encountered, but also problems with other neuroimaging modalities data availability, such as DTI, which could have added certainty to the classification and led to better performances.

Finally, mobility and capacity restrictions supposed that all the meetings of the team were online, adding an extra difficulty in communication and organization.

### 1.6. Scope of the project

As stated before, the main objective of the project is to perform a pipeline able to classify EOAD patients from healthy subjects. Furthermore, a third group with FTD dementia patients aims to be added and classified.

In order to meet this goal, the scope of the project encompasses: delving into the machine learning world and its applications related to sMRI neuroimaging for the diagnosis of neurodegenerative diseases; the development of an automated pipeline for further classification between controls, EOAD and FTD patients, including a dimensionality reduction step, a classification step and a performance assessment. In addition, to obtain clinical information useful for research, the retrieval of the disease-related patterns of both EOAD and FTD, therefore analysing which features or brain regions contribute with the highest variance to the data.

On the other hand, the T1-weighted scans pre-processing, as well as the extraction of the CTh and the cortical and subcortical volumes from the images is out of the scope of the project. Also, the addition of other biomarkers of AD and FTD, such as PET-fluorodeoxyglucose (PET-FDG) scans or CSF analysis, or other MRI modalities like DTI, is also non contemplated. Finally, since this project has purely research objectives and needs further investigation, the algorithm is not intended to be commercialized.

## 2. STATE OF THE ART

In the last few years, many multivariate data analysis tools and machine learning algorithms were used in the field of neurodegenerative diseases, and particularly of Alzheimer Disease, since it is the most common one. Assuming that AD is correlated with progressive changes in brain structure and functionality, significant results have outcome from the implementation of automated softwares capable of finding patterns in the data extracted from sMRI that lead to a high accuracy diagnosis of the disease.

### 2.1. Search criteria

Since wide range of different cohorts, features and methods were used in numerous studies, a set of different papers regarding ML application in AD research centred on sMRI were reviewed, as well as classification pipelines proposed for distinguishing AD patients from FTD ones. However, many similar ML approaches for classifying MCI patients likely to convert to AD (cMCI) from those non-converters (ncMCI) were published, so the different algorithms and methods implemented in this classification task have also been reviewed.

The search was done in PubMed, looking for articles in English published from 2008 up to March 2021 using the search term “Alzheimer Disease” combined with “Machine Learning”, “Frontotemporal Dementia”, “structural MRI”, “SVM”, “Decision Trees”, “KNN”, “PCA” and “MRI data”. To include an article in the state-of-the-art review, it was checked that both T1-weighted sMRI data and a ML algorithm were used.

Usually, a classification framework includes feature extraction, a feature selection or dimensionality reduction step, a classification algorithm to develop a robust predictive model and a validation method to validate the performance (Figure 1). Therefore, the diverse methods employed in each step in the studies were pointed out. Nevertheless, as mentioned in the previous section, feature extraction is out of the scope of the project, so little research was made of this subject.

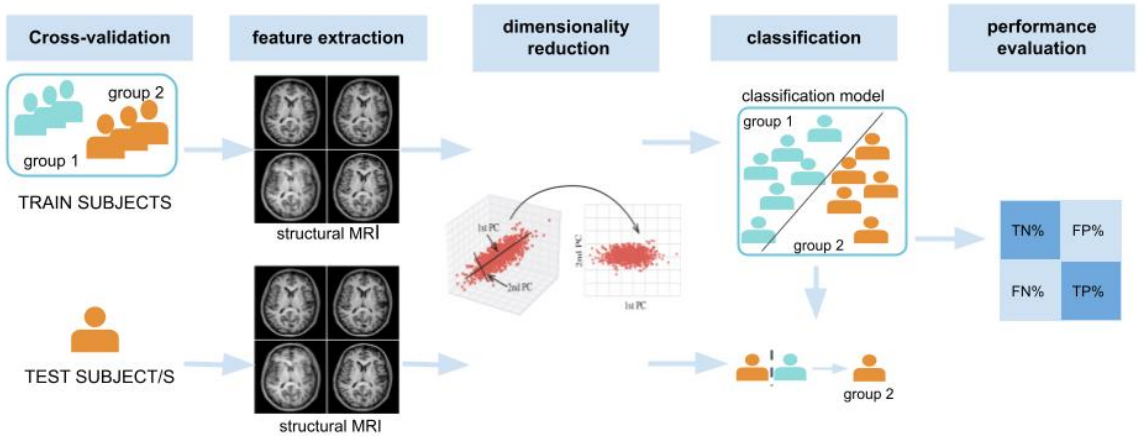



***Figure 1:*** *main steps of a conventional neuroimaging classification pipeline. 1) Cross-validation, 2) feature extraction, 3) dimensionality reduction, 4) classification and 5) performance evaluation.*

Followingly, a summary table (Table 1) with most of the articles reviewed is presented, showing the different steps beforementioned:

| Article | Type of classification | Number of observations | Feature extraction | Feature Selection | Classifier Algorithm | Validation method | Performance evaluation (best results) |
|---|---|---|---|---|---|---|---|
| [20] | Binary: AD-HC, FTD-HC AD-FTD | 37 AD, 12 FTD, 49 age-matched HC | sMRI: GM and WM VBM | PCA | SVM | 10-iterations LOO CV | Accuracy: AD-HC 99.7%, AD-FDT 84.3%, FTD-HC 100% |
| [21] | Binary: AD-HC, FTD-HC AD-FTD | 84 AD, 51 FTD, 94 HC | sMRI: GM-VM, WM-VBM, CSF | *None* | SVM | Holdout method | Accuracy: AD-HC 88% FTD-HC 85%, AD-FTD 82%. |
| [25] | Binary: EOAD-HC, EOFTD-HC EOAD-EOFTD | 24 EOAD, 33 EOFTD, 34 HC | sMRI: WM-VBM, GM-VBM, CSF, DTI: FA | *None* | SVM | 4-fold CV | AUC: EOFTD-EOAD 84% (sMRI + FA), 72% (sMRI). |
| [18] | Binary: AD-HC, HC-cMCI, cMCI-ncMCI | 144 AD, 189 HC, 136 cMCI, 166 cnMCI | Measures of subcortical volumes from sMRI | Random Forest | SVM | 20-fold CV | AUC: AD-HC 97%, HC-cMCI 92%, 75% cMCI- ncMCI. |
| [24] | Binary: AD-HC, HC-cMCI, cMCI-ncMCI | 81 AD, 171 HC, 39 cMCI, 35 ncMCI | sMRI: CSC thicknesses, hippocampal volume, and VBM | PCA | SVM, RF and KNN | *Not mentioned* | AUC: AD-HC 94%, HC-cMCI 95%, cMCI- ncMCI 87%. |
| [19] | Binary: HC-AD. Multiclass: HC-AD-MCI | 70 HC, 70 AD, 74 MCI | CSC thicknesses, subcortical volumes | PCA | SVM and RELM | 10-fold CV | Accuracy: HC-AD 80%, HC-AD-MCI 61.5% |
| [28] | Binary: HC-AD, cMCI- ncMCI | 226 HC, 182 AD, 389 (cMCI+ ncMCI). | VBM and CSC thicknesses | RFE | SVM | 10-fold CV | Accuracy: AD-HC 90%, cMCI-ncMCI 72%. |
| [22] | Binary: AD-HC, HC-MCI, cMCI-ncMCI | 117 AD, 122 MCI, 112 HC | Subcortical volumes and hippocampal volume | None | Orthogonal Partial Least Squares | 7-fold CV | Accuracy: AD-HC 89%, HC-MCI 84%, CMCI/ ncMCI 68% |
| [26] | Binary: AD-HC, HC-MCI | 51 AD, 99 MCI, 52 HC | 93 GM-ROI from MRI, 93 ROI from PET and 3 features from CSF | t-test | SVM | 10-fold CV | Accuracy: AD-HC 93.2% HC-MCI 76.4% |
| [29] | Binary: HC vs FTD Multiclass: HC vs FTD subtypes | 96 FTD, 84 HC | sMRI: CSC GM, subcortical volumes and total WM DTI: FA | PCA | KNN, SVM, RF, LR and LDA | 10-fold CV | Accuracy: 88% HC-FTD, 76% HC-FTD subtypes |
| [31] | Multiclass: AD-FTD-HC | 34 AD, 30 FTD, 14 HC | sMRI: CSC GM, subcortical volumes and total WM | PCA | SVM | LOO CV | AUC: 76.5% HC-AD-FTD |

***Table 1:*** *state of the art resume table. Reviewed studies for AD classification pipelines and the methods used in each step: groups being classified, nº of observations, feature extraction method, dimensionality reduction algorithm, classification algorithm, validation method and performance evaluation and results. GM: gray matter, WM: white matter.*

### 2.2. Feature extraction

In machine learning, features are a subset of variables that are given as input data to the classifiers, so they aim to quantify accurate information from neuroimaging data able to cover the most relevant patterns of the neurodegenerative disease. Since it is an important step in the classification framework and can directly affect its performance, many automated and semi-automated methods were implemented for analyzing sMRI images.

The features extracted by the different methods range from a single voxel to cortical and subcortical substructures, or even whole brain data. However, for neuroimage classification purposes, the most common measurements are cortical or subcortical volumes and CTh [18,19], while voxel-based morphometry (VBM) [20,21] is mostly used for structural comparisons between subjects.

Also, methods based on single regions are used, in particular segmentations of the hippocampus [22] and the entorhinal cortex, which appear in the literature as atrophied regions in AD. Nonetheless, adding more regional and global measures of the brain to the hippocampal volume has proven better performances [23]. Finally, other studies combine as well different sets of features that have shown to improve the performance, such as VBM and cortical and subcortical (CSC) measures [24], also including features from other neuroimaging modalities, such as DTI [25], or PET-FDG [26].

### 2.3. Feature selection

As stated in the introduction, feature extraction methods tend to generate huge amounts of data, leading to the so-called *curse of dimensionality*, which consists of computational difficulties when dealing with high dimensional data and a small number of subjects. In such circumstances, the classification tends to overfit the data, which makes the algorithm non-generalizable. Therefore, to overcome this problem, a reduction of the data dimensionality is needed, with the aim to reduce the number of features that are given as inputs to the classifier. In [27], it was demonstrated that choosing the appropriate feature selection or dimensionality reduction method has a positive effect in the resulting accuracies of the classification, so many approaches were implemented in different studies.

The main methods used for feature selection in the literature can be divided between unsupervised, which do not see the class labels and choose those features that maximize data variance, and supervised. Among the unsupervised methods, the most common approach implemented in many studies classifying AD patients from HC or DFT patients, was principal component analysis [20,22,24], which consists of transforming the data into a subspace of lower dimensionality that explains most of the data variance.

On the other hand, supervised methods can be classified between filter, wrapper and embedded methods. The first ones, which are independent from the classifier being used and select relevant features regarding general traits of the data, are used by means of a t-test in [26] to select the most discriminative brain regions. Wrapper methods, on the contrary, consider the classifier being used and aim to find the feature subset that maximizes its performance. Among these methods, Recursive Feature Elimination (RFE), which eliminates a specified number of features at each iteration, was used in [28] to classify AD patients from controls and cMCI from ncMCI. Finally, embedded methods, such as random forests used in [18] for feature selection, combine filter and

wrapper methods properties. It is implemented by classification algorithms which already have their own feature selection methods built-in or, in other words, that perform feature selection in their training.

### 2.4. Classification algorithms

A wide range of supervised classification algorithms were implemented in the field of AD diagnosis, as well as in classification works that aim to distinguish between AD and other diseases such as FTD. As mentioned, these kinds of classifiers used data labels to learn from a training set for further prediction of unknown samples.

The most common algorithm used in the neurodegenerative field for binary classifications were Support Vector Machines [20,21,18,19]. In short, this algorithm consists of finding a hyperplane in the feature space that best separates the datapoints belonging to different classes. Then, new datapoints are mapped into the feature space and predicted as one class or another depending on which side of the hyperplane they fit.

Although SVM is the most used classifier, other algorithms are reported in the literature. In [24], the K-Nearest-Neighbours (KNN) classifier was compared to SVM for classifying MCI types and AD from controls and proved to not perform as good as SVM. KNN classifier assumes that datapoints belonging to a same group are mapped closed to each other in the feature space. Therefore, as in SVM, the algorithm maps the different datapoints and stores their labels, but in this case, when the test samples are mapped as well in the feature space, they are labelled as the most common class in their k-nearest training points.

Other studies such as [29] have implemented an embedded algorithm such as Random Forests (RF) or Decision Trees for both feature selection and classification tasks. These are supervised classifiers that follow a flowchart-like structure composed by nodes, branches and leafs. Each node is labelled with an input feature and the branches coming out from that node are labelled each of the possible values that the feature can take. Then, these branches lead either to another internal node labelled with a different feature or to a leaf, a terminal node that will be labelled with a class or a probability distribution of the classes. When choosing which feature performs the best split at a given node, the certainty in a particular prediction aims to be maximized.

Logistic Regression (LR) and Linear Discriminant Analysis (LDA) performances, both supervised algorithms, were also assessed in [29]. In binary classification problems, LR uses a logistic function to output a linear equation result between 0 and 1, which corresponds to the probability of belonging to a class or another. On the other hand, LDA projects the data onto a one-dimensional straight line, such that the data points belonging to a same class are as close as possible.

When performing multiclass classifications between AD, FTD and HC, some research groups have used deep neural networks [30]. However, machine learning algorithms such as SVM have also showed good results [31].

Finally, each of the mentioned classifiers depends on a series of parameters, which can be optimized for a given training set with a cross validation method, such as in [25] or [24]. Nonetheless, attention must be paid to the chosen parameters to avoid overfitting.

### 2.5. Validation method

The main idea of model validation is to use a part of the data set to fit and train the classifier and another part to test and evaluate it.

Among the different methods, the holdout method, the k-fold cross-validation (CV) and the Leave One Out (LOO) CV are examples. The holdout method consists of randomly splitting the dataset into a train and a test sets, the first one used for the learning step and the second one to evaluate the performance. In [21], the holdout method was used to evaluate the classification accuracy between AD and FTD patients. On the other hand, k-fold CV is one of the most common approaches, and consists of dividing the subject's sample into k-folds and using each of the folds as a test set in different iterations while the other folds remain as training set. It was used in [18,28] to classify AD from HC and MCI converters from non-converters. Finally, LOO cross validation was used in [20] to evaluate the performance of the classification between AD and FTD patients. It consists of taking k-fold CV to the extreme, since k happens to be equal to the number of samples in the data set, leaving a single subject aside for testing at each iteration.

In [19], Kumar R. et al. compare the three cross-validation methods that have just been explained in both a binary and a multiclass classification frameworks. The results show that in the binary classification the performance was better with LOO CV and in the multiclass classification the outcomes were better with 10-fold CV.

### 2.6. Performance evaluation

It is not a simple task to compare the performances of the methods obtained in the different studies by just looking at the results presented. The reason to this, in addition to the fact that each study uses its own feature extraction and selection method and classifier, is that the performances of the pipelines are assessed with different metrics.

The most common way of evaluating performance along the different studies was accuracy, which is defined as the number of correct predictions divided by the total number of predictions. Also, other metrics such as Cohen's kappa value were used to evaluate the model performance [24], which, unlike accuracy, takes into account the imbalance in class distribution. It is defined as the overall accuracy of the model minus the probability of agreement between the model predictions and the real class values if randomly predicted, all of this divided by 1 minus the same agreement.

Other studies like [25] evaluate the performance of the classifier with the Area Under the Curve (AUC), which measures the two-dimensional area under the ROC curve, which is a curve defined by the true positive and the true negative values.

### 2.7. Studies that reported significant regions in AD and FTD

Finally, some of the papers reviewed have reported a list of brain regions found to be significant and discriminative when classifying AD patients from controls, FTD patients from controls and AD patients from FTD.

In [20], many areas from the temporal and frontal lobes (e.g., orbitofrontal cortex), as well as the parietal lobes and medial structures appeared to be discriminative for AD-FTD classification. Also, in [25], the results confirm the temporal lobe contribution to the AD-FTD distinction, but also specify

the middle frontal gyrus and periventricular regions as discriminative. In [31], the discriminative power of hippocampus and ventricles was assessed and showed greater ventricular differences between the two groups: AD and FTD.

On the other hand, brain regions that were reported as significant for AD-HC classification are temporal lobe and periventricular regions, as well as the hippocampus and the amygdalas [31]. In FTD-HC classification, the temporal lobe is the one with grater discriminative power, but also amygdala+ and the bilateral frontal lobes contribute.

# 3. MARKET ANALYSIS

## 3.1. Addressed sector

As stated in the introduction, the classification pipeline implemented was mainly addressed to Alzheimer patients. According to Alzheimer Disease International (ADI), almost 10 million people worldwide develop dementia every year, from which around 5 million cases are AD. The incidence rates observed in early 2000s for younger adults (below the age of 60) were of 2.4 cases per 100,000 people per year, whereas this rate increased to 127 cases per 100,000 people per year for the population over 60. After 65, the incidence of the disease exponentially increases with age. Thus, the main target sector are subjects over 65 suspected of suffering dementia caused by AD.

The prevalence of AD is on the rise, mainly attributed to the increasing diagnostic rates and awareness of the disease [32]. The conventional methods for diagnosing AD, as can be cognitive ability tests such as Mini Mental State Examination (MMSE), or biopsies, fail the 50% of the time, mainly due to a lack of consensus on AD biomarkers. Thus, the need for effective and well-orchestrated diagnostic tools for the disease gives place for brain imaging.

## 3.2. MRI versus other neuroimaging techniques for AD diagnosis

Brain imaging is a recent diagnostic improvement method thanks to the progressive advancements in technologies and proved correlations in structural, functional and molecular findings to the disease onset and progression. Since there are no definitive guidelines for a diagnosis of AD, the most popular approach is to exclude other possible diagnosis, narrowing the possibilities to a probable AD case. Actually, brain imaging facilitates this exclusionary process by providing structural imaging of the brain and its regions, also enabling the differentiation of AD from other similar pathologies as FTD or vascular dementia. With the findings of biomarkers linked with the disease in clinical trials, brain imaging techniques are increasingly being used for safety monitoring.

In Table 2 [32], some of the most used brain imaging technologies in the AD diagnosis context and their main advantages and disadvantages are summarized. Although PET brain imaging allows for an accurate early diagnosis of AD, its price and invasive nature limits the technique, since radioactive substances must be injected in the patient's body. Also, PET becomes less useful when monitoring the disease progression and later stages of AD. In these aspects, MRI emerges as a great advantage since it is really sensitive to structural brain changes of the disease, allowing for a progressive disease monitoring. Moreover, MRI is completely non-invasive.

| Technique | Application | Advantages | Disadvantages | Average Price (€) |
|---|---|---|---|---|
| **CT** | Structural | Quick, easy and widely available. | Low resolution and limited use. Not sensitive to structural changes. | 348.70 |
| **MRI** | Structural | Widely available, high-resolution and safe. Sensitive to structural changes. | Low, lacks molecular specificity. | 2137.22 |
| **SPECT** | Functional | Detects brain changes through disease progression, low-cost alternative to PET. | Less specific and sensitive tool than PET. Low spatial resolution of the brain. | 900.40 |

| | | | | |
|---|---|---|---|---|
| **PET** | Molecular | Good for early detection of AD. Applied for monitoring Aβ loads in the brain via tracer molecules. | Limited availability. Radioactive chemicals injected into body. | 5484.25 |

***Table 2:*** *comparision between different neuroimaging techniques that can be applied for AD diagnosis. Computed Tomography (CT), strucural MRI, Single-Photon Emission Computed Tomography (SPECT) and Positron Emission Tomography (PET) are compared. [32]*

### 3.3. Historical evolution of the market

Once brain imaging started to become a key biomarker in AD diagnosis and other neurodegenerative diseases, in the mid-90's, statistical mapping and voxel-based analysis of the brain began to gain prominence in the neuroimaging field. The use of these methods was first limited to explore anatomical or functional differences between groups or to investigate correlations between imaging and cognitive symptoms. However, group comparisons usually do not allow to develop individually based imaging scores, which is essential to establish diagnosis indices for a single subject [33].

