Universidade do Minho
Escola de Engenharia

Paulo Jorge Alves

Mining MRI Predictors for Mild Cognitive Impairment

October 22

Mining MRI Predictors for Mild Cognitive Impairment
Paulo Jorge Alves
UMINHO | 2022

Universidade do Minho
Escola de Engenharia

Paulo Jorge Alves

Mining MRI Predictors for Mild Cognitive Impairment

Master Dissertation
Master's Degree in Biomedical Engineering
Medical Informatics

Project supervised by
Victor Manuel Rodrigues Alves
Tiago Gil Oliveira

October 2022

DECLARATION

Name: Paulo Jorge Alves

Dissertation Title: Mining MRI Predictors for Mild Cognitive Impairment

Mentors: Victor Manuel Rodrigues Alves and Tiago Gil Oliveira

Conclusion Year: 2022

Master Designation: Master's Degree in Biomedical Engineering

Master Branch: Medical Informatics

Universidade do Minho, 28 / 10 / 2022

Signature: Paulo Jorge Alves

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Universidade do Minho, 28 / 10 / 2022

Signature: Paulo Jorge Alves

Acknowledgements

A master's degree is a long journey, which includes a trajectory composed of numerous challenges, uncertainties, sorrows, joys and many mishaps along the way. Walking this path was only possible with the support and strength of several people, to whom I especially dedicate this life project.

I want first to thank my supervisor Victor Alves, to whom I am truly grateful for his guidance. He was essential during all process of the dissertation findings, discussions and writing. He was always available, motivated and ready to guide me through this experience. Thank you so very much for all the hours spent in *Discord* and for all the help, even in the most difficult moments.

I also want to thank doctor Tiago Gil and his team at the ICVS laboratory for all the knowledge, kindness and friendship. Meeting the whole team has contributed to my growth as a professional and as a person. Thank you for integrating me into your team!

I also want to thank my family. To my parents, sister, grandparents and godparents, thank you for all the support along this journey.

I cannot forget to thank my friends. And for them, "thank you" cannot even describe how I feel. I don't have enough words to express how grateful I am to have them with me.

To my best friends, Marco Castro, Rita Leite and Leonor Padrão, I thank the unconditional support, the hours of study together and all the motivation. That was all the easier. Thank you for our friendship.

Besides, my university years would never be the same without my colleagues and friends. Thank you for sharing with me memorable moments, walks, laughs and happy tears.

"O mundo está nas mãos daqueles que têm a coragem de sonhar e correr o risco de viver seus sonhos."

Paulo Coelho

ABSTRACT

Currently, without a cure, Alzheimer's disease (AD) is the most common form of dementia, accounting for up to 70% of dementia cases worldwide. Given the increasingly significant social and economic impact of AD, its earlier detection is then a point of great interest, since it contributes to the deceleration of its progression. Currently, several machine learning techniques are already implemented in the detection and classification of both patients with AD and patients with signs of Mild Cognitive Impairment (MCI) who may or may not develop AD, with satisfactory results.

The combination of medical imaging and biomedical informatics has given rise to a discipline called medical imaging informatics, used to promote the improvement and knowledge of clinical care. If on the one hand, medical imaging is an in vivo alternative to study the state of a disease, on the other hand, biomedical informatics is related to the development of computer techniques to create and manage medical data.

Technological advancement has caused further study in Deep Learning from the beginning of the last decade, proving that these artificial intelligence techniques are proving to be better than what already exists in the medical field.

Systems based on representational deep learning are often considered "black boxes" due to their lack of explainability. Therefore, the interpretability of Machine Learning systems is a crucial research area, given the high performances currently achieved with these systems.

The purpose of this project is, to use Magnetic Resonance Imaging (MRI) taken from the Alzheimer's Disease Neuroimaging Initiative (ADNI) platform, not only to discover radiomic predictors to find out how an individual with MCI will evolve over the years but also intended to find out which areas of the brain are most affected by this development.

Keywords: Alzheimer's disease, Mild Cognitive Impairment, Radiomics, Deep learning, MRI.

RESUMO

Presentemente sem cura, a doença de Alzheimer (AD) é a forma mais comum de demência, sendo responsável por quase 70% dos casos de demência registados mundialmente. Atendendo ao impacto social e económico cada vez mais significativo da AD, a sua deteção atempada é, então, um ponto de grande interesse, uma vez que contribui para a desaceleração da progressão da mesma. Atualmente, várias técnicas de aprendizagem automática estão já implementadas na deteção e classificação quer de pacientes com AD, quer de doentes com sinais de défice cognitivo ligeiro (MCI) que poderão, ou não, desenvolver AD, com resultados satisfatórios.

A combinação de imagiologia médica e informática biomédica, originou uma disciplina intitulada de informática de imagem médica, usada para promover a melhoria e o conhecimento do cuidado clínico. Se por um lado, a imagiologia médica é uma alternativa *in vivo* de estudar o estado de uma doença, por outro lado, a informática biomédica está relacionada com o desenvolvimento de técnicas de informática para criar e gerenciar dados médicos.

O avanço tecnológico provocou um estudo mais aprofundado no *Deep Learning* a partir do início da última década, provando que estas técnicas de inteligência artificial estão a revelar-se melhores do que o que já existe na área médica.

Os sistemas baseados na aprendizagem profunda e representativa são frequentemente considerados como "caixas negras" devido à sua falta de explicabilidade. Assim sendo, a interpretabilidade de sistemas de *Machine Learning* é uma área de investigação crucial, dado os elevados desempenhos atualmente atingidos com estes sistemas.