Because of this, in the early 2000's, the application of machine learning methods to neuroimaging studies aroused a lot of interest. This enabled the development of imaging signatures of brain structure that could be detected in single subjects. The earlier studies published on this topic were focused on SVM algorithms, which are an actual key aspect in neuroimaging diagnosis because of its ease of use and robustness. Another family of methods that gained popularity around 2001-2002 were random forests since they became very generalizable algorithms. Deep learning was the latest incorporation in the field and allowed to learn complex features in a hierarchical way.

In the last few years, the applications of machine and deep learning in neuroimaging studies were numerous, not just for AD but also for schizophrenia, Parkinson, and other diseases. However, this is a field that, until now, has been mainly limited to research studies and centres.

In Figure 2, the publications in PubMed from 2004 until 2020 found by searching MRI and Machine Learning are presented. As can be seen, the number of publications has increased exponentially in the last few years.

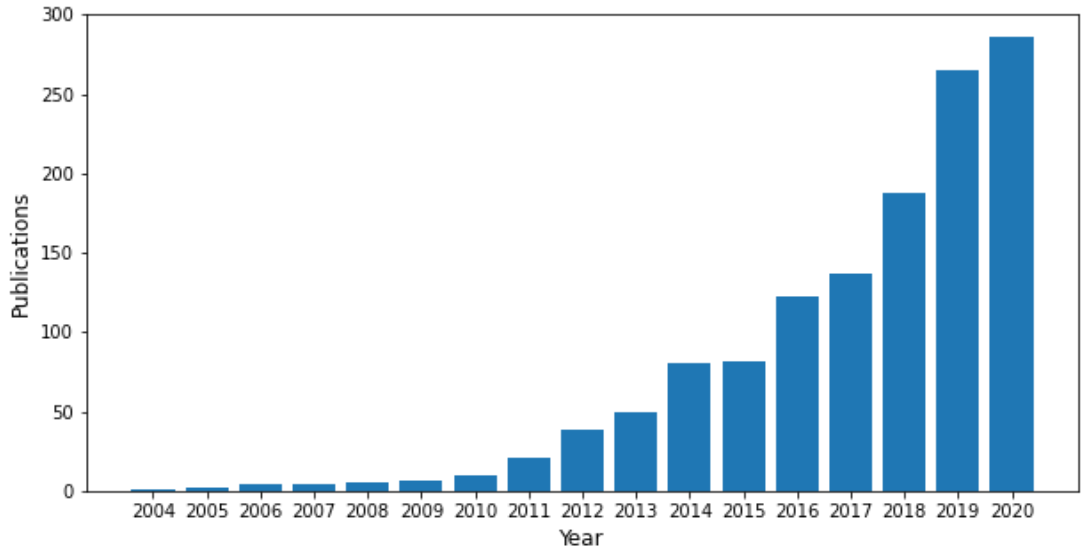



***Figure 2:*** *number of publications found in PubMed by year with the keywords "structural MRI" AND "Machine Learning" AND "AD" from 2004 to 2020.*

### 3.4. Future perspectives

In light of the above, with an increasingly aging population, Alzheimer's cases are expected to increase up to 75 million in 2030. And, although until now the field of AI applied to the diagnosis of AD has been quite limited to the research framework, it is expected that in the years to come new branches of large companies including this type of diagnosis will appear. In addition, new start-ups will be founded, such as the already existing *qmenta* [34] or *qubiotec* [35]. These enterprises offer powerful clouds for clinical trials, research and clinical care, where the clients can upload their neuroimages and they have them back together with a probability assigned to a determined diagnosis.

Moreover, new innovation centres focused on health technology, born from hospitals together with other technological entities, are also expected to increase in number. An actual example of this is the Clinical Advanced Technologies Innovation (CATI), a new centre promoted by Hospital Clínic and the technological centre Leitat. In this public-private collaboration, many enterprises have shown their interest in being part of it, some of them including AI solutions towards a personalized medicine.

Finally, it is worth mentioning that in the specific AD context these emerging AI algorithms not only provide diagnostic support but also become a key tool to assess the efficacy of treatments for the disease. An example is aducanumab, a monoclonal antibody which was recently commercialized with the name of Aduhelm and that acts on beta-amyloid plaques.

# 4. CONCEPTION ENGINEERING

Once the contextualization and the background of the project were reviewed, in this section, the different software options and algorithms for dimensionality reduction, classification and validation that can be implemented to address the previously defined objectives, will be discussed.

## 4.1. Analysis of the solutions

Below, a summary table with the different options raised regarding the programming software, the dimensionality reduction and classification algorithms, and the cross-validation method.

| | **Studied solutions** |
|---|---|
| **Programming software** | • Python<br>• R studio<br>• MATLAB |
| **Dimensionality reduction algorithms** | • Unsupervised: PCA<br>• Supervised<br>  – Wrapper methods: RFE / FFS / BFS<br>  – Filter methods |
| **Classification algorithms** | • K-Nearest Neighbours<br>• Support Vector Machine<br>• Decision Trees |
| **Validation method** | • Leave One Out CV<br>• K-fold CV<br>• Holdout method |

***Table 3:*** *Conception engineering. Studied solutions for the programming software, dimensionality reduction algorithm, classification algorithm, and validation method.*

### 4.1.1. Programming software

The proposed model could be implemented with different programming languages: Python [36], R studio [37] or MATLAB [38] among others.

**Python** is a free simple programming language which stands out for its easy to learn syntax. However, despite its fast interpretability, it is a high-level object-oriented and interpreted language. It supports many modules and packages, among which there is *Scikit-learn* or *Sklearn*, a widely used library for ML applications. *Sklearn* [39] is built on NumPy, SciPy and matplotlib, and supports classification, regression, clustering, dimensionality reduction, model selection and pre-processing tools.

**MATLAB** is a private system of numeric computing that uses its own programming language called "M language". It is also a user-friendly environment, mainly used for matrices manipulation, data and functions representation and algorithms implementation. It has its own range of packages, including a *Statistics and Machine Learning Toolbox* [40] which applications are remarkably similar to those in *Sklearn*: descriptive statistics, clustering, multidimensional data analysis and feature extraction, supervised classification, etc.

Finally, **R studio** is a free integrated development environment that uses R programming language for statistical computation and graphical representations. R, together with Python, is one of the more powerful tools for ML programming. It supports many ML packages such as *CARET* for

classification and regression problems, *random forest* or *e1071* for SVM and Naive Bayes classifiers, etc.

### 4.1.2. Dimensionality reduction

**Principal Component Analysis** is an unsupervised learning algorithm that transforms the original dataset into a subspace of lower dimensionality while capturing its maximum variability. This is done by finding a few new variables, directions uncorrelated with each other, that successively maximize variance and that are linear combinations of the features in the original set. These directions are the so-called Principal Components (PC) [41]. The result is a first PC being the direction along which the samples show the greatest variance, the second PC the direction, uncorrelated to the first PC, that shows the second greatest variance, and so on. The main uses of PCA are descriptive since it does not require any distributional assumptions regarding the dataset.

On the other hand, the most relevant supervised methods include Linear Discriminant Analysis and feature selection algorithms such as wrapper and filter methods.

**LDA** is a dimensionality reduction technique which, similarly to PCA, aims to project the dataset into a lower dimensionality subspace. However, LDA is supervised, so it creates the new subspace while maintaining the class-discriminatory information, finding linear combinations of the original features that maximize the separation between multiple classes.

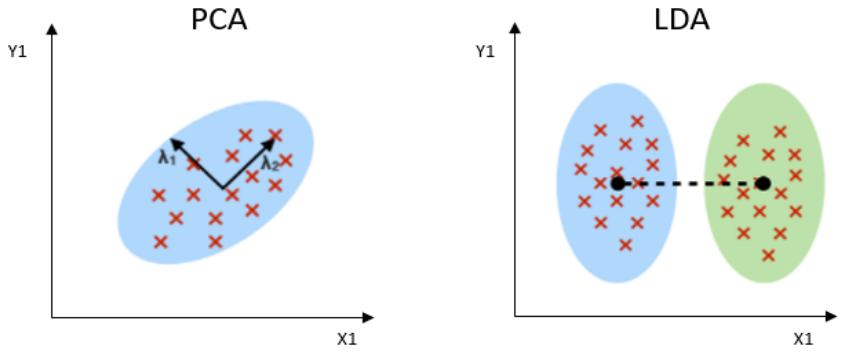



***Figure 3:*** *PCA (left) and LDA (right) graphical differentiation. PCA seeks for the directions in the dataset that capture the maximum data variability, while LDA aims to maximize the separation between groups. [42]*

**Wrapper methods** include a classifier into the feature selection algorithm and search for a feature subset that maximizes its performance. Wrapper methods are usually classified into Sequential Selection and Recursive Feature Elimination (RFE).

Sequential selection begins with an empty subset and first looks for the feature that maximizes the objective function, which in this case is the algorithm performance. Then, searches for another feature in the whole dataset that, added to the first one, also maximizes the performance. And keeps adding features in that way until the specified number of selected features is reached. The feature selection technique just described is the **Forward Feature Selection (FFS)** algorithm. If, on the contrary, the algorithm starts with a full subset and keeps eliminating the feature that causes the least performance loss at each iteration, it is called **Backwards Feature Selection (BFS).**

On the other hand, in order to find the best performing subset, **recursive feature elimination** eliminates n features from the total number (m), leaving the desired number of features (m-n), specified at the beginning. This is performed by fitting the given ML model multiple times and, at each iteration, ranking features by importance and discarding the weakest ones.

On the contrary, **filter methods** are independent of the classification algorithm and performance. They look for significant features by means of an information entropy test or other statistical dependencies. Starting with a feature subset (randomly selected or empty), they first evaluate the subset with the corresponding test and check if the result for the previous best subset was better than the actual one. If true, it is set as the current best feature subset. This process is continued until a certain criterion is met, which can be either the fact that the addition or removal of a new feature does not achieve better results or that a predefined performance threshold is achieved [43].

#### 4.1.3. Classification algorithm

**Support Vector Machine** is a supervised machine learning algorithm that can be used both for classification and regression analysis, although it is more commonly employed for classification. This algorithm aims to find an optimal hyperplane of *M -1* dimensions in an M-dimensional space (M: number of features) that best separates the different classes of data points from a training data set. The data points with the minimum distance to the hyperplane are called support vectors and, this minimum distance, also called margin, is maximized to find the optimal hyperplane. Then, new examples or cases are mapped to the M-dimensional space and predicted to belong to a class or another based on which side of the hyperplane they are mapped.

**K nearest neighbours** is also a supervised machine learning classifier algorithm that works under the assumption that similar datapoints are close to each other in the M-dimensional feature space. In the training phase, KNN maps the different feature vectors and stores the class labels of the training examples. Afterwards, each of the unlabelled data points are classified by assigning them the most common label among the *k* training points nearest to them. The distance *d* from the tested point to its *k* nearest train points can be assessed using different metrics, being the Euclidean or the Hamming distances two of the more common [44].

A **Decision Tree** is a supervised embedded algorithm that can be used both for regression and classification problems. It has a flowchart-like structure composed by nodes, branches and leafs. Each node is labelled with an input feature, and the branches coming out from a node are labelled with each of the possible values that the feature can take. These branches lead either to another internal node or to a leaf, a terminal node that will be labelled with a class or a probability distribution of the classes. When predicting a class label for a new data observation, the algorithm starts from what is called the root of the tree, the beginning node containing the entire dataset, and the different paths from the root to the leafs represent classification rules. The growth of these paths involves deciding which feature to choose for each node that best splits the set of data, and what conditions to use for splitting an internal node in two or more branches [45].

#### 4.1.4. Validation method

As stated in section 2, the **holdout method** simply consists of splitting the dataset into single train and test sets, the first one for teaching the algorithm and the second one to check the algorithm performance on unseen data. A common split of hold-out method is to put the 80% of the data into the training set and the remaining 20% into the test set.

On the other hand, cross-validation methods, such as **k-fold CV**, rely on a resampling procedure. Usually, the implementation of these models follows 4 steps. First, the dataset is shuffled randomly, secondly it is split into k-groups, thirdly, iteratively each group is used once as a test set while the

other groups are forming the training set with which the model being tested will be fitted. Finally, the performance of the model is summarized with the mean of the *k* iterations. Thus, k-fold CV makes sure that each fold is only used once for validation.

Finally, **Leave One Out CV** is a type of k-fold CV where k happens to be equal to the number of samples in the data set, leaving a single subject aside at each iteration. It gives each sample a chance of being test by the algorithm.

Both k-fold and LOO present a clear advantage with respect to the simple holdout method, since in small datasets the holdout is highly dependent on the splits produced. This problem is overcome with cross-validation. However, the main disadvantage of LOO is its computational cost, since the process requires a fitting and validation of the model as many times as the number of samples [46].

### 4.2. Proposed solution

In accordance with the above-mentioned, the programming environment finally chosen was Python. The main reason was that a ML introduction course was done using *Sklearn*, so the library was already familiar. Also, Python offers an open-source software, which allows to open the code in other computers without the need of downloading any payment software. However, R studio was also implemented to perform a statistical analysis between the demographic data of the different subject groups and to plot the diseases patterns onto an atlas.

The data dimensionality was finally reduced with PCA. The choice was partly made because of computational reasons since feature selection algorithms are iterative and therefore much more time consuming. Moreover, PCA was applied because it becomes a very powerful tool to obtain both conceptual and graphical clinical information about which brain regions are differing more between groups. And, also, since it is an unsupervised method, it results a generalizable algorithm with a low risk of overfitting despite the number of samples is limited, which lead to choose it over LDA.

In order to choose the number of PCs, the cumulative explained variance was taken as condition. An optimal number of PCs is a number which cumulative variance sums at least the 80% of the total variance of the dataset. However, if the threshold is set much higher, there is a risk of overfitting.

At the end, the ML classification algorithm used were support vector machines. The main reason that led to this choice was that SVM have been reported in the literature to be the ML algorithm that leads to better performances [24,29]. Moreover, Decision Trees were discarded because of their nature as embedded algorithms. This means that decision trees include (embed) the feature selection step into the own algorithm, making it difficult to retrieve the features selected in each node and extract the most significant brain regions. Finally, since the dataset used was relatively small and SVM is reported to obtain better results than KNN, the last one was also discarded.

Finally, the validation method implemented was k-fold CV to avoid split dependency. K-fold was chosen over LOO because of computational reasons, since LOO performs a number of iterations equal to the number of observations N, meaning that the model must be fitted N times. Moreover, LOO has a higher risk of overfitting, since the final performance is meaned between N models

trained with practically the same data, with the only difference of one observation. In contrast, the overlapping of training data in k-fold is lower.

## 5. DETAILED ENGINEERING

As mentioned, the main goal of the project was to develop an automated machine learning pipeline able to determine if a new patient who had undergone a structural MRI was suffering from AD or FTD, or if he/she was healthy. Also, the aim was to retrieve relevant clinical information about which brain regions were differing between the two ill and healthy brains. Therefore, both a binary classification study and a multiclass one were performed. The first mentioned aimed to classify between pairs of groups (AD patients and HC, FTD patients and HC, and AD patients and FTD patients) and study the brain atrophy patterns between the pairs. On the other hand, the multiclass study purposed to classify a new patient in one of the three groups.

To do so, the incoming structural MRI data, segmented with FreeSurfer into cortical thicknesses and cortical and subcortical gray matter volumes, was first divided into train and test sets with a k-fold cross-validation. Then, the dimensionality was reduced with PCA. In this step, the weights given from the first PC to each feature were stored and used to determine the most relevant brain regions between groups, which were plotted in a brain map. Followingly, the data transformed into the PCA subspace entered the classification step, were the SVM algorithm, which parameters were cross-validated, was fitted with the train set and then predicted the test set data labels. Finally, the performance of the model was assessed with the mean accuracy of the classification and a confusion matrix. In Figure 4, the general implemented pipeline is represented.

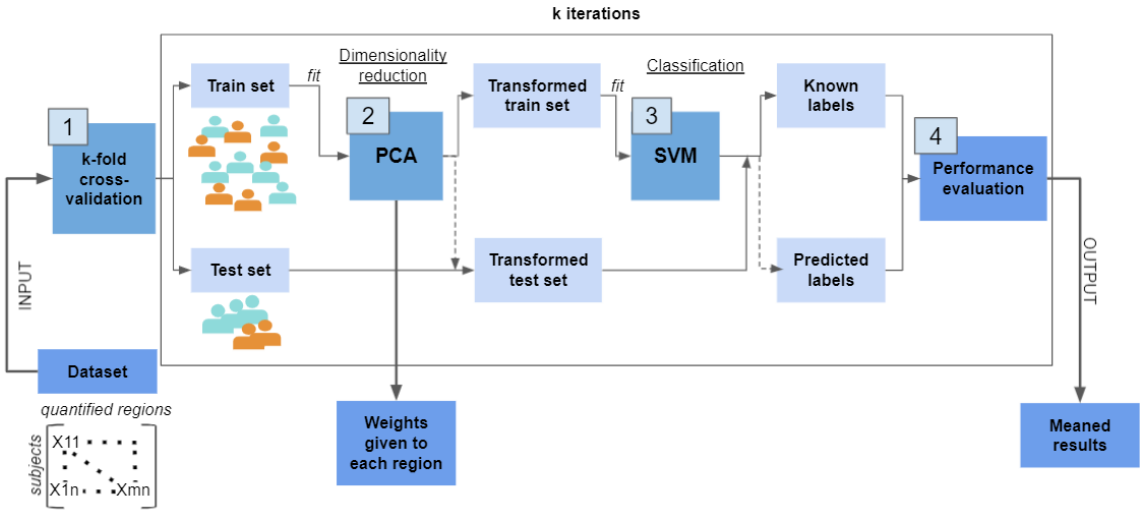



***Figure 4:*** *diagram showing the implemented pipeline. Starts splitting the dataset according to the cross-validation method (1), then Principal Component Analysis (PCA) is implemented (2) and the weights of the quantified brain regions are extracted, followed by Support Vector Machine (SVM) for the classification step (3). Finally, the performance is assessed (4). Parameter k corresponds to the number of iterations.*

### 5.1. Data management

The data used in this study comes from the Alzheimer and Memory unit in Hospital Clinic of Barcelona. It consists of a subset of 53 EOAD patients and 44 HC and 64 FTD patients, resulting in 161 subjects. Each subject was doubly acquired with a 3T Siemens MRI scanner with a T1-weighted magnetization-prepared 180 degrees radio-frequency pulses and rapid gradient-echo (MP RAGE) sequence. For each subject, cortical thicknesses and subcortical GM volumes were segmented by the Alzheimer and Memory unit (Agnès Pérez) using FreeSurfer, which is an open software for analyzing human brain structural and functional MRI images, used for the study of

cortical and subcortical brain anatomy. From this segmentation, a total of 36 cortical thicknesses from different structures were obtained for both right and left hemispheres, and also 65 subcortical volumes, including both right and left hemisphere measurements. However, the features that contained many null values or had a large number of void values were eliminated from the raw dataset. In order to work with manageable data, the format was transformed to *.csv* and read with Pandas library in Python.

Since the scope of this study did not include comparing differences between left or right hemisphere regions, we calculated the mean between left and right measures. Also, when comparing volumes between different subjects, it is good practice to normalize them for the total intracranial volume (TIV) to avoid significant differences that are due to the brain size and not because of the disease. Thus, each volume measure was normalized to the TIV and the intracranial volume was dropped from the dataset. Finally, a z-score normalization (Equation 1) was applied to the data in order to have comparable ranges of values for all features. HCs were labelled with a 0, the EOADs with a 1 and the FTDs with a 2.

$$z = \frac{x - mean}{sd} \quad \text{(eq. 1)}$$

Moreover, demographic data was acquired for each group, containing both age and sex. To check if age differences existed between the three groups, an ANOVA test was carried out with R studio. On the other hand, sex differences were checked with a Fisher test. Regarding age, the ANOVA test showed a p-value of $1.4 \cdot 10^{-6}$ (<0.05), meaning that significant differences in age between the three groups existed. In sex, although differences were not that big, a p-value of 0.03 was obtained. Thus, a post-hoc study was carried out using t-test to identify the pair-wise differences driving each result. The p-values obtained for age were 0.26, $4.4 \cdot 10^{-6}$ and $1.3 \cdot 10^{-5}$ between EOAD and HC, HC and FTD, and EOAD and FTD data, respectively. Regarding sex, the p-values were 0.3, 0.03 and 0.3 for EOAD-HC, FTD-HC and EOAD-FTD respectively.

Thus, the age and sex variables were included in the multiclass pipeline. In binary studies, age was added in all of them except for HC-EOAD, and sex was only added to HC-FTD. At the end, the feature space dimension was reduced to 59 for the multiclass and the HC-FTD studies, 58 for EAOD-FTD and 57 for HC-EOAD. The header of the EOAD-FTD feature set is shown in Figure 5.

| | BrainSegVol | BrainSegVolNotVent | BrainSegVolNotVentSurf | SubCortGrayVol | TotalGrayVol | SupraTentorialVol | SupraTentorialVolNotVent |
|---|---|---|---|---|---|---|---|
| 0 | 0.849142 | 0.995824 | 1.021203 | 0.857016 | 0.907915 | 0.808774 | 0.982985 |
| 1 | 1.374887 | 1.391053 | 1.437048 | 0.487682 | 1.381922 | 1.361360 | 1.411649 |
| 2 | -0.369672 | -0.143384 | -0.169192 | 0.030567 | 0.036704 | -0.311629 | -0.105541 |
| 3 | 0.300184 | 0.525530 | 0.534408 | 1.119336 | 0.565338 | 0.228762 | 0.471044 |
| 4 | -0.006120 | -0.288914 | -0.316743 | -0.536164 | -0.172684 | -0.073891 | -0.323309 |

5 rows × 58 columns

***Figure 5:*** *example taken from the data table for HC vs EOAD. Showing only 5 subjects and 7 features from the total.*

## 5.2. Implemented pipeline

### 5.2.1. Cross-validation

With the data already imported and normalized using z-score, the data was first divided into features (X) and labels (y) Then, it was split into train (*X_train*, *y_train*), and test (*X_test*, *y_test*) sets with a stratified **5-fold cross-validation.** The fact that the k-fold was stratified allowed that the splits, instead of being completely random, had a ratio between the target classes in each fold

equal as the one in the full dataset, so the folds are compensated. Also, 5 folds were chosen so that in each split approximately the 20% of the data was left for testing, while the resting 80% was used for training. In Figure 6, a graphical representation of the stratified 5-fold cross-validation implemented in the 3-group classification is shown.

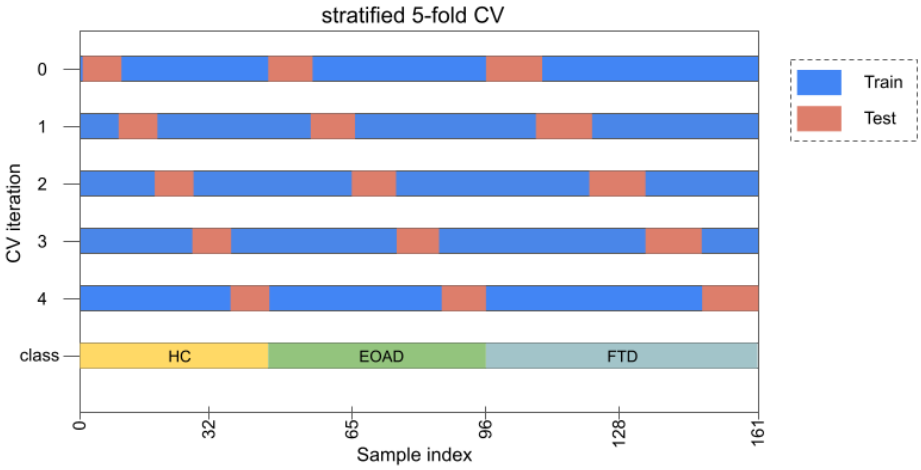



***Figure 6:*** *stratified 5-fold cross-validation implemented for the 3 groups. HC, EOAD and FTD subjects. At each iteration, a different fold containing the 20% of the dataset is reserved for testing.*

### 5.2.2. Dimensionality reduction

Once the data was split into a train and a test sets, and already in the cross-validation loop, the feature space dimensionality was reduced using **Principal Component Analysis**, which was also used to retrieve those features apporting more variance from the whole sample of each study.

As mentioned before, PCA is an unsupervised algorithm that reduces the dimensionality *m* of a dataset of N observations to a subspace of dimensionality *p*<*m*. It looks for uncorrelated PCs, which are linear combinations of the original features comprising the maximum variability of the data. The classic method for finding the PCs is the maximum variance approach [47]. The process starts with a N x *m* matrix, K, with mean-centred columns ($k_j$). The algorithm seeks for the linear combination of columns of the matrix with maximum variance, and these linear combinations are given by equation 2, where *a* is an *m*-dimensional vector of weights $a_1, a_2,...,a_m$.

$$\sum_{j=1}^{m} a_j k_j = Ka \quad \text{(Eq. 2)}$$

Then, the variance of such linear combination is given by equation 3, where C is the covariance matrix of K. This will consist of a square matrix of *mxm* dimensions denoting the covariance of each feature with the others (Figure 7). From here, finding the linear combination with maximum variance or first principal component (PC1), translates into looking for a vector *a* that maximizes this equation (Eq.3).

At the end, assuming that *a* must be unit-norm vectors, what results is the following equality (equation 4), where $\lambda$ and *a* end up being the eigenvalues and eigenvectors of C, respectively, which come from the diagonalization of the covariance matrix.

$$var(Ka) = a^T Ca \ (\text{Eq. } 3) \qquad Ca = \lambda a \ (\text{Eq. } 4)$$

$$\begin{bmatrix} V_{11} & C_{12} & \dots & C_{1m} \\ C_{21} & V_{22} & \dots & C_{2m} \\ \dots & \dots & \dots & \dots. \\ C_{m1} & \dots & \dots & V_{mm} \end{bmatrix}$$

***Figure 7:*** *covariance matrix. m x m matrix representing the covariance of each feature with the other features.*

The higher the eigenvalue, the higher variability in that direction, so the algorithm aims to find the largest eigenvalue and its corresponding eigenvector. A matrix composed by the eigenvectors can be constructed, where each vector or column corresponds to a PC. Then, by arranging them in descending order of variability, that is, by their eigenvalues, one acquires a matrix with the first PC as the first column, the second PC as the following column etc.

In Python, the *decomposition* module of the *Sklearn* library supports a PCA function in which the user can indicate the number of features or PCs to be extracted. In order to calculate the optimal number of PCs, in a previous step to the cross-validation, a PCA model was fitted with all the X data and set to get all the possible principal components, which equal the original feature space minus one: 55 for the EOAD vs HC study 56 for EOAD vs FTD and 57 the rest. From here, the explained variance accumulated by each component was calculated and the number of components that accumulate the 80% of the variance was assessed. In Figure 8, the cumulative variances versus the number of principal components for each group classification is shown.

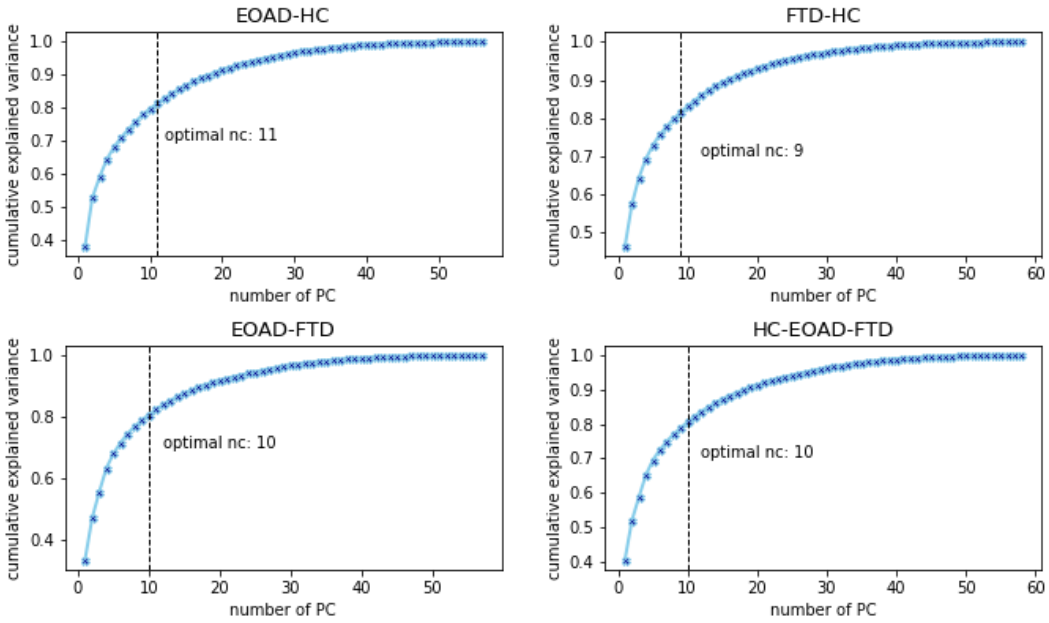



***Figure 8:*** *cumulative variances for all the principal components in each classification study. Minimum number of components needed to accumulate the 80% of the data variance is marked with a dashed line. Where "nc" states for number of principal components.*

The minimum number of PCs needed to accumulate the 80% of the dataset variance for each study was stored in a variable and entered to the cross-validation loop. There, the *X_train* was used to fit PCA, which was set to assess the number of PCs found in the previous step, and both *X_train* and *X_test* data were transformed into the new subspace. Followingly, the lists of weights given to the different features or regions by each component were stored in a dictionary with keys equal to the number of PC and values equal to a list of shape (*nº iterations, nº features*). At the end of the cross-validation, the mean and standard deviations of the weights given to each feature were calculated.

Since the first principal component itself accumulated approximately the 40% of the data variance, the weights given by this component to each feature were then stored in a .csv file. The file was read in R studio and used to represent the weights of each region graphically in a brain map.

### 5.2.3. Classification

Once with the train and test data transformed into the PCA subspace, the **Support Vector Machine** algorithm was fitted with the train set and used to classify the test data into the different classes.

As explained, SVM aims to find a hyperplane of *m*-1 dimensions that best separates the different data classes from the training set. When the dataset is linearly separable, two parallel hyperplanes are defined such as the distance between them is maximized (Figure 9). Then, the optimal hyperplane with the maximum margin, is the one lying halfway the other two. The equations defining the two parallel hyperplanes are (Eq. 5) and (Eq. 6), where *b* is a biased term, $W^T$ is the summatory of the normal vectors to the hyperplane, and X is the the summatory of the data point vectors. Both *b* and *W* parameters must be found so they maximize the margin distance. Then, every point on or above boundary $\Pi_1$ is of a determined class and every point on or below boundary $\Pi_2$ is of another class [48]. Therefore, for the classification of every new datapoint, the decision function in Eq. 7 is used:

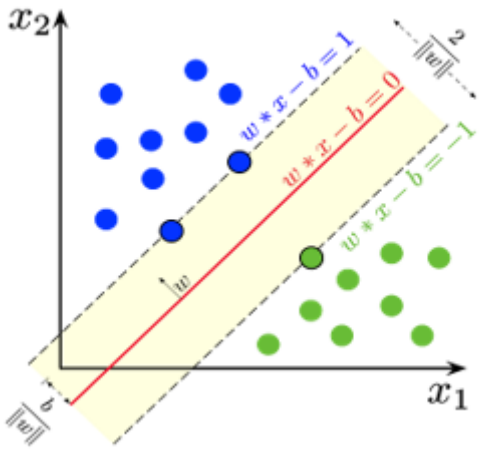



***Figure 9**: SVM classification. Grpahical representation of the hyperplanes generated by the algorithm.*

$$\Pi_1 = w^T X - b = 1 \quad (Eq.5)$$
$$\Pi_2 = w^T X - b = -1 \quad (Eq.6)$$

$$D(x) = sgn(w^T x + b) \; (Eq.7)$$

Sometimes, when data is not linearly separable, a projection of the data to a higher dimension, where it can be linearly separated, is needed to avoid outliers. This is what is called the *kernel trick*. In this non-linear classification, every dot product of two vectors in the algorithm is replaced by a kernel function, that allows to fit the hyperplane in a transformed feature space. Different kernel functions can be used (Figure 10), among others, there exist:

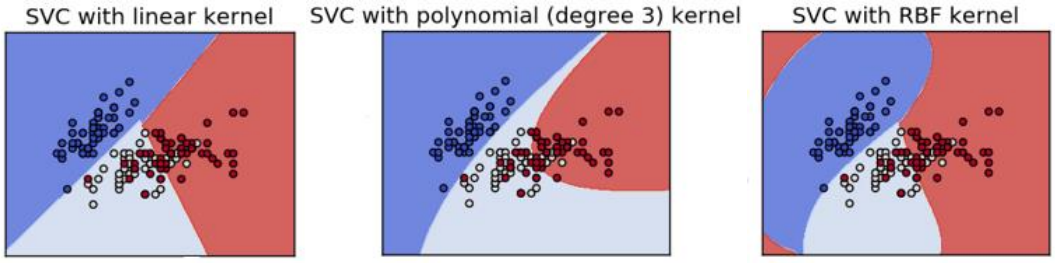



***Figure 10**. From left to right: graphic exemplification of Support Vector Machine classification with linear, polynomial and RBF kernels, from left to right [49].*

Linear kernel: $K(\vec{x_i}, \vec{x_j}) = (\vec{x}^T{}_i, \vec{x_j})$. It is equivalent to use a SVM without the kernel trick.

Polynomial kernel. It is a more generalized linear kernel, since it can classify curved or non-linear input spaces. $K(\vec{x}_i, \vec{x_j}) = (\vec{x}^T{}_i, \vec{x_j} + C)^d$, where *C* is a constant term and *d* is the kernel's degree.

RBF (Radial Basis Function). The Gaussian Radial Basis function is the most common one: $K(\vec{x}_i, center) = \exp(-\gamma||\vec{x_i} - center||^2)$. The γ parameter controls the weight of new training points. The higher its value, the more dependent of the closer points the decision boundary will be. Therefore, if γ is set too high, there might be risk of overfitting.

The effectiveness of SVM depends both on the kernel selection and its parameters. As mentioned before, when RBF kernel is used, the γ parameter must be chosen. On the other hand, the soft margin *C* parameter dictates the trade-off between maximizing the margin of the decision function and minimizing mistakes in the classification of training points [50]. For a smaller *C*, classification mistakes of the training data are given less importance and the algorithm focuses on maximizing the margin, while for higher *C* values, a smaller margin is accepted if the decision function classifies better all training points. Hence, when choosing a higher *C* value, overfitting must be watched out.

In order to find the optimal kernel and parameters for the SVM classification, the *GridSearchCV* function from the *Sklearn model_selection* module was implemented. This function takes as main inputs an estimator and a parameter grid. It performs an exhaustive combinatory search over the parameter's ranges given, fitting the *X_train* for each parameter combination and then scoring the estimator performance with the train labels *y_train*. Also, the performance can be cross validated any number of iterations by passing an integer number to the *cv* parameter of the *GridSearchCV* function. In this case, the *cv* parameter was set to 10.

The different parameter combinations passed as a grid to the *GridSearchCV* function consisted on three different kernels (linear, polynomial and RBF) combined with a C range of values from 0.1 to 1000 in logarithmic scale. Also, for the RBF kernel, a γ value was added to the combination, ranging from 0.001 to 1 in logarithmic scale. The combinations are shown in Table 3.

| KERNEL | C | γ |
|---|---|---|
| LINEAR | Range: 0.1, 1, 10, 100, 1000 | ------ |
| POLYNOMIAL | Range: 0.1, 1, 10, 100, 1000 | ------ |
| RBF | Range: 0.1, 1, 10, 100, 1000 | Range: 0.001, 0.01, 0.1, 1 |

***Table 4:*** *combination of parameters for the SVM estimator. C and γ values combined with each kernel in the grid search. C: soft margin parameter. γ : specific parameter for the RBF kernel that controls the weight given to new training points.*

Once the *GridSearchCV* function finds the optimal combination, it builds a classifier with that parameters. The resulting SVM estimator was then trained with the *X_train* and *y_train* data and used to predict the *X_test* labels, which were stored in a variable called *y_pred*.

Finally, the predicted labels were compared with the true labels (*y_test*) in order to build a confusion matrix and to assess the overall accuracy of the prediction and the specific accuracy of each class. The overall accuracies were calculated using the *accuracy_score* function and the specific ones with the *classification_report* function, both provided by the *metrics* module of *Sklearn.* Then, once outside the cross-validation loop, the mean values of the accuracies and standard deviations were calculated for the 5 iterations.

### 5.3. Results

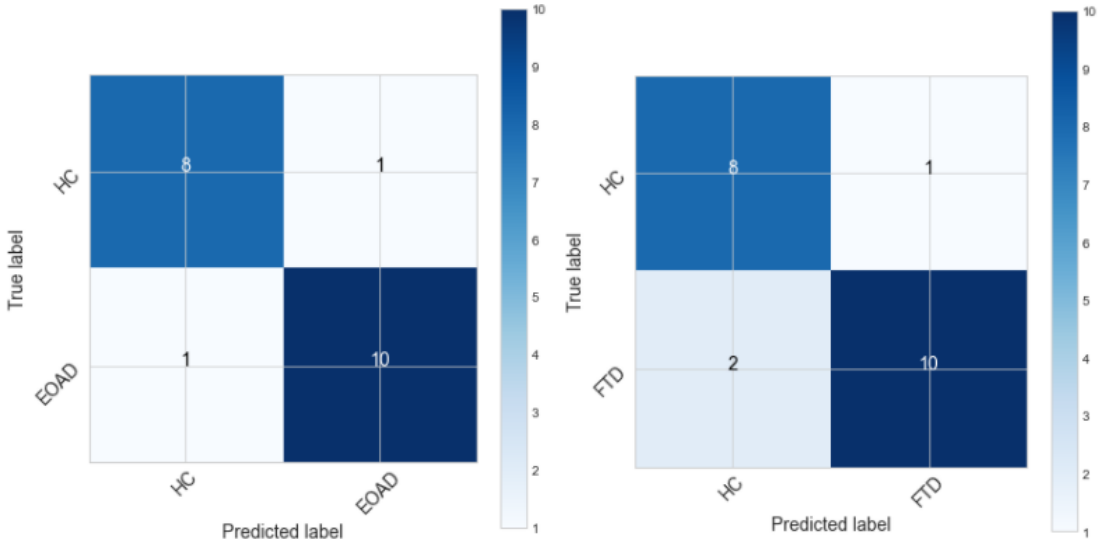
The results obtained could be separated in two different categories: a first one regarding the achieved algorithm performance and a second one concerning all the clinical information that resulted from the exhaustive analysis realised to the implemented algorithm.

#### 5.3.1. Algorithm performance

As mentioned, the number of PC to be used in each classification study was assessed with the cumulative explained variance. The results were shown in Figure 8. The number of PC used were 11, 9, 10 and 10 for the HC vs EOAD, HC vs FTD, EOAD vs FTD and the multiclass studies, respectively. The classification capability of the pipeline implemented was obtained by assessing its performance using the 20% of the data for testing in a 5-fold cross-validation. For each study (HC vs EOAD, HC vs FTD, EOAD vs FTD and HC vs EOAD vs FTD), the specific precisions for each group of subjects, as well as the overall classification accuracies, were calculated and the mean values were obtained across the 5 iterations. In Table 5, the total accuracy of each classification study, as well as each group precision, accompanied by the standard deviations, are shown. Also, Figure 11 presents the confusion matrices showing the mean of the true-predicted and false-predicted values for each study.

| | **STUDY** | | | |
|---|---|---|---|---|
| | **HC (Group 1) vs EOAD (Group 2)** | **HC (Group 1) vs FTD (Group 2)** | **EOAD (Group 1) vs FTD (Group 2)** | **HC (group 1) vs EOAD (group 2) vs FTD (group 3)** |
| **Mean accuracy** | 91.7% ± 5.8% | 83.3% ± 5.2% | 83.0% ± 5.8% | 77.7% ± 5.2% |
| **Precision group 1** | 92.8% ± 6.6% | 76.0% ± 4.7% | 81.79% ±10.5% | 70.7% ± 8.3% |
| **Precision group 2** | 91.0% ± 5.9% | 89.9% ± 6.6% | 86.3% ± 5.8% | 80.8% ± 4.2% |
| **Precision group 3** | ---------- | ---------- | ---------- | 83.3% ± 10.4% |

***Table 5:*** *accuracies mean values (%) and standard deviations (%) obtained for each classification study (HC vs EOAD, HC vs FTD, EOAD vs FTD and HC vs EOAD vs FTD). Also, the mean classification precision (%) of each individual group and the corresponding standard deviations (%) are shown.*



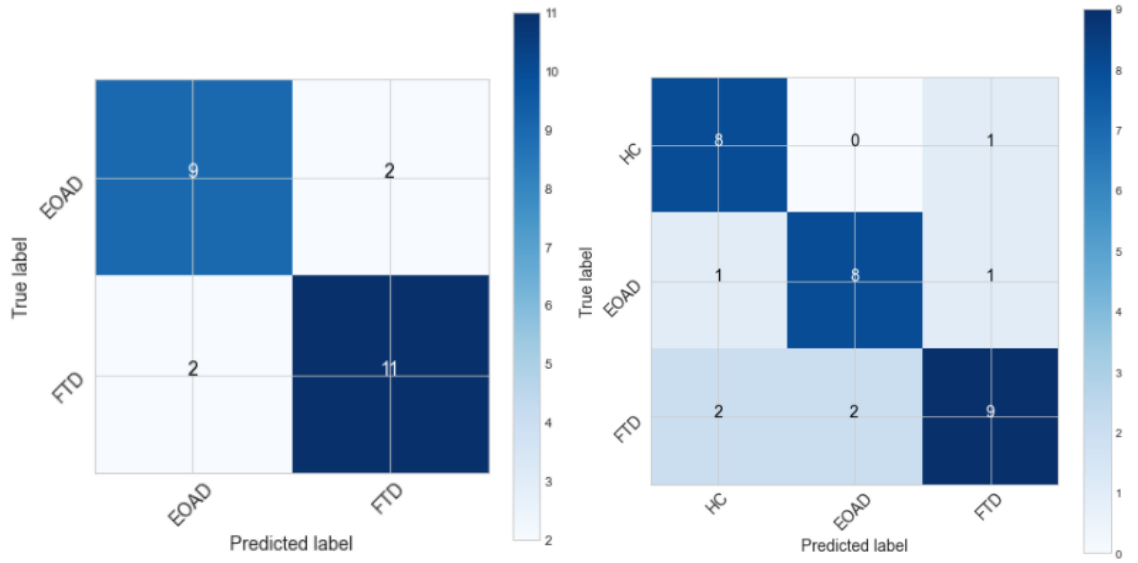



***Figure 11***: confusion matrices showing the classification precision for each group in the four studies. Top left: HC vs EOAD. Top right: HC vs FTD. Bottom left: EOAD vs FTD. Bottom right: HC vs EOAD vs FTD. The values correspond to the rounded means across the 5 iterations.

### 5.3.2. Significant brain regions

As already mentioned, the first principal component was used to retrieve the linear combination of features accumulating the highest variance from the original dataset. In such linear combination, features with the higher weights correspond to the ones contributing more to the variance accumulated by that PC. In Figures 12 to 15, the weights given by PC1 to each brain region are represented in four boxplots, each one corresponding to a different classification study.

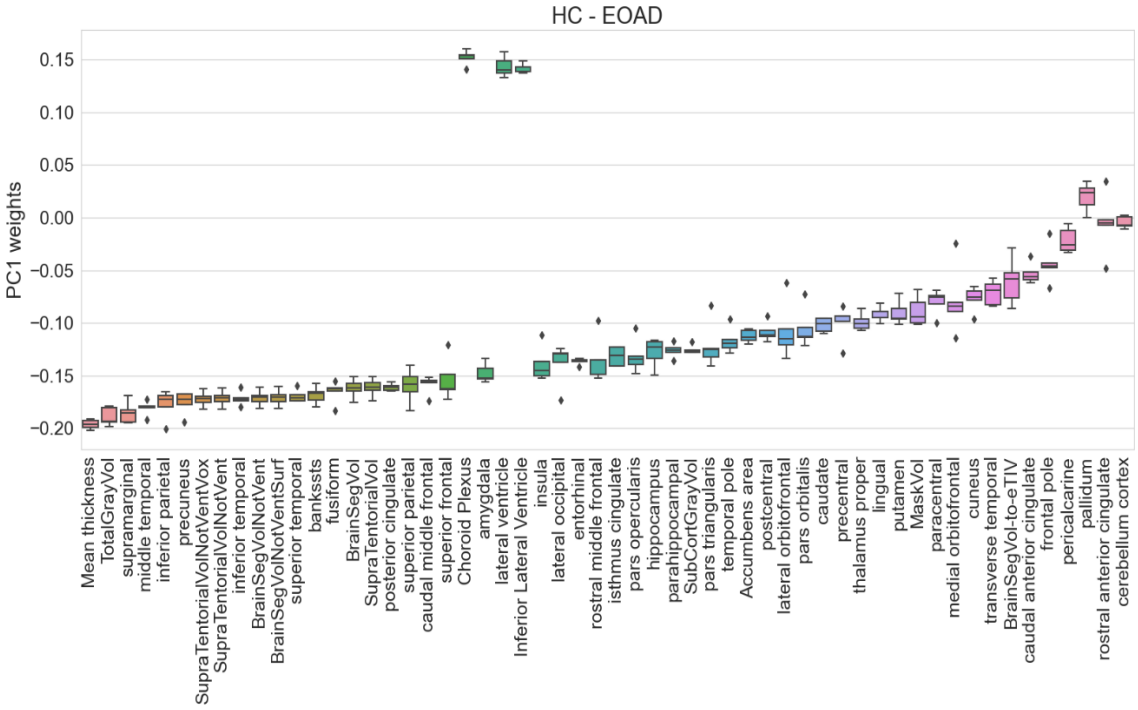



***Figure 12.*** *Boxplot showing the weights given to every feature by the first principal component (PC1) in HC and EOAD study, the mean and standard deviations across the 5 cross-validation iterations are represented. The features are sorted from higher to lower mean weights. In the x axis the names of the features are shown.*

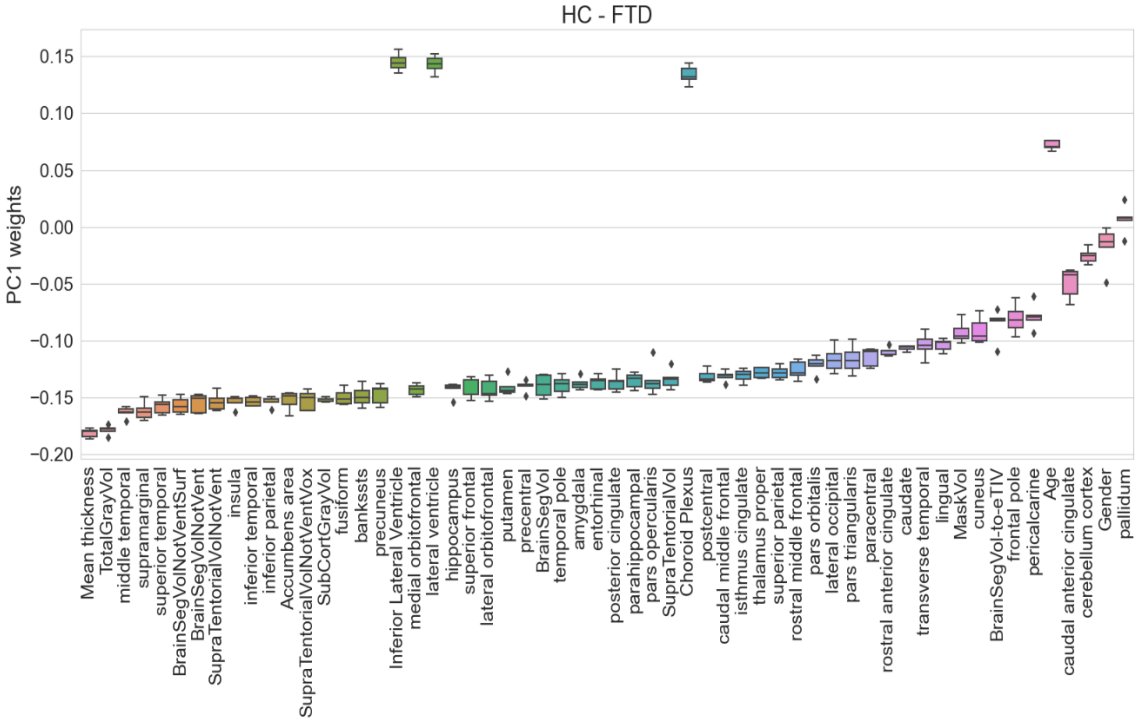



***Figure 13.*** *Boxplot showing the weights given to every feature by the first principal component (PC1) in HC and FTD study, the mean and standard deviations across the 5 cross-validation iterations are represented. The features are sorted from higher to lower mean weights. In the x axis the names of the features are shown.*

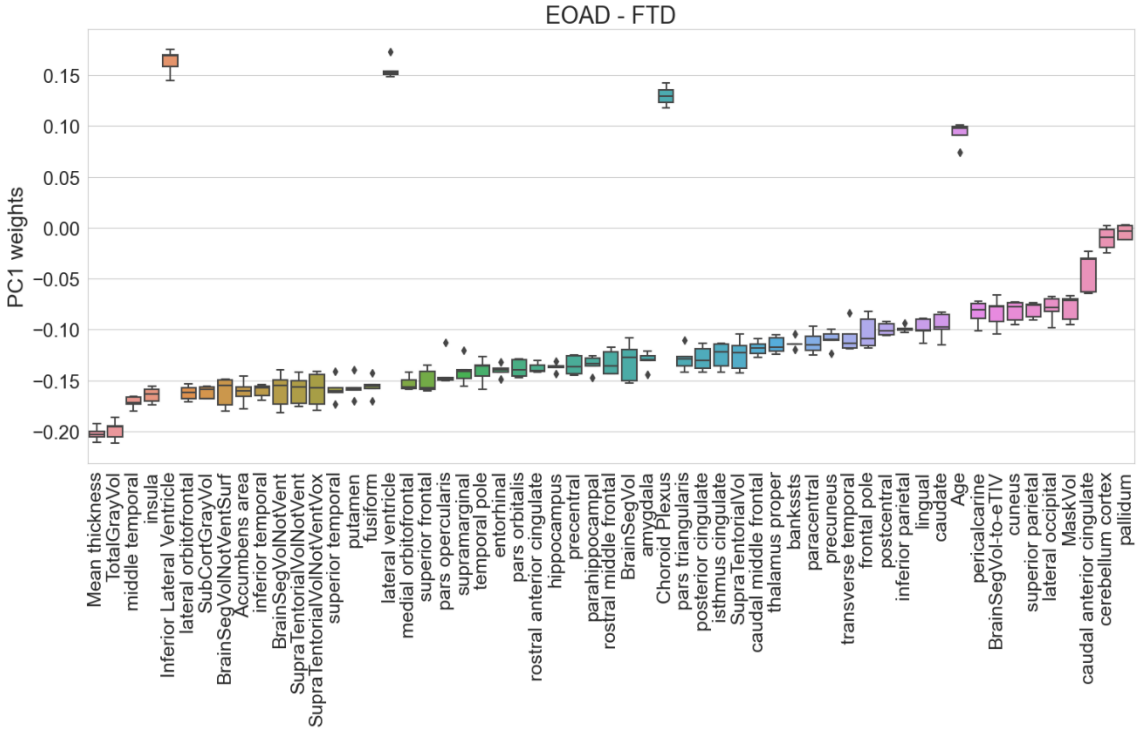



***Figure 14.*** *Boxplot showing the weights given to every feature by the first principal component (PC1) in EOAD and FTD study, the mean and standard deviations across the 5 cross-validation iterations are represented. The features are sorted from higher to lower mean weights. In the x axis the names of the features are shown.*

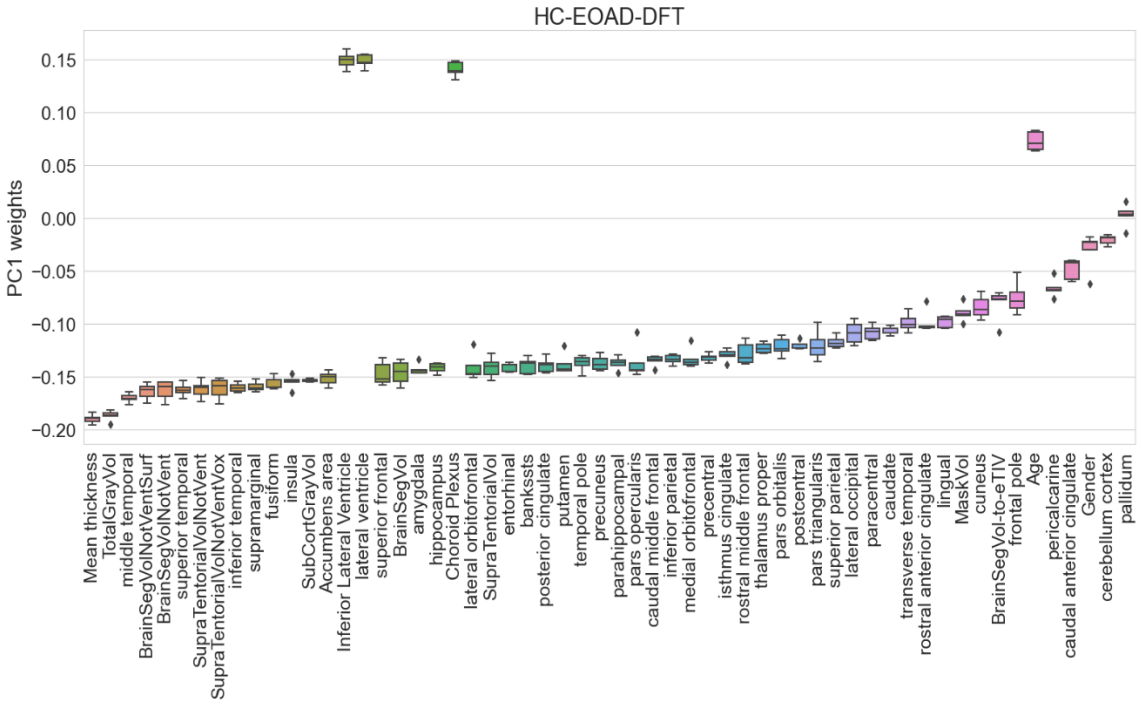



***Figure 15.*** *Boxplot showing the weights given to every feature by the first principal component (PC1) in HC, EOAD and FTD study, the mean and standard deviations across the 5 cross-validation iterations are represented. The features are sorted from higher to lower mean weights. In the x axis the names of the features are shown.*

In Figures 12 to 15, it can be noticed that most of the features were given negative weights and just a few of them, such as the ventricles size and volume or the choroid plexus were positively weighted. However, the sign of the weight given to a feature is just indicating the sign of its correlation with the principal component. That is, if an increase in a variable results in an increase of a PC and vice versa, it means that they are positively correlated and therefore the variable is positively weighted. And, on the contrary, a negative weight given to a variable means that it is negatively correlated with the PC. The larger a variable's weight, the more that variable is contributing to that PC.

The weights given to each variable were then plotted into a brain plot with the *ggseg* library from R studio. Since the sign of the weights is not relevant to the variable's contribution to the PC accumulated variance, the weights were represented in absolute value. The resulting cerebral plots show the relevance of each brain region when differentiating between the classified groups. Two different plots were made for each classification study, one of them showing the subcortical volumes (Aseg atlas) and another one showing the cortical and subcortical thicknesses (Desikan-Killiany or DK atlas). The Aseg atlases are shown in figure 16 and the DK atlases in Figure 17.

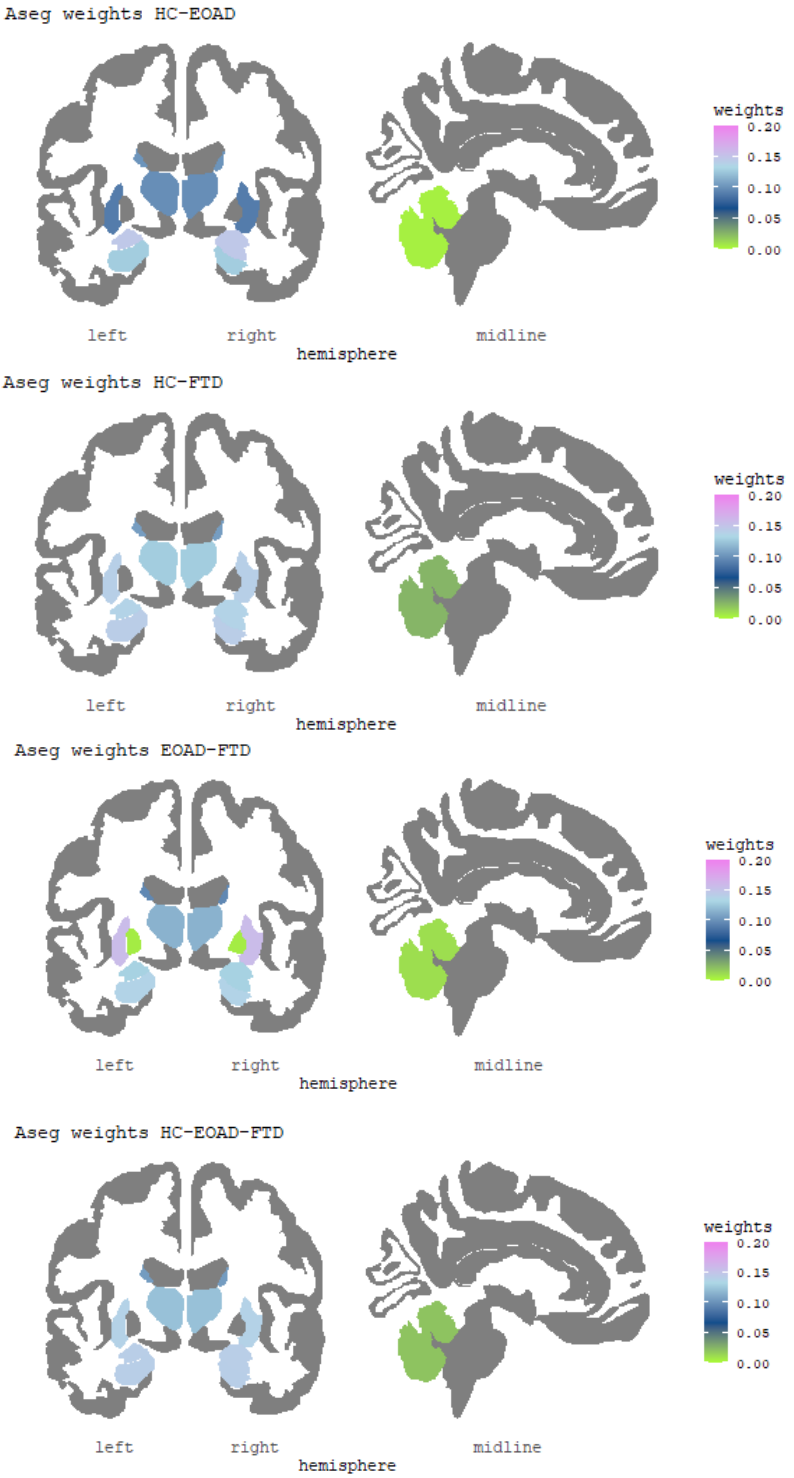



***Figure 16:*** *Aseg atlases of the subcortical brain volumes painted according to the weights, in absolute value, given by the first principal component of PCA in each of the 4 studies. 1) HC vs EOAD, 2) HC vs FTD, 3) EOAD vs FTD and 4) HC vs EOAD vs FTD.*

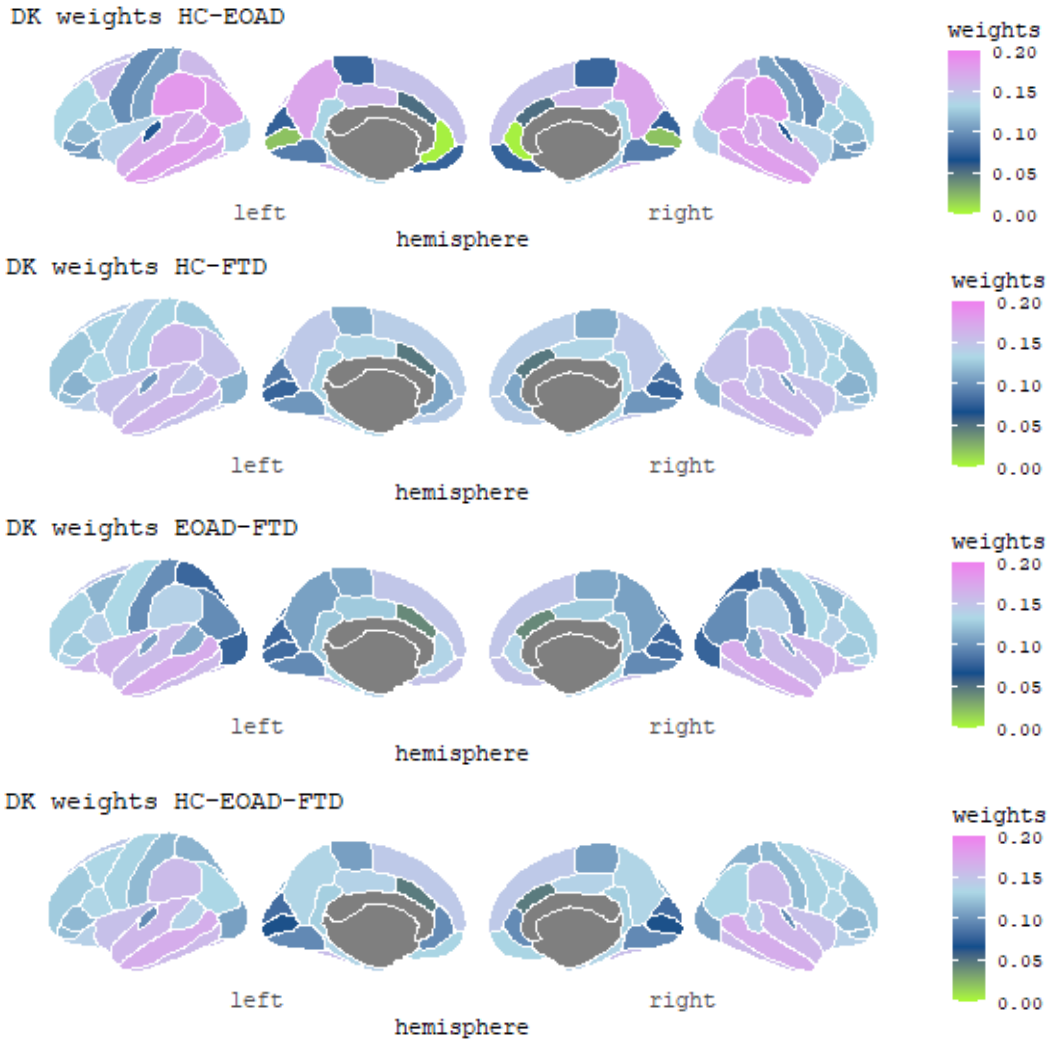



***Figure 17**: Desikan-Killiany atlases of the cortical and subcortical thicknesses of the brain painted according to the weights, in absolute value, given by the first principal component of PCA in each study.*

### 5.4. Discussion of the results

For each binary classification study, the probability of randomly predicting correctly a test subject was of 50%, whereas in the multiclass study that probability was of 33.3%. Thus, these last two values were taken as the accuracy baselines for the binary and multiclass studies, respectively.

By looking at the mean accuracies in Table 5, it can be seen that the binary classification with better performance was the HC vs EOAD one, improving the baseline estimation in a 41.74%. The high accuracies here might be because the atrophy patterns between EOAD and HC patients are clearly distinguishable. Also, when comparing the HC and EOAD individual precisions in Figure 11, it was observed that both groups were equally identified, indicating that the percentage of subjects correctly identified as EOAD patients was similar to the one of HC. The same happened for the misclassification of both groups.

In the classification between HC and FTD, the overall accuracy achieved was lower, meaning that the number of misclassified subjects was higher than in the previous study. This is probably due to the fact that the cerebral atrophy of FTD patients does not follow a pattern as specific as the one of EOAD, leading to greater difficulty in differentiating them from another group of patients. By

taking a closer look at the results, we observe that the number of subjects correctly classified as FTD was quite higher than the number of correctly classified HC. Thus, the probability of misclassifying a HC subject is higher than misclassifying a subject the other way around. However, both precisions exceeded 75%, which led to a reliable classification.

On the other hand, the binary study between EOAD and FTD patients showed similar results to the one between HC and FTD, with an overall accuracy of 83%. In this case, the lower accuracy might be as well due to the more diffuse atrophy pattern of the FTD subjects. Again, the probability of misclassifying a EOAD patient into FTD is higher than in the opposite direction. However, the difference in the precisions between the EOAD and FTD groups is lower than in the HC vs FTD study due to higher variance in the data .

Regarding the proposed multiclass study between all three groups, it improved the performance with respect to the 33.33% baseline in a 44.39%, outperforming with respect to binary studies. Also, when looking at the individual precisions of each group, the FTD was once again the group with the lower number of misclassifications. The reason keeps being its less defined atrophy pattern, that leads to the algorithm to confuse atrophy patterns from other groups with the one of FTD.

As can be seen in Figures from 12 to 15, the mean cortical thickness in the brain and the total gray volume were the two more weighted features of the dataset in all studies, meaning that were the ones contributing more to the PC1 variance. This proved that in both EOAD and FTD diseases there is an important gray matter loss with respect to healthy subjects. The fact that in the EOAD and FTD binary study these two features also accumulated variance might be because in FTD the amount of gray matter degeneration is not as much as in EOAD patients.

By taking a look at the features contributing with more variance in the HC and EOAD patients (Figure 12), besides from the mean cortical thickness and the total gray volume, the specific regions showing more variance were the supramarginal, middle temporal and inferior parietal thicknesses. Also, a highly weighted feature was the supra tentorial volume, which is a feature that includes all the pial structures as well as the hippocampus and amygdala, among others. The EOAD pattern found highly correlates with the one found in [51] by Möller et al.

When looking at Figure 13, it was observed that, compared to EOAD, FTD subjects also have a degeneration in nucleus accumbens, orbitofrontal areas and frontal insula in addition to a general cortical and subcortical loss of GM. This was also observed in Figure 14, where EOAD and FTD study results are shown. In this group differentiation, frontal insula plays an important role, which was also reported in the literature reviewed [52][53].

Moreover, Figure 15 shows the features contributing with more variance between the three groups. Once more, both the mean cortical thickness and the subcortical GM loss became the features accumulating more variance, followed by the middle temporal thickness and the total brain volume, which indicated a clear volume reduction in both neurodegenerative diseases with respect to controls.

Finally, the general patterns of both diseases can be observed in Figures 16 and 17. From these plots, it was confirmed the more specific degenerative pattern of EOAD patients with respect to FTD. EOAD showed a more focused GM loss in temporal and posterior areas, while FTD subjects showed a more diffuse pattern, with weights more evenly distributed in the different regions.

## 6. EXECUTION SCHEDULE: GANTT DIAGRAM

In order to develop the project in the expected time frame, from February 2021 to June 21th 2021, the different tasks to be carried out for the objectives accomplishment were listed and a specific time was assigned to each of them. In Figure 19, the Gantt diagram with the estimated times for each of the tasks is presented.

The tasks to be performed in order to develop the project in the expected time frame were divided in six blocks:

**6.1. Planning.** This block corresponds to the first two weeks of the project, since the first meeting with the tutor takes place in January 26 until the scope of the project is fully defined.

**6.1.1.** Fix the objectives: the most important milestones to be fulfilled at the end of the project are defined. Checkpoint: list with the definitive objectives.

**6.1.2.** Task's definition: definition of the different tasks to be carried out in order to achieve the previously listed objectives. The realization of the Gantt chart is included in this sub-block. Checkpoint: list with the ordered tasks and definitive Gantt chart.

**6.2. Literature review**. An in-depth research is conducted in different studies that have carried out ML approaches for the diagnosis of AD or FTD using MRI data or others such DTI data. This task is expected to last from February to mid-April.

**6.2.1.** Literature review. The studies are searched and collected if they meet the inclusion criteria. This subtask lasts the same as the whole task 2 because of possible new publications. Checkpoint: table with the studies reviewed and their main characteristics.

**6.2.2.** Evaluate feature selection methods. The feature selection methods and dimensionality reduction techniques used in the gathered papers are reviewed and the most appropriate technique for the project data is chosen. Checkpoint: feature selection/dimensionality reduction method chosen.

**6.2.3.** Evaluate Machine Learning algorithm. Once with the feature selection methods evaluated, in mid-March, the same subtask must be performed for the ML classification algorithm. Checkpoint: ML classification algorithm chosen.

**6.2.4.** Evaluate cross-validation method. The same review is done for the cross-validation method that aims to evaluate the algorithm performance. The subtask is expected to be finished by April 14. Checkpoint: cross-validation method chosen.

**6.3. Machine Learning model.** The objective of this task, to be performed from mid-February for three months, is to code the ML pipeline.

**6.3.1.** Model building. The first subtask is to build the model pipeline, with the feature selection, the ML algorithm and the cross-validation steps. Checkpoint: pipeline coded and implemented.

**6.3.2.** Model training and testing. Then, the model must be trained and tested with the available data and the chosen cross-validation method. Checkpoint: algorithm trained and tested with the cross-validation method chosen.

**6.3.3.** Algorithm adjustment. Once the model was trained and tested, depending on the outcomes obtained the appropriate corrections and adjustments must be done. This subtask will close the ML model task approximately in mid-May. Checkpoint: readjusted pipeline.

**6.4. Results.** With the algorithm already adjusted, it is time to present and analyse the results obtained. This aims to be done during the last three weeks of the project development, not considering the project delivery week.

**6.4.1.** Present the results. The results must be analyzed and presented in the most appropriate way, with the help of tables and graphs that clarify them. Checkpoint: results understood and formally presented.

**6.4.2.** Discussion of the results. With the results clearly presented, in mid-May, they must be discussed by performing an in-depth analysis. Checkpoint: section of the result's discussion finished.

**6.4.3.** Conclusion. Finally, while the results discussion is being done, conclusions must be extracted and expressed in the report. Checkpoint: conclusion section finished.

**6.5. Closeout.** This stage closes the project development in the last week, from June 7 until June 21.

**6.5.1.** Submission of the project. The project memory must be submitted on June 14.

**6.5.2.** Presentation of the project. The project must be presented and defended on June 22.

| PROJECT TASK | Start | Due |
|---|---|---|
| 1. PLANNING | 26/01/21 | 14/02/21 |
| 1.1 Fix the objectives | 26/01/21 | 05/02/21 |
| 1.2 Tasks definition | 26/01/21 | 14/02/21 |
| 2. LITERATURE REVIEW | 03/02/21 | 14/04/21 |
| 2.1 Literature review | 03/02/21 | 14/04/21 |
| 2.2 Evaluate feature selection method | 03/02/21 | 07/03/21 |
| 2.3 Evaluate Machine Learning algorithm | 07/02/21 | 01/04/21 |
| 2.4 Evaluate cross-validation method | 01/03/21 | 14/04/21 |
| 3. MACHINE LEARNING MODEL | 15/02/21 | 14/05/21 |
| 3.1 Model building | 15/02/21 | 15/04/21 |
| 3.2 Model training and testing | 15/03/21 | 21/04/21 |
| 3.3 Algorithm adjustment | 22/04/21 | 14/05/21 |
| 4. RESULTS | 01/05/21 | 07/06/21 |
| 4.1 Present results | 07/05/21 | 21/05/21 |
| 4.2 Discussion | 15/05/21 | 07/06/21 |
| 4.3 Conclussion | 15/05/21 | 07/06/21 |
| 5. CLOSEOUT | 07/06/21 | 21/06/21 |
| 5.1 Submission of the project | 07/06/21 | 14/06/21 |
| 5.2 Presentation of the project | 22/06/21 | |

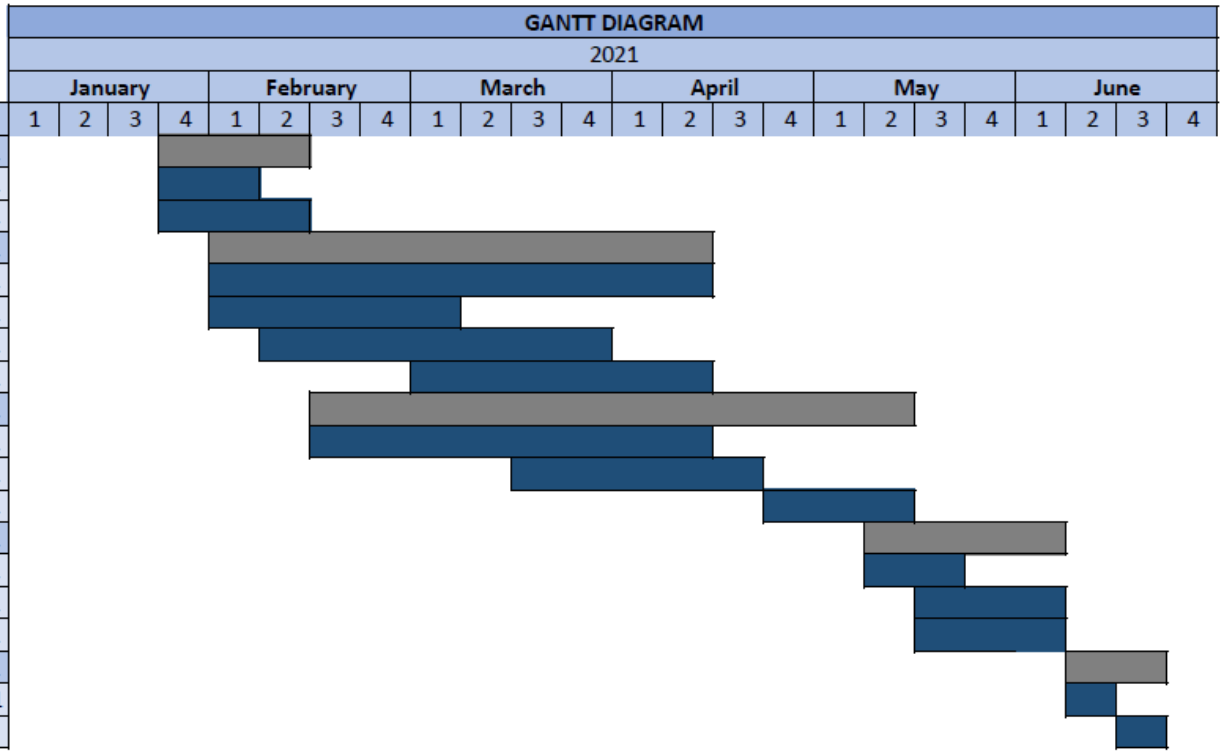



***Figure 18:*** *GANTT. Execution schedule by weeks of the project. On the left column the different tasks to be performed along the project execution (February 2021 – June 2021) are listed. The start and the due dates of each task are shown.*

## 7. TECHNICAL FEASABILITY

In this section, the SWOT analysis was developed. Both internal (strengths and weaknesses) and the external (opportunities and threads) factors affecting the current project were analysed.

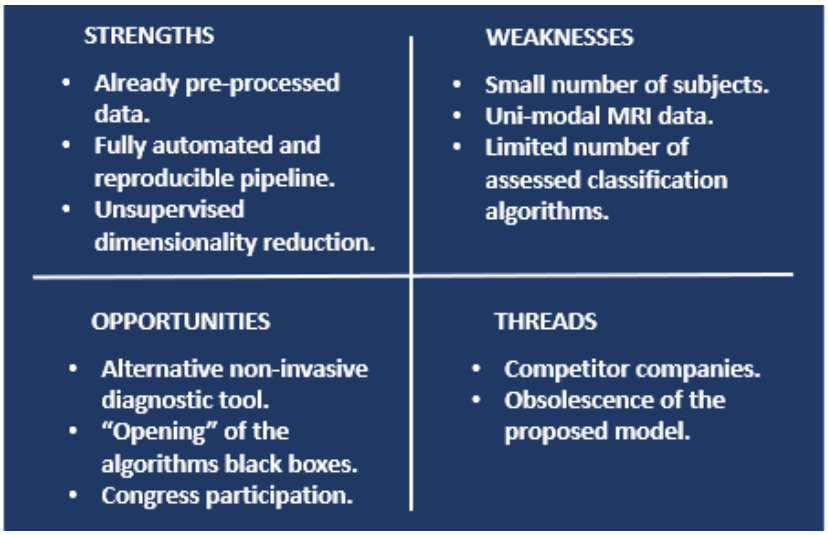



***Figure 19**: SWOT analysis of the project. Strengths, Weaknesses, Opportunities and Threads are listed.*

### 7.1 Strengths

Firstly, the data used in this study was already pre-processed and labelled. Therefore, since all the features were already extracted from the MRI scans, the focus of the project could be placed on the ML classification and dimensionality reduction algorithms.

Secondly, the classification pipeline implemented in Python was fully automated. It could perform a dimensionality reduction step of the data, plotting the brain regions providing more variance to the dataset, fitting the classification algorithm to the selected data, and to automatically cross-validate the model and present its performance. At the end, the resulting pipeline provided reproducible results, which allows for applying the model to other datasets different than ours or being used in other centres. Moreover, it could be used with other neuroimaging modalities alone or combined with structural MRI.

Finally, an unsupervised learning algorithm, PCA, was used to extract the most relevant brain regions and plot them in a brain atlas. This is a real advantage, since it provides a more clinical view of the problem, also allowing for tracking the disease evolution pattern in the future when new data of the patients is obtained.

### 7.2 Weaknesses

The first weakness this project encountered was the lack of data. It counted with a relatively small number of subjects, 161 (44 HC, 53 EOAD patients and 64 FTD patients), while to improve the results of the study more subjects should be included. Also, the data used belonged to a single neuroimaging modality, structural MRI, which also limited the algorithm performance. Therefore, better results would be expected if DTI or PET data had been available.

Another weakness is the number of classification algorithms implemented, which was limited by the time available to carry out the project. Only SVM was assessed for the classification, while other

algorithms that proved promising results in the literature, such as Artificial Neural Networks or Regularised Extreme Machine Learning were not implemented.

### 7.3 Opportunities

The first opportunity is that ML was recently proposed as an alternative diagnostic tool for neurodegenerative diseases by means of neuroimaging data. It allows for better precision in diagnostics but also leads to explainable systems able to recognize patterns associated with healthy or pathological brain structures. All of this makes these techniques suitable for clinical applications and research.

Also, although a wide range of studies and commercial softwares which aim to classify with the best accuracy as possible were developed, the algorithms were mostly treated as black boxes and little research was done in retrieving the most relevant features for the classification. In this aspect, the proposed study has an open opportunity.

Finally, during the project development the 1st ISMRM Iberian Chapter Annual Meeting took place, and the current work, together with the doctoral student Agnès Pérez work, could be presented to the congress.

### 7.4 Threads' analysis

Of course, in case the research project ended up giving rise to a start-up, an important thread would be enterprises such as *qmenta* that commercialize very powerful AI algorithms for the diagnosis of neurological diseases using neuroimage. Therefore, the market competition would be one of the obstacles for the project.

However, the fact that nowadays machine learning techniques are shifting towards convolutional neural networks, that embed into a single algorithm both feature selection and classification methods, gives rise to another thread: the obsolescence of the presented algorithm.

## 8. ECONOMICAL FEASABILITY

Although it is a publicly funded research thesis, in this section, the theoretical cost of the project based on the GANTT diagram is going to be described. To do so, the costs of human resources, the data used, software and hardware were divided in different packages with an associated cost. A summary table of the costs is presented in Table 6.

| Package | Element of the package | Element's cost | Package cost |
|---|---|---|---|
| **Human resources** | | | |
| | Learning: 105 h | 20 €/h | |
| | Development: 180 h | 20 €/h | |
| | Close-out: 95 h | 20 €/h | |
| | | | **7.600 €** |
| **Data** | | | |
| | FreeSurfer segmentations | 0 € | |
| | Demographic data | 0 € | |
| | | | **0 €** |
| **Software** | | | |
| | Python | 0 € | |
| | R studio | 0 € | |
| | | | **0 €** |
| **Hardware** | | | |
| | Computer | 800 € | |
| | Consumed electricity | 21,98 € | |
| | | | **821,98 €** |
| **Total cost** | | | **8.421,98 €** |

***Table 6:*** *table showing the theoretical cost of the project. The costs are divided in 4 packages: human resources, data, software and hardware. Each package is formed by different elements, which costs are presented.*

Firstly, regarding **human resources**, the salary of the student should be taken into account. The project development is designed to cover approximately 380 hours of work, which can be divided into a learning stage (105h), encompassing both *planning* and *literature review*; a development stage (180h), including the *ML algorithm building* and *results* tasks; and a close-out stage (95h) including the *memory writing* and the *oral presentation* of the project. Nonetheless, the salary of the student in the different stages is the same: 20€/h. Thus, the cost of this package amounts to **7.600€**. In figure 21, the hours implemented by the student in the different stages are represented.

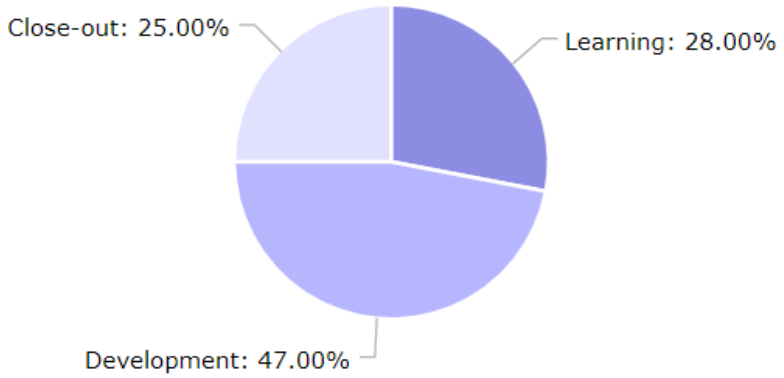



***Figure 20**: hours dedicated to each stage of the project by the student:.47% to the development, 28% to the learning stage and 25% to the close-out stage.*

Secondly, the images from which the data used in this project comes from were provided free of charge by the neurology service of Hospital Clinic. Moreover, the biomedicine department of the same hospital segmented the images with the FreeSurfer software. The resulting segmentations, as well as the corresponding demographic data, were also dispensed to the project free of charge. Thus, the overall cost of the **data** package is **0€**.

Thirdly, the **software** programs used in the development part were Python and R studio, both of them with completely free access, so the software package has no cost at all: **0€**.

Finally, an element to consider in the **hardware** package is the computer used, which is an HP pavilion of 2016 valued in approximately 800€. Also, the electricity consumption by the computer is considered. In all the hours of work the computer was used, and it consumes a mean of 400 W/h, so the electricity consumed for the project amounts to 152 kW. Taking into account that the price of electricity is around 0,14458 €/kW, the cost of electricity during the project was: 152 kW · 0,14458 €/kW = 21,98 €. Thus, the total cost of the hardware package is **821,98€.**

As can be seen in Table 6, the overall cost of the project is **8.421,98€**.

## 9. NORMATIVE AND LEGAL ASPECTS

This project is framed into the university, so it adheres to the University of Barcelona statute's [Estatut de la Universitat de Barcelona - Universitat de Barcelona], specifically to "*TÍTOL V*", chapters I, II and III. It states, among other remarkable points, that the research in University of Barcelona does not tolerate any project that does not aim to advance knowledge, improve life conditions, reduce social and economic inequalities and raise up innovation and business competitivity.

Moreover, as stated in section 5, the project presented uses data retrieved from T1-weighted MRI scans of real patients. All the scans were acquired in the diagnostic imaging centre of Hospital Clinic, thus adhering to the *bioethics committee* of the hospital. Also, all the patients involved in the study must have provided their consent in written, and their demographic and clinical information is protected by the *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales* of the BOE. This normative includes a section, called *"Disposición adicional decimoséptima. Tratamientos de datos de salud*", that states that the use of pseudonymised personal data for public health research is considered lawful as long as a technical separation exists between the research group and the group in charge of the pseudonymisation, so the real identifications remain unknown by the researchers.

The current project fits into the Computer-Aided Detection/Diagnosis Software category, since it aims to process information retrieved from MRI images to generate a diagnosis support. Therefore, the proposed pipeline must be implemented in accordance to *ISO 27799:2016 Health informatics – Information security management in health using ISO/IEC 27002*. This International Standards, similarly to *Ley Orgánica 3/2018*, gives guidelines to appropriately preserve the confidentiality of personal health information that can be used in any project, whatever the form that information takes: numbers, words or, as in the present case, medical images.

Finally, it cannot be omitted that the proposed algorithm aims to apply Artificial Intelligence to an actual medical diagnosis. However, although one would expect to find many regulations about it, there are currently no harmonized standards or laws that specifically regulate the use of Machine Learning or other forms of AI in the medical practice [54]. Nonetheless, Food and Drinks Administration (FDA) recently published a paper [55] where it describes a new potential approach to a regulation for AI and ML-driven software as medical devices. From this paper it can be derived that FDA would be enabled to evaluate and monitor a new software product from its premarket development to its postmarked performance to ensure patient's safety.

## 10. CONCLUSIONS AND FUTURE LINES

In conclusion, the presented project highlighted the importance of Artificial Intelligence algorithms for diagnostic imaging support. Specifically, the proposed algorithm showed the notable utility of machine learning automatic classification in the neuroimaging field, allowing to support the diagnosis of AD and other neurodegenerative diseases such as FTD. The main goals were achieved regarding both the algorithm performance and the retrieval of the atrophy patterns in both diseases. In addition, the algorithm proposed was accepted in the ISRMRM Iberian annual congress, which takes place on June 16 and 17.

In light of the results obtained, we could affirm that EOAD's patients present a more characteristic and less spread degenerative brain pattern than FTD patients, showing higher degeneration in temporal and posterior regions. This led to better classification performances when being classified versus control subjects. However, the algorithm improved the baseline performances (i.e., the ones obtained with a random classification of subjects) in more than a 30% in all the four studies, reaching a 44% of improvement in the multiclass classification between HC, EOAD and FTD controls, which is of great medical interest.

It is relevant to mention that the pipeline proposed presented a clear advantage with respect to many of the classification algorithms reviewed. While in most studies the algorithms used were treated as black boxes, by performing an exhaustive analysis of PCA, the presented work was able to retrieve the brain regions that accumulated the highest variance of the dataset containing the subject groups to be classified. Since PCA is an unsupervised method, this milestone allows to study the neurodegenerative pattern of an unlabelled dataset where the different groups where a subject might belong are known, in this case HC, AD or FTD. Moreover, if data from the same subjects are collected throughout the disease, the brain signatures plotted become of strong interest for studying the neurodegenerative evolution of both AD and FTD diseases.

Despite the accuracy of the presented pipeline reached optimal performance levels, it could be still improved. One way of doing so could be to feed the algorithm with a much larger dataset, leading to a better training and consequently to better results. Another interesting future implementation would be to both train and test the algorithm with other MRI modalities, such as DTI, so that other characteristic features, like white matter loss, were considered. Also, the current features could be fused with non-imaging data, as fluid biomarkers either from blood or CSF. Lastly, the proposed algorithm might be useful for the diagnosis of other neurodegenerative diseases such as Parkinson. Thus, future work could be made on the code in order to classify with higher accuracies between EOAD and FTD patients, and also to distinguish other diseases patients.

To conclude, this project is another proof of concept of how medicine is moving towards a massive digitalization, improving speed and quality of healthcare. In this regard, biomedical engineering becomes a key character which still has a lot to offer.

## 11. REFERENCES


[1] National Institute on Aging. (2019, May 22nd). *Alzheimer's Disease Fact Sheet*. https://www.nia.nih.gov/health/alzheimers-disease-fact-sheet

[2] Nussbaum, R.L., & Ellis, C. E. (2003). Alzheimer's Disease and Parkinson's Disease. *New England Journal of Medicine, 348(14),* 1356-1364. https://www.nejm.org/doi/full/10.1056/NEJM2003ra020003 .

[3] World Health Organization. (2020, September 21st). *Dementia.* https://www.who.int/news-room/fact-sheets/detail/dementia#:~:text=Worldwide%2C%20around%2050%20million%20people%20have%20dementia%2C%20with%20nearly%2060,is%20between%205%2D8%25.

[4] National Institute on Aging. (2019, June 5th). *Alzheimer's Disease Diagnostic Guidelines.* https://www.nia.nih.gov/health/alzheimers-disease-diagnostic-guidelines

[5] National Institute on Aging. (2011, April 19th). *Alzheimer's diagnostic guidelines updated for first time in decades*. https://www.nia.nih.gov/news/alzheimers-diagnostic-guidelines-updated-first-time-decades

[6] McKhann, G. M., *et al*. (2011). The diagnosis of dementia due to Alzheimer's disease: Recommendations from the National Institute on Aging-Alzheimer's Association workgroups on diagnostic guidelines for Alzheimer's disease. *Alzheimer's & Dementia, 7*(3), 263–269. https://doi.org/10.1016/j.jalz.2011.03.005 .

[7] Lama, R. K., Gwak, J., Park, J. S., & Lee, S. W. (2017). Diagnosis of Alzheimer's Disease Based on Structural MRI Images Using a Regularized Extreme Learning Machine and PCA Features. *Journal of Healthcare Engineering*, *2017*, 1–11. https://doi.org/10.1155/2017/5485080

[8] Kloppel, S., *et al*. (2008). Automatic classification of MR scans in Alzheimer's disease. *Brain, 131*(3), 681–689. https://doi.org/10.1093/brain/awm319

[9] Fischl, B. (2012). FreeSurfer. *NeuroImage, 62*(2), 774–781. https://doi.org/10.1016/j.neuroimage.2012.01.021

[10] Falahati, F., Westman, E., & Simmons, A. (2014). Multivariate Data Analysis and Machine Learning in Alzheimer's Disease with a Focus on Structural Magnetic Resonance Imaging. *Journal of Alzheimer's Disease, 41*(3), 685–708. https://doi.org/10.3233/jad-131928

[11] HLEG, A. I. (2019). Ethics guidelines for trustworthy AI. B-1049 Brussels. https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

[12] AI Watch Historical Evolution of Artificial Intelligence. (2020). *JRC TECHNICAL REPORTS*. Published. https://doi.org/10.2760/801580

[13] Saravanan, R., & Sujatha, P. (2018). A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification. *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. Published. https://doi.org/10.1109/iccons.2018.8663155

[14] Gupta, Y., Lee, K. H., Choi, K. Y., Lee, J. J., Kim, B. C., & Kwon, G. R. (2019b). Alzheimer's Disease Diagnosis Based on Cortical and Subcortical Features. *Journal of Healthcare Engineering, 2019, 1–13*. https://doi.org/10.1155/2019/2492719

[15] Duara, R., Barker, W., & Luis, C. A. (1999). Frontotemporal Dementia and Alzheimer's Disease:Differential Diagnosis. *Dementia and Geriatric Cognitive Disorders, 10*(1), 37–42. https://doi.org/10.1159/000051210

[16] Fletcher, E., *et al*. (2018). Brain volume change and cognitive trajectories in aging. *Neuropsychology, 32*(4), 436–449. https://doi.org/10.1037/neu0000447

[17] Farooq, A., Anwar, S., Awais, M., & Rehman, S. (2017). A deep CNN based multi-class classification of Alzheimer's disease using MRI. *2017 IEEE International Conference on Imaging Systems and Techniques (IST)*. Published. https://doi.org/10.1109/ist.2017.8261460

[18] Chincarini, A., *et al* (2011). Local MRI analysis approach in the diagnosis of early and prodromal Alzheimer's disease. *NeuroImage, 58*(2), 469–480. https://doi.org/10.1016/j.neuroimage.2011.05.083

[19] Lama, R. K., *et al*. (2017). Diagnosis of Alzheimer's Disease Based on Structural MRI Images Using a Regularized Extreme Learning Machine and PCA Features. Journal of Healthcare Engineering, 2017, 1–11. https://doi.org/10.1155/2017/5485080

[20] Davatzikos, C., Resnick, S., Wu, X., Parmpi, P., & Clark, C. (2008). Individual patient diagnosis of AD and FTD via high-dimensional pattern classification of MRI. NeuroImage, 41(4), 1220–1227. https://doi.org/10.1016/j.neuroimage.2008.03.050

[21] Möller, C., *et al*. (2016). Alzheimer Disease and Behavioral Variant Frontotemporal Dementia: Automatic Classification Based on Cortical Atrophy for Single-Subject Diagnosis. *Radiology, 279*(3), 838–848. https://doi.org/10.1148/radiol.2015150220

[22] Wolz, R., Julkunen, V., Koikkalainen, J., Niskanen, E., Zhang, D. P., Rueckert, D., Soininen, H., & Lötjönen, J. (2011). Multi-Method Analysis of MRI Images in Early Diagnostics of Alzheimer's Disease. *PLoS ONE, 6*(10), e25446. https://doi.org/10.1371/journal.pone.0025446

[23] Westman, E., *et al*. (2011). Multivariate analysis of MRI data for Alzheimer's disease, mild cognitive impairment and healthy controls. *NeuroImage, 54*(2), 1178–1187. https://doi.org/10.1016/j.neuroimage.2010.08.044

[24] Gupta, Y., *et al*. (2019). Early diagnosis of Alzheimer's disease using combined features from voxel-based morphometry and cortical, subcortical, and hippocampus regions of MRI T1 brain images. *PLOS ONE, 14*(10). https://doi.org/10.1371/journal.pone.0222446

[25] Bron, E. E., *et al*. (2016). Multiparametric computer-aided differential diagnosis of Alzheimer's disease and frontotemporal dementia using structural and advanced MRI. *European Radiology, 27*(8), 3372–3382. https://doi.org/10.1007/s00330-016-4691-x

[26] Zhang, D., Wang, Y., Zhou, L., Yuan, H., & Shen, D. (2011). Multimodal classification of Alzheimer's disease and mild cognitive impairment. *NeuroImage, 55*(3), 856–867. https://doi.org/10.1016/j.neuroimage.2011.01.008

[27] Chu, C., Hsu, A. L., Chou, K. H., Bandettini, P., & Lin, C. (2012). Does feature selection improve classification accuracy? Impact of sample size and feature selection on classification using anatomical magnetic resonance images. *NeuroImage, 60*(1), 59–70. https://doi.org/10.1016/j.neuroimage.2011.11.066

[28] Nho, K., Shen, L., Kim, S., Risacher, S. L., West, J. D., Foroud, T., Jack, C. R., Weiner, M. W., & Saykin, A. J. (2010). Automatic Prediction of Conversion from Mild Cognitive Impairment to Probable Alzheimer's Disease using Structural Magnetic Resonance Imaging. *AMIA ... Annual Symposium proceedings*. AMIA Symposium, 2010, 542–546.

[29] Torso, M., Bozzali, M., Cercignani, M., Jenkinson, M., & Chance, S. A. (2020). Using diffusion tensor imaging to detect cortical changes in fronto-temporal dementia subtypes. *Scientific Reports, 10*(1). https://doi.org/10.1038/s41598-020-68118-8

[30] Ma, D., Lu, D., Popuri, K., Wang, L., & Beg, M. F. (2020). Differential Diagnosis of Frontotemporal Dementia, Alzheimer's Disease, and Normal Aging Using a Multi-Scale Multi-Type Feature Generative Adversarial Deep Neural Network on Structural Magnetic Resonance Images. *Frontiers in Neuroscience, 14*. https://doi.org/10.3389/fnins.2020.00853

[31] Raamana, P. R., Rosen, H., Miller, B., Weiner, M. W., Wang, L., & Beg, M. F. (2014). Three-Class Differential Diagnosis among Alzheimer Disease, Frontotemporal Dementia, and Controls. *Frontiers in Neurology, 5*. https://doi.org/10.3389/fneur.2014.00071

[32] Zutshi Y. (2020). Alzheimer's Disease Therapeutics and Diagnostics: Global Markets. *BCC Research.* https://www.bccresearch.com/market-research/pharmaceuticals/alzheimers-disease-therapeutics-diagnostics-markets-report.html

[33] Davatzikos, C. (2019). Machine learning in neuroimaging: Progress and challenges. *NeuroImage, 197*, 652–656. https://doi.org/10.1016/j.neuroimage.2018.10.003

[34] QMENTA. (2021). *QMENTA Platform*. https://www.qmenta.com/qmenta-platform/

[35] Qubiotech. (2021). *Qubiotech | Bienvenido a la neuroimagen aumentada*. https://www.qubiotech.com/

[36] Python Software Foundation. Python Language Reference, version 3. Available at https://www.python.org/

[37] RStudio Team (2021). RStudio: Integrated Development for R. RStudio, PBC, Boston, MA. Version 1.4.1106. Available at http://www.rstudio.com/ .

[38] MATLAB. (2021). version R2021a. Natick, Massachusetts: The MathWorks Inc. Available at: https://es.mathworks.com/products/matlab.html

[39] Scikit-Learn library. *Python*. Available at: https://scikit-learn.org/stable/

[40] Statistics and Machine Learning Toolbox. *Matlab*. Available at: https://es.mathworks.com/products/statistics.html

[41] Kubat, M. (2017). An Introduction to Machine Learning (2nd 2017 ed.). Springer.

[42] *Linear Discriminant Analysis*. (2014, August 3rd). Dr. Sebastian Raschka. https://sebastianraschka.com/Articles/2014_python_lda.html

[43] Peng, Y., Wu, Z., & Jiang, J. (2010). A novel feature selection approach for biomedical data classification. *Journal of Biomedical Informatics, 43*(1), 15–23. https://doi.org/10.1016/j.jbi.2009.07.008

[44] Abu Alfeilat, H. A., Hassanat, A. B., Lasassmeh, O., Tarawneh, A. S., Alhasanat, M. B., Eyal Salman, H. S. y Prasath, V. S. (2019). Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review. *Big Data, 7* (4), 221–248. https://doi.org/10.1089/big.2018.0175

[45] Hehn, T. M., Kooij, J. F. P., & Hamprecht, F. A. (2019). End-to-End Learning of Decision Trees and Forests. *International Journal of Computer Vision, 128* (4), 997–1011. https://doi.org/10.1007/s11263-019-01237-6

[46] Yadav, S., & Shukla, S. (2016). Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification. *2016 IEEE 6th International Conference on Advanced Computing (IACC).* Published. https://doi.org/10.1109/iacc.2016.25

[47] Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374*(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

[48] Pisner, D. A., & Schnyer, D. M. (2020). Support vector machine. *Machine Learning*, 101–121. https://doi.org/10.1016/b978-0-12-815739-8.00006-7

[49] *Plot different SVM classifiers in the iris dataset — scikit-learn 0.18.2 documentation.* (2020). Sklearn. https://scikit-learn.org/0.18/auto_examples/svm/plot_iris.html

[50] *RBF SVM parameters — scikit-learn 0.24.2 documentation.* (2020). Scikit-Learn. https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

[51] Möller, C., et al. (2015). Joint assessment of white matter integrity, cortical and subcortical atrophy to distinguish AD from behavioral variant FTD: A two-center study. *NeuroImage: Clinical, 9*, 418–429. https://doi.org/10.1016/j.nicl.2015.08.022

[52] Rabinovici, G., et al. (2008). Distinct MRI Atrophy Patterns in Autopsy-Proven Alzheimer's Disease and Frontotemporal Lobar Degeneration. *American Journal of Alzheimer's Disease & Other Dementiasr, 22*(6), 474–488. https://doi.org/10.1177/1533317507308779

[53] Falgàs, N., et al. (2020). Contribution of CSF biomarkers to early-onset Alzheimer's disease and frontotemporal dementia neuroimaging signatures. *Human Brain Mapping, 41*(8), 2004–2013. https://doi.org/10.1002/hbm.24925

[54] Minssen, T., Gerke, S., Aboy, M., Price, N., & Cohen, G. (2020). Regulatory responses to medical machine learning. *Journal of Law and the Biosciences, 7*(1). https://doi.org/10.1093/jlb/lsaa002

[55] U.S. Food & Drug Administration. (2019) Proposed Regulatory Framework for Modifications to Artificial Intelligence/Machine Learning (AI/ML)-Based Software as a Medical Device (SaMD).

# 12. ANEXES

## 12.1. Python code: classification pipeline.

```
#---------------------------- IMPORTS ----------------------
#-----------------------------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn import svm
from sklearn.model_selection import GridSearchCV
from sklearn.decomposition import PCA
    import seaborn as sns
    from scipy.stats import zscore
    import itertools
    from sklearn.model_selection import StratifiedKFold
    import warnings
    warnings.filterwarnings("ignore")
    import os
    #set working directory with the data tables available
    os.chdir(r"C:\Users\Laia.LAPTOP-1BS820LB\Desktop\4t\TFG\CODI")

    #---------------------------- FUNCIONS ---------------------
    #-----------------------------------------------------------
    def dic_pc(component,feature_weights,features):
        '''
        Parameters
        ----------
        component : int
            number of components that want to be added
        feature_weights : dictionary, with num of components as
keys, another dictionary as value with features as keys and weights
as values
        features : list
            feature's names list
        Returns
        -------
        mwf_pc : dictionary
            name of features as keys and weight given to each
feature as values.(sorted from higer to lower weights)
        '''
        pc=feature_weights[component][0]
        pc_abs=abs(np.array(pc))
        sorted_pc=pc_abs
        sorted_pc=sorted_pc.tolist()
        sorted_pc.sort(reverse=True)
        mwf_pc={} #more weighted features: dictionary with features
as keys and weights as values
        for i in range(0,len(sorted_pc)):
            index=list(pc_abs).index(sorted_pc[i])
            mwf_pc[features[index]]=pc[index]
        return mwf_pc

    #PLOT BAR OF X FEATURES MORE WEIGHTED

    def plot_bar(l,n,component_dic):
        '''

        Parameters
        ----------
```

```
        l : int
            starting index number, indicates which feature you want
    to start
            plotting
        n : int
            stop index number, indicates which feature you want to
    stop plotting
        component_dic : dictionary
            dictionary with the features as keys and weights as
    vaules (sorted from higher to lower weights.)

        Returns
        -------
        None.

        '''
    blues=['steelblue','dodgerblue','deepskyblue','skyblue','lightskybl
    ue',
                'darkturquoise','paleturquoise']
        values=list(component_dic.values())[l:n]
        tags=list(component_dic.keys())[l:n]
        plt.bar(range(l,n),values,color=blues) #plotejo de la 1 a
    la n
        plt.xticks(np.arange(l,n),
    tags,rotation='vertical',fontsize=14)
        plt.yticks(fontsize=14)


    #PLOT CONFUSION MATRIX
    def plot_confusion_matrix(cm, classes,
                              normalize=False,
                              title='Confusion matrix',
                              cmap=plt.cm.Blues):
        """
        This function prints and plots the confusion matrix.
        Normalization can be applied by setting `normalize=True`.
        """
        if normalize:
            cm=cm.astype("float")
            cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
            print("Normalized confusion matrix")
        else:
            print('Confusion matrix, without normalization')
            cm=np.round(cm,decimals=0)
            cm=cm.astype(np.int64)
        print(cm)

        plt.imshow(cm, interpolation='nearest', cmap=cmap)
        plt.title(title,fontsize=16)
        plt.colorbar(shrink=1)
        tick_marks = np.arange(len(classes))
        plt.xticks(tick_marks, classes, rotation=45,fontsize=14)
        plt.yticks(tick_marks, classes,fontsize=14,rotation=45)

        fmt = '.2f' if normalize else 'd'
        thresh = cm.max() / 2.
        for i, j in itertools.product(range(cm.shape[0]),
    range(cm.shape[1])):
            plt.text(j, i, format(cm[i, j], fmt),
                     horizontalalignment="center",
```

```
                    color="white" if cm[i, j] > thresh else
   "black",fontsize=14)

        plt.tight_layout()
        plt.ylabel('True label',fontsize=14,labelpad=5)
        plt.xlabel('Predicted label',fontsize=14,labelpad=5)

    #--------------------------- ATLAS -------------------------
    #-----------------------------------------------------------

    #atlasses regions for the brain plot in R

    features_dk=["rh_middletemporal_thickness","rh_supramarginal_th
   ickness", "rh_superiortemporal_thickness","rh_insula_thickness",
   "rh_inferiortemporal_thickness","rh_inferiorparietal_thickness",
   "rh_fusiform_thickness","rh_bankssts_thickness",
   "rh_precuneus_thickness","rh_lateralorbitofrontal_thickness",
   "rh_medialorbitofrontal_thickness","rh_superiorfrontal_thickness",
   "rh_temporalpole_thickness","rh_precentral_thickness",
   "rh_entorhinal_thickness","rh_posteriorcingulate_thickness",
   "rh_parahippocampal_thickness","rh_parsopercularis_thickness",
   "rh_isthmuscingulate_thickness","rh_caudalmiddlefrontal_thickness",
   "rh_postcentral_thickness","rh_superiorparietal_thickness",
   "rh_rostralmiddlefrontal_thickness","rh_parsorbitalis_thickness",
   "rh_parstriangularis_thickness","rh_lateraloccipital_thickness",
   "rh_paracentral_thickness","rh_rostralanteriorcingulate_thickness",
   "rh_transversetemporal_thickness","rh_lingual_thickness",
   "rh_cuneus_thickness","rh_frontalpole_thickness",
   "rh_pericalcarine_thickness","rh_caudalanteriorcingulate_thickness"
   ]

    atlas_dk=["middle temporal","supramarginal","superior
   temporal","insula","inferior temporal","inferior
   parietal","fusiform","bankssts","precuneus","lateral
   orbitofrontal","medial orbitofrontal","superior frontal","temporal
   pole", "precentral","entorhinal","posterior cingulate",
   "parahippocampal","pars opercularis", "isthmus cingulate","caudal
   middle frontal", "postcentral", "superior parietal","rostral middle
   frontal","pars orbitalis","pars triangularis","lateral
   occipital","paracentral","rostral anterior cingulate", "transverse
   temporal", "lingual","cuneus", "frontal pole",
   "pericalcarine","caudal anterior cingulate"]

    features_aseg=["Norm-Hippocampus","Norm-Putamen","Norm-Lateral-
   Ventricle","Norm-Amygdala", "Norm-Thalamus-Proper","Norm-Caudate",
   "Norm-Cerebellum-Cortex","Norm-Pallidum"]

    atlas_aseg=["hippocampus","putamen","lateral
   ventricle","amygdala","thalamus proper","caudate","cerebellum
   cortex","pallidum"]

    #---------------- PARAMETERS (CHECK BEFORE RUN !!)---------
    #-----------------------------------------------------------

    #PARAMETERS TO SELECT
    N_splits=5 # number of folds, 5-fold CV
    estudi='AD' #'AD'--> AD vs CTR, 'DFT'--> DFT vs CTR, "ADDFT"-->
   AD vs DFT
    m="yes" #If documents want to be saved, 'yes', else 'no'
    plot=False #if pca plots want to be showed, then True
```

```
135.
136.    #---------------------------- DATA ---------------------------
137.    #-------------------------------------------------------------
138.
139.    # We import cth (corticalsd) and volumes (subcorticals) data
140.    if estudi=='AD':
141.        np.random.seed(10) #we place a seed to obtain reproducible
   results
142.        Ngroup1=44 # number of subjects from the first group
143.        Ngroup2=53  # number of subjects from the second group
144.        N=Ngroup1+Ngroup2 # total number of subjects
145.        df_rh=pd.read_csv("rh.tablecth.csv",sep='\t')
146.        df_rh=df_rh.drop(columns=['rh.aparc.thickness'])
147.        df_lh=pd.read_csv("lh.tablecth.csv",sep='\t')
148.        df_lh=df_lh.drop(columns=['lh.aparc.thickness'])
149.        df_subc=pd.read_csv("asegtable.csv",sep='\t')
150.
151.    if estudi=='DFT':
152.        np.random.seed(123) #we place a seed to obtain reproducible
   results
153.        Ngroup1=44 # number of subjects from the first group
154.        Ngroup2=64 # number of subjects from the second group
155.        N=Ngroup1+Ngroup2 # total number of subjects
156.        # AD + CTR
157.        df_rh1 = pd.read_csv("rh.tablecth.csv",sep='\t')
158.        df_rh1=df_rh1.drop(columns=['rh.aparc.thickness'])
159.        df_lh1=pd.read_csv("lh.tablecth.csv",sep='\t')
160.        df_lh1=df_lh1.drop(columns=['lh.aparc.thickness'])
161.        df_subc1=pd.read_csv("asegtable.csv",sep='\t')
162.
163.        #Eliminate AD
164.        df_rh1=df_rh1[:Ngroup1]
165.        df_lh1=df_lh1[:Ngroup1]
166.        df_subc1=df_subc1[:Ngroup1]
167.
168.        #DFT
169.        df_rh2 = pd.read_csv("rh.tablecthDFT.csv",sep=';')
170.        df_rh2=df_rh2.drop(columns=['rh.aparc.thickness'])
171.        df_lh2=pd.read_csv("lh.tablecthDFT.csv",sep=';')
172.        df_lh2=df_lh2.drop(columns=['lh.aparc.thickness'])
173.        df_subc2=pd.read_csv("asegtableDFT.csv",sep=';')
174.
175.        # we concatenate the data
176.        df_rh= pd.concat([df_rh1, df_rh2])
177.        df_lh= pd.concat([df_lh1, df_lh2])
178.        df_subc= pd.concat([df_subc1, df_subc2])
179.        df_rh=df_rh.reset_index()
180.        df_lh=df_lh.reset_index()
181.        df_subc=df_subc.reset_index()
182.        df_rh=df_rh.drop(['index'], axis=1)
183.        df_lh=df_lh.drop(['index'], axis=1)
184.        df_subc=df_subc.drop(['index'], axis=1)
185.
186.    if estudi=='ADDFT':
187.        np.random.seed(10) #we place a seed to obtain reproducible
   results
188.        Ngroup1=53 # number of subjects from the first group
189.        Ngroup2=64 # number of subjects from the second group
190.        N=117 # total number of subjects
191.        # AD + CTR
192.        df_rh1 = pd.read_csv("rh.tablecth.csv",sep='\t')
```

```
193.        df_rh1=df_rh1.drop(columns=['rh.aparc.thickness'])
194.        df_lh1=pd.read_csv("lh.tablecth.csv",sep='\t')
195.        df_lh1=df_lh1.drop(columns=['lh.aparc.thickness'])
196.        df_subc1=pd.read_csv("asegtable.csv",sep='\t')
197.
198.        df_rh1=df_rh1[44:44+Ngroup2]
199.        df_lh1=df_lh1[44:44+Ngroup2]
200.        df_subc1=df_subc1[44:44+Ngroup2]
201.
202.        df_rh1=df_rh1.reset_index()
203.        df_lh1=df_lh1.reset_index()
204.        df_subc1=df_subc1.reset_index()
205.
206.        df_rh1=df_rh1.drop(['index'], axis=1)
207.        df_lh1=df_lh1.drop(['index'], axis=1)
208.        df_subc1=df_subc1.drop(['index'], axis=1)
209.
210.        #DFT
211.        df_rh2 = pd.read_csv("rh.tablecthDFT.csv",sep=';')
212.        df_rh2=df_rh2.sort_values(["rh.aparc.thickness"], ascending
    = True)
213.        df_rh2=df_rh2.drop(columns=['rh.aparc.thickness'])
214.        df_lh2=pd.read_csv("lh.tablecthDFT.csv",sep=';')
215.        df_lh2=df_lh2.sort_values(["lh.aparc.thickness"], ascending
    = True)
216.        df_lh2=df_lh2.drop(columns=['lh.aparc.thickness'])
217.        df_subc2=pd.read_csv("asegtableDFT.csv",sep=';')
218.        df_subc2=df_subc2.sort_values(["Measure:volume"], ascending
    = True)
219.
220.        # we concatenate data
221.        df_rh= pd.concat([df_rh1, df_rh2])
222.        df_lh= pd.concat([df_lh1, df_lh2])
223.        df_subc= pd.concat([df_subc1, df_subc2])
224.        df_rh=df_rh.reset_index()
225.        df_lh=df_lh.reset_index()
226.        df_subc=df_subc.reset_index()
227.        df_rh=df_rh.drop(['index'], axis=1)
228.        df_lh=df_lh.drop(['index'], axis=1)
229.        df_subc=df_subc.drop(['index'], axis=1)
230.
231.    erase = ["Measure:volume", "EstimatedTotalIntraCranialVol",
    'Right-VentralDC', 'Right-vessel','WM-hypointensities','non-WM-
    hypointensities','Left-VentralDC','Left-vessel', 'Left-Cerebellum-
    White-Matter', 'lhCerebralWhiteMatterVol','Brain-Stem','3rd-
    Ventricle', 'Left-non-WM-hypointensities','Right-non-WM-
    hypointensities', '4th-Ventricle','5th-Ventricle', 'CSF',
    'CC_Anterior','CC_Central', 'CC_Mid_Anterior', 'CC_Mid_Posterior',
    'CC_Posterior','lhCortexVol','rhCortexVol', 'Optic-Chiasm','Right-
    Cerebellum-White-Matter', 'rhCerebralWhiteMatterVol',
    "CerebralWhiteMatterVol",'Left-WM-hypointensities','Right-WM-
    hypointensities',"CortexVol",'lhSurfaceHoles','rhSurfaceHoles',"Sur
    faceHoles"]

232.
233.    df_subc=df_subc.drop(columns=erase)
234.
235.    #----- MEAN BETWEEN LEFT AND RIGHT CORTICAL MEASURES -------
236.
237.    features_r=(df_rh.columns).to_list()
238.    norm_r_l=pd.DataFrame(index=np.arange(N))
```

```
    for i in range(0,35):
        col_r=(df_rh.iloc[:,i]).to_numpy()
        col_l=(df_lh.iloc[:,i]).to_numpy()
        mean=(col_r+col_l)/2
        norm=mean
        name=features_r[i]
        norm_r_l.insert(i,name,norm)
    norm_r_l.insert(35,'eTIV',df_rh.iloc[:,36])

    #------------- normalization + mean between left and right subcortical measures -------------------------

    #normalization to intracranial volume
    features_subc=(df_subc.columns).to_list()
    for f in features_subc:
        df_subc[f]=df_subc[f]/df_rh['eTIV']

    #mean between left and right measures
    hippo=df_subc[['Right-Hippocampus','Left-Hippocampus']]
    norm_hippo=hippo.mean(axis=1)
    df_subc['Norm-Hippocampus']=norm_hippo

    lat_ventr=df_subc[['Left-Lateral-Ventricle','Right-Lateral-Ventricle']]
    norm_lat_ventr=lat_ventr.mean(axis=1)
    df_subc['Norm-Lateral-Ventricle']=norm_lat_ventr

    inf_lat_ventr=df_subc[['Left-Inf-Lat-Vent','Right-Inf-Lat-Vent']]
    norm_inf_lat_ventr=inf_lat_ventr.mean(axis=1)
    df_subc['Norm-Inf-Lat-Ventr']=norm_inf_lat_ventr

    cer_cort=df_subc[['Left-Cerebellum-Cortex','Right-Cerebellum-Cortex']]
    norm_cer_cort=cer_cort.mean(axis=1)
    df_subc['Norm-Cerebellum-Cortex']=norm_cer_cort

    thal_pro=df_subc[['Left-Thalamus-Proper','Right-Thalamus-Proper']]
    norm_thal_pro=thal_pro.mean(axis=1)
    df_subc['Norm-Thalamus-Proper']=norm_thal_pro

    caudate=df_subc[['Left-Caudate','Right-Caudate']]
    norm_cau=caudate.mean(axis=1)
    df_subc['Norm-Caudate']=norm_cau

    putamen=df_subc[['Left-Putamen','Right-Putamen']]
    norm_put=putamen.mean(axis=1)
    df_subc['Norm-Putamen']=norm_put

    pallidum=df_subc[['Left-Pallidum','Right-Pallidum']]
    norm_pall=pallidum.mean(axis=1)
    df_subc['Norm-Pallidum']=norm_pall

    amyg=df_subc[['Left-Amygdala','Right-Amygdala']]
    norm_amyg=amyg.mean(axis=1)
    df_subc['Norm-Amygdala']=norm_amyg

    accumbens=df_subc[['Left-Accumbens-area','Right-Accumbens-area']]
    norm_acc=accumbens.mean(axis=1)
```

```
294.    df_subc['Norm-Accumbens-area']=norm_acc
295.
296.    c_p=df_subc[['Left-choroid-plexus','Right-choroid-plexus']]
297.    norm_cp=c_p.mean(axis=1)
298.    df_subc['Norm-choroid-plexus']=norm_cp
299.
300.    df_subc=df_subc.drop(columns=['Right-Hippocampus','Left-
   Hippocampus','Left-Lateral-Ventricle','Right-Lateral-Ventricle',
   'Left-Inf-Lat-Vent','Right-Inf-Lat-Vent', 'Left-Cerebellum-
   Cortex','Right-Cerebellum-Cortex','Left-Thalamus-Proper','Right-
   Thalamus-Proper','Left-Caudate','Right-Caudate','Left-Putamen',
   'Right-Putamen','Left-Pallidum','Right-Pallidum', 'Left-
   Amygdala','Right-Amygdala','Left-Accumbens-area','Right-Accumbens-
   area','Left-choroid-plexus', 'Right-choroid-plexus'])
301.
302.    df=pd.concat([df_subc, norm_r_l], axis=1)
303.    df=df.drop(['eTIV'], axis=1) #eliminem eTIV de les features
304.
305.    # we add demographic data if the study is not EOAD vs HC
306.
307.    if estudi=="DFT":
308.        df_demo1=pd.read_csv("demo-neuropsico.csv",sep=';')
309.        df_demo1.drop(df_demo1.columns.difference(["FS
   NEUROPSICO","Age","Gender"]),
310.                      1, inplace=True)
311.        df_demo1=df_demo1[:Ngroup1]
312.        df_demo1=df_demo1.sort_values(["FS NEUROPSICO"], ascending
   = True)
313.        df_demo1=df_demo1.drop(columns=["FS NEUROPSICO"])
314.
315.        df_demo2=pd.read_csv("TransversalDFT.csv",sep=';')
316.
   df_demo2.drop(df_demo2.columns.difference(["code","edadMRI","sexo"]
   ),
317.                      1, inplace=True)
318.        df_demo2["Age"]=df_demo2["edadMRI"]
319.        df_demo2["Gender"]=df_demo2["sexo"]
320.        df_demo2=df_demo2.drop(columns=["edadMRI","sexo"])
321.        df_demo2=df_demo2.sort_values(["code"], ascending = True)
322.        df_demo2=df_demo2.drop(columns=["code"])
323.
324.        df_demo=pd.concat([df_demo1,df_demo2],axis=0)
325.        df_demo=df_demo.reset_index()
326.        df_demo=df_demo.drop(['index'], axis=1) #eliminem index nou
327.        df["Age"]=df_demo["Age"]
328.        df["Gender"]=df_demo["Gender"]
329.
330.    if estudi=="ADDFT":
331.        df_demo1=pd.read_csv("demo-neuropsico.csv",sep=';')
332.        df_demo1.drop(df_demo1.columns.difference(["FS
   NEUROPSICO","Age"]),
333.                      1, inplace=True)
334.        df_demo1=df_demo1[44:44+Ngroup2]
335.        df_demo1=df_demo1.sort_values(["FS NEUROPSICO"], ascending
   = True)
336.        df_demo1=df_demo1.drop(columns=["FS NEUROPSICO"])
337.
338.        df_demo2=pd.read_csv("TransversalDFT.csv",sep=';')
339.
   df_demo2.drop(df_demo2.columns.difference(["code","edadMRI"]), 1,
340.                      inplace=True)
```

```
341.        df_demo2["Age"]=df_demo2["edadMRI"]
342.        df_demo2=df_demo2.drop(columns=["edadMRI"])
343.        df_demo2=df_demo2.sort_values(["code"], ascending = True)
344.        df_demo2=df_demo2.drop(columns=["code"])
345.
346.        df_demo=pd.concat([df_demo1,df_demo2],axis=0)
347.        df_demo=df_demo.reset_index()
348.        df_demo=df_demo.drop(['index'], axis=1)
349.        df["Age"]=df_demo["Age"]
350.
351.    # we add the LABELS
352.
353.    label=[0]*Ngroup1+[1]*Ngroup2
354.    df['Label']=label
355.    print(df.shape) # size of data
356.
357.    #------------------ ZSCORE NORMALIZATION -----------------
358.
359.    df_norm=pd.DataFrame()
360.    df_features=(df.columns).to_list()
361.    for col in df_features:
362.        if col!='Label':
363.            df_norm[col]=zscore(df[col])
364.        else:
365.            df_norm['Label']=df['Label']
366.
367.    df_norm.head()
368.
369.    #---------------------- ML PARAMETERS --------------------
370.    #---------------------------------------------------------
371.
372.    features=(df_norm.columns).to_list()
373.    del features[-1]
374.    X = np.asarray(df_norm[features]) #DATA
375.    y= np.asarray(df_norm['Label'])  #LABELS
376.
377.    #SVM
378.    models={"SVM":svm.SVC()}
379.
380.    #PARAMETER SELECTION
381.    # we define the range of the C and gamma parameters (remember
   that gamma
382.    # is only used with rbf.
383.    svm_params = [{'C': [0.1,1, 10, 100,1000],
384.                   'gamma': [1, 0.1, 0.01, 0.001],
385.                   'kernel': ['rbf']},
386.                  {'C': [0.1,1, 10, 100, 1000], 'kernel':
   ['linear']},
387.                  {'C': [0.1,1, 10, 100, 1000], 'kernel': ['poly']}]
388.
389.    params={"SVM": svm_params}
390.
391.    #----------------------CUMULATIVE VARIANCE ----------------
392.    #---------------------------------------------------------
393.    if estudi=="AD":
394.        tit='HC and AD'
395.        png='expl_variance_AD.png'
396.    elif estudi=="DFT":
397.        tit='HC and DFT'
398.        png='expl_variance_DFT.png'
399.    else:
```

```
        tit='AD and DFT'
        png='expl_variance_ADDFT.png'

    num_comp=len(features)-1
    pca = PCA(n_components=num_comp).fit(X) #we fit it with the train data,
                                            #but apply into the whole data set
    expl_var=np.cumsum(pca.explained_variance_ratio_)

    idxs=np.where(expl_var>=0.80)[0] #minimum n° of components needed, where the
                                     #cumulative expl.variance is 80% or greater
    nc=idxs[0]+1
    print("The minimum number of components needed is: ", nc)

    plt.figure(figsize=(10,6))
    plt.plot(np.arange(1,num_comp+1,1),expl_var,color='skyblue',linewidth=2,
             marker='X',markerfacecolor='steelblue', markersize=7)
    plt.title('Cumulative explained variance '+tit,fontsize=15)
    plt.xlabel('number of components',fontsize=15)
    plt.xticks(fontsize=14)
    plt.ylabel('cumulative explained variance',fontsize=15)
    plt.yticks(fontsize=14)
    plt.axvline(x=nc, color='k', linestyle='--',linewidth=1)
    plt.annotate("optimal nc: {}".format(nc),xy=(13,0.7),fontsize=14)
    plt.savefig(png)

    #---------------------------- ML ---------------------------
    #-----------------------------------------------------------

    #EVALUATION (PERFORMANCE)
    accuracies_PCA={} #"name_model": {01:acc1, 02:acc2...}
                      # where keys are the number of the iteration

    #CONFUSION MATRIX
    confusion_matrix={"TN":[],"FP":[],"FN":[],"TP":[]} #TP(true positive): well
    # predicted as patient; #FP: predicted as patient, truely control;
    #TN: well predicted as control; #FN: predicted as control, truely patient

    #PRECISION
    reportgroup1=[]
    reportgroup2=[]

    #PCA
    weights={} #{'num_component': {'name_feature': weight,...},..}

    #SVM
    best_params=[]

    skf = StratifiedKFold(n_splits=N_splits)
    for train_idx,test_idx in skf.split(X,y):
        X_train, X_test = X[train_idx],X[test_idx]
        y_train, y_test = y[train_idx],y[test_idx]
```

```
451.        for name,model in models.items():
452.            pca = PCA(n_components=nc)
453.            X_tr = pca.fit_transform(X_train) #we fit it with the
   train data, but apply it into the whole data set
454.            i=1
455.            for component in pca.components_: #pca.components_:
   weights given to each feature in the order they
456.                                              #appear in the
   dataset
457.                if i in weights.keys():
458.                    weights[i].append(list(component))
459.                else:
460.                    weights[i]=[]
461.                    weights[i].append(component)
462.                i+=1
463.
464.            X_tst= pca.transform(X_test)
465.
466.            for name,model in models.items():
467.                #grid search of the best parameter combination
468.                grid = GridSearchCV(model, params[name], refit =
   True,
469.                                    verbose = 0,cv=10)
470.                grid.fit(X_tr,y_train)
471.                best_params.append(grid.best_params_)
472.                clf=grid.best_estimator_ #classificator with the
   chosen parameters
473.
474.                #classification
475.                clf.fit(X_tr,y_train)
476.                y_pred=clf.predict(X_tst) #prediction
477.
478.
479.                TN,FP,FN,TP=(0,0,0,0)
480.                for idx in range(0,len(y_test)):
481.                    if y_test[idx]==y_pred[idx]:
482.                        if y_test[idx]==0:
483.                            TN+=1
484.                        else:
485.                            TP+=1
486.                    else: #si són diferents
487.                        if y_test[idx]==0:
488.                            FP+=1
489.                        else:
490.                            FN+=1
491.
492.                confusion_matrix["TN"].append(TN)
493.                confusion_matrix["TP"].append(TP)
494.                confusion_matrix["FP"].append(FP)
495.                confusion_matrix["FN"].append(FN)
496.
497.                #Precsion
498.                report=metrics.classification_report(y_test,
   y_pred,
499.
   output_dict=True)
500.
501.                reportg1=report['0']
502.                reportg1=reportg1['precision']
503.                reportgroup1.append(reportg1)
504.
```

```
                reportg2=report['1']
                reportg2=reportg2['precision']
                reportgroup2.append(reportg2)


                #Accuracy
                accuracy=metrics.accuracy_score(y_test, y_pred)
                if name in accuracies_PCA.keys():
                    accuracies_PCA[name].append(accuracy)
                else:
                    accuracies_PCA[name]=[accuracy]

    #mean confusion matrix
    confusion_matrix["TN"]=np.mean(confusion_matrix["TN"])
    confusion_matrix["TP"]=np.mean(confusion_matrix["TP"])
    confusion_matrix["FP"]=np.mean(confusion_matrix["FP"])
    confusion_matrix["FN"]=np.mean(confusion_matrix["FN"])

    #weights
    mean_std_weights={}
    for k in weights.keys():
        multiple_lists = weights[k]
        arrays = [np.array(x) for x in multiple_lists]
        mean_std_weights[k]=[[np.mean(k) for k in zip(*arrays)],
                             [np.std(k) for k in zip(*arrays)]]

    ############## MEAN ACCURACY
    from statistics import mean, stdev
    ACmean=accuracies_PCA['SVM']
    print( "MEAN ACCURACY:" , mean(ACmean))
    print( "SD :" , stdev(ACmean))

    ############## MEAN Precision
    print( "Group 1 precison:" , mean(reportgroup1))
    print( "Group 1 precison std:" , stdev(reportgroup1))
    print( "Group 2 precison:" , mean(reportgroup2))
    print( "Group 2 precison std:" , stdev(reportgroup2))

    ################### CONFUSION MATRIX ######################
    CM=[]
    for k,value in confusion_matrix.items():
        CM.append(value)
    CM=(np.array(CM)).reshape(2,2)
    print(CM)
    # Plot non-normalized confusion matrix
    plt.figure(figsize=(6.5,6.5))
    if estudi=="AD":
        classes=['HC','EOAD']
    elif estudi=="DFT":
        classes=['HC','FTD']
    else:
        classes=['EOAD','FTD']
    plt.figure(figsize=(6,6))
    plot_confusion_matrix(CM, classes=classes, normalize= False,
  title='')
    plt.savefig("conf_matrix "+estudi)


    ################## DATA PCA weights

    if m=="yes":
```

```
565.          mwf_pc1=dic_pc(1,mean_std_weights,features) #weighted
   features pc1
566.          #mwf_pc2=dic_pc(2,mean_std_weights) #idem pc2
567.          #mwf_pc3=dic_pc(3,mean_std_weights) #idem pc3
568.          #mwf_pc4=dic_pc(4,mean_std_weights) #idem pc4
569.
570.
571.    #We save the weights from the first component
572.    my_dict = mwf_pc1
573.
574.    aseg_dic={}
575.    dk_dic={}
576.    c1=0 #counter
577.    c2=0
578.    for key,val in my_dict.items():
579.        if key in features_aseg:
580.            idx=features_aseg.index(key)
581.            aseg_dic[c1]=[atlas_aseg[idx],val]
582.            c1+=1
583.        elif key in features_dk:
584.            idx=features_dk.index(key)
585.            dk_dic[c2]=[atlas_dk[idx],val]
586.            c2+=1
587.        else:
588.            None
589.
590.    if estudi=='AD':
591.        PC_aseg_csv=pd.DataFrame.from_dict(aseg_dic,orient='index',
592.
   columns=["region","weights"])
593.        PC_dk_csv=pd.DataFrame.from_dict(dk_dic,orient='index',
594.
   columns=["region","weights"])
595.        PC_aseg_csv.to_csv("PC1_HCvsEOAD_aseg.csv")
596.        PC_dk_csv.to_csv("PC1_HCvsEOAD_dk.csv")
597.        title="HC - EOAD"
598.    if estudi=='DFT':
599.        PC_aseg_csv=pd.DataFrame.from_dict(aseg_dic,orient='index',
600.
   columns=["region","weights"])
601.        PC_dk_csv=pd.DataFrame.from_dict(dk_dic,orient='index',
602.
   columns=["region","weights"])
603.        PC_aseg_csv.to_csv("PC1_HCvsDFT_aseg.csv")
604.        PC_dk_csv.to_csv("PC1_HCvsDFT_dk.csv")
605.        title="HC - FTD"
606.    if estudi=='ADDFT':
607.        PC_aseg_csv=pd.DataFrame.from_dict(aseg_dic,orient='index',
608.
   columns=["region","weights"])
609.        PC_dk_csv=pd.DataFrame.from_dict(dk_dic,orient='index',
610.
   columns=["region","weights"])
611.        PC_aseg_csv.to_csv("PC1_ADvsDFT_aseg.csv")
612.        PC_dk_csv.to_csv("PC1_ADvsDFT_dk.csv")
613.        title="EOAD - FTD"
614.
615.    old_names=["rh_middletemporal_thickness","rh_supramarginal_thic
   kness", "rh_superiortemporal_thickness", "rh_insula_thickness",
   "rh_inferiortemporal_thickness","rh_inferiorparietal_thickness",
```

```
    "rh_fusiform_thickness","rh_bankssts_thickness",
    "rh_precuneus_thickness","rh_lateralorbitofrontal_thickness",
    "rh_medialorbitofrontal_thickness","rh_superiorfrontal_thickness",
    "rh_temporalpole_thickness","rh_precentral_thickness",
    "rh_entorhinal_thickness","rh_posteriorcingulate_thickness",
    "rh_parahippocampal_thickness","rh_parsopercularis_thickness",
    "rh_isthmuscingulate_thickness","rh_caudalmiddlefrontal_thickness",
    "rh_postcentral_thickness","rh_superiorparietal_thickness",
    "rh_rostralmiddlefrontal_thickness","rh_parsorbitalis_thickness",
    "rh_parstriangularis_thickness","rh_lateraloccipital_thickness",
    "rh_paracentral_thickness","rh_rostralanteriorcingulate_thickness",
    "rh_transversetemporal_thickness","rh_lingual_thickness","rh_cuneus
    _thickness","rh_frontalpole_thickness","rh_pericalcarine_thickness"
    ,"rh_caudalanteriorcingulate_thickness","Norm-Hippocampus","Norm-
    Putamen","Norm-Lateral-Ventricle","Norm-Amygdala", "Norm-Thalamus-
    Proper","Norm-Caudate","Norm-Cerebellum-Cortex","Norm-Pallidum"]
616.
617.    new_names=["middle temporal","supramarginal","superior
    temporal","insula","inferior temporal","inferior
    parietal","fusiform","bankssts","precuneus","lateral
    orbitofrontal","medial orbitofrontal",  "superior
    frontal","temporal pole","precentral","entorhinal","posterior
    cingulate","parahippocampal","pars opercularis","isthmus
    cingulate","caudal middle frontal","postcentral","superior
    parietal","rostral middle frontal","pars orbitalis","pars
    triangularis","lateral occipital","paracentral","rostral anterior
    cingulate","transverse temporal","lingual","cuneus","frontal
    pole","pericalcarine","caudal anterior cingulate",
    "hippocampus","putamen","lateral ventricle","amygdala","thalamus
    proper","caudate","cerebellum cortex","pallidum"]
618.
619.    change_names={}
620.    for old,new in zip(old_names,new_names):
621.        change_names[old]=new
622.    change_names["rh_MeanThickness_thickness"]="Mean thickness"
623.    change_names["Norm-Accumbens-area"]="Accumbens area"
624.    change_names["Norm-Inf-Lat-Ventr"]="Inferior Lateral Ventricle"
625.    change_names["Norm-choroid-plexus"]="Choroid Plexus"
626.
627.    pc1_df=pd.DataFrame (weights[1], columns = features)
628.    pc1_df=pc1_df.reindex(abs(pc1_df.mean()).sort_values(ascending=
    False).index, axis=1)
629.    pc1_df=pc1_df.rename(columns = change_names, inplace = False)
630.    plt.figure(figsize=(20,10))
631.    plt.title(title,fontsize=22)
632.    ax = sns.boxplot(data=pc1_df)
633.    plt.ylabel("PC1 weights",fontsize=20)
634.    plt.xticks(rotation=90,fontsize=18)
635.    plt.yticks(fontsize=18)
636.    plt.show()
637.
638.    #Plots weights PCA
639.    plt.figure(figsize=(20,10))
640.    plot_bar(0,len(mwf_pc1.keys()),mwf_pc1)
641.    plt.title(title,fontsize=17)
642.    plt.savefig("Barplot weights "+ title)
643.    plt.show()
```

### 12.2. R studio code: brain atlases.

```
library(ggplot2)
library(ggseg)

#Path
setwd("C:\Users\Laia.LAPTOP-1BS820LB\Desktop\4t\TFG\Treball amb agnes\brain tables+plot")

asegplot<-function(data,title){
  data$X <- NULL
  data<-subset(data,data$weights<0)
      data$weights<-abs(data$weights)
      return(ggseg(.data=data, atlas="aseg", mapping=aes(fill=weights))+
scale_fill_gradientn(colours=c("greenyellow","dodgerblue4","light blue","violet"),limits=c(0.0,0.2))+
            labs(title=title,cex=3))
    }

    aparcplot<-function(data,title){
      data$X <- NULL
      data$weights<-abs(data$weights)
      return(ggseg(.data=data, colour="white", mapping=aes(fill=weights))+
scale_fill_gradientn(colours=c("greenyellow","dodgerblue4","light blue","violet"),limits=c(0.0,0.2))+
             labs(title=title,cex=3))
    }

    ## DATA i PLOTS

    #HC vs EOAD

    data_dk=read.csv("PC1_HCvsEOAD_dk.csv",header = TRUE)
    data_aseg=read.csv("PC1_HCvsEOAD_aseg.csv",header = TRUE)

    png("dk_CTRvsAD.png", width = 558, height = 389)
    aparcplot(data_dk,title="DK weights HC-EOAD")
    dev.off()

    png("aseg_CTRvsAD.png", width = 558, height = 389)
    asegplot(data_aseg,title='Aseg weights HC-EOAD')
    dev.off()

    # CTR vs EOAD vs DFT

    data_dk=read.csv("PC1_all_dk.csv",header = TRUE)
    data_aseg=read.csv("PC1_all_aseg.csv",header = TRUE)

    png("dk_all.png", width = 558, height = 389)
    aparcplot(data_dk,title="DK weights HC-EOAD-FTD")
    dev.off()

    png("aseg_all.png", width = 558, height = 389)
    asegplot(data_aseg,title="Aseg weights HC-EOAD-FTD")
```

```
50.     dev.off()
51.
52.     # CTR vs DFT
53.     data_dk=read.csv("PC1_HCvsDFT_dk.csv",header = TRUE)
54.     data_aseg=read.csv("PC1_HCvsDFT_aseg.csv",header = TRUE)
55.
56.     png("dk_CTRvsDFT.png", width = 558, height = 389)
57.     aparcplot(data_dk,title="DK weights HC-FTD")
58.     dev.off()
59.
60.     png("aseg_CTRvsDFT.png", width = 558, height = 389)
61.     asegplot(data_aseg,title="Aseg weights HC-FTD")
62.     dev.off()
63.
64.     # AD vs DFT
65.     data_dk=read.csv("PC1_ADvsDFT_dk.csv",header = TRUE)
66.     data_aseg=read.csv("PC1_ADvsDFT_aseg.csv",header = TRUE)
67.
68.     png("dk_ADvsDFT.png", width = 558, height = 389)
69.     aparcplot(data_dk,title="DK weights EOAD-FTD")
70.     dev.off()
71.
72.     png("aseg_ADvsDFT.png", width = 558, height = 389)
73.     asegplot(data_aseg,title="Aseg weights EOAD-FTD")
74.     dev.off()
```