O objetivo deste projeto é, através de imagens de ressonância magnética retiradas da plataforma *Alzheimer's Disease Neuroimaging Initiative*, não só descobrir preditores radiómicos para descobrir como um indivíduo com MCI irá evoluir ao longo dos anos, mas também se destina a descobrir quais as áreas do cérebro mais afetadas por este desenvolvimento.

Palavras-Chave: Doença de Alzheimer, Défice cognitivo ligeiro, *Radiomics*, Aprendizagem Profunda, Ressonância Magnética.

Table of Contents

LIST OF FIGURES

List of Tables

List of Abbreviations and acronyms

A

AI	Artificial Intelligence
ANN	Artificial Neural Network
AD	Alzheimer's Disease
AUC	Area Under the Curve

B

BET	Brain Extraction Tool

C

CT	Computer Tomography
CNN	Convolution Neural Networks
CDR	Clinical Dementia Rating
CPU	Central Process Unit
CSV	Comma-Separated Values

D

DL	Deep Learning
DNN	Deep Neural Network
DICOM	Digital Imaging and Communications in Medicine

G

GDS	Global Deterioration Scale
GPU	Graphics Processing Unit
GLCM	Grey Level Co-Occurrence Matrix
GLDM	Grey Level Dependence Matrix
GLM	General Linear Model
GLRLM	Grey Level Run Length Matrix
GLSZM	Grey Level Size Zone Matrix

I

IDE	Integrated Development Environment

M

ML	Machine Learning
MLP	Multilayer Perceptron
MRI	Magnetic Resonance Imaging
MCI	Mild Cognitive Impairment
MMSE	Mini Mental State Examination

N

NIfTI	Neuroimaging Informatics Technology Initiative
NGTDM	Neighborhood Grey Tone Difference Matrix
NN	Neural Networks

R

RNN	Recurrent Neural Network
ROI	Region of Interest
ReLU	Rectifier Linear Unit

S

SVM	Support-Vector Machine

X

XGBoost	Extreme Gradient Boosting

GLOSSARY

Activation Function	Used in neural networks, decides whether the neuron is activated or not. The neuron receives an input value from previous neurons. If the value is large enough, the neural unit passes the value to the next neuron, thus acting as a threshold function.
Artificial Intelligence (AI)	The capacity of a computer to do some human activities like thinking, making decisions or learning.
Artificial Neural Networks (ANN)	A model that is inspired by the brain and that performs the mapping from a received input to the desired output using the "neurons" of the network, also referred to as a unit and nodes that are weighted interconnected.
Batch	In Neural Networks, samples are used in one iteration.
Class	Refers to the output category of the data, and a label in a dataset refers to one of the classes.
Cross-Validation	Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model.
Dataset	A collection of data. The data contains features and, if using supervised training, a label.
Deep Learning (DL)	A subfield of Machine Learning belongs to representation learning, where the algorithm is trained to learn the feature hierarchy to classify the data provided.
Dropout	A form of regularization useful in training ANN. It works by removing a random selection of a fixed number of the units in a network layer for a single gradient step. The more units dropped out, the stronger the regularization.
Epoch	One forward pass and one backward pass through all training samples.
Feature	In Machine Learning, a feature is the measurable propriety of an attribute.
FSL	A library of analysis tools for FMRI, MRI and DTI brain imaging data. This library contains the tools used in the spatial normalization process.
Hyperparameter	In machine learning, a hyperparameter is a parameter whose value is used to control the learning process.
Loss Function	Also called Cost Function, it is the function that computes the distance between the current output of the algorithm and the expected output. It is a method to evaluate how your algorithm models the data.

Machine Learning (ML)	An approach to achieving Artificial Intelligence, where a computer is trained using specific algorithms and a large amount of data to gain the ability to adapt to new situations and predict patterns.
Magnetic Resonance Imaging (MRI)	A non-invasive imaging technique based on the absorption and emission of energy in the radiofrequency (RF) spectrum of the electromagnetic spectrum. MRI is widely often used for clinical assistance and also in clinical research.
Medical Imaging Informatics	A discipline that results from the combination of medical images and biomedical informatics. Serves to enhance the improvement and knowledge of clinical care. On the one hand, medical imaging allows the study of a disease state *in vivo*. On the other hand, biomedical informatics is involved in the development of computer science techniques for creating and managing medical data.
Multilayer Perceptron (MLP)	The multilayer perceptron is a neural network similar to the perceptron, but with more than one layer of neurons in direct feed-forward. Such a network is composed of layers of neurons connected together by synapses with weights.
NiBabel	Python package that provides easy read and write access to some of the common neuroimaging file formats. Among the accepted formats are the NIfTI format, the format of the MRI images of the dataset used.
Nifti	A simple, minimalistic format which has been widely adopted in neuroimaging research, allowing scientists to mix and match image processing and analysis tools developed by different teams.
Optimizer	It is a mathematical algorithm used in Machine Learning to find the best available alternative under the given constraints which aims to minimize the loss/penalty value by reaching the global minimum.
Radiomics	Is a field of medical study that aims to extract a large number of quantitative features from medical images using data characterization algorithms. The data is assessed for improved decision support. It has the potential to uncover disease characteristics that are difficult to identify by human vision alone.
Region of Interest (ROI)	Contours or surfaces outlining an object in a region identified for a particular purpose.
Train/Test set	The subset of the dataset used to train/test the model.
Validation set	A subset of the dataset — disjunct from the training set—that is used to adjust hyperparameters.
XGBoost	Stands for “Extreme Gradient Boosting”, it is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework.

1 INTRODUCTION

The relationship between informatics and medicine is truly astonishing. Nowadays, there isn't anything more important than the connection between the patient and their physician. However, the existence of medical informatics and its fields deliver targeted and accurate data.

Medical informatics, a young science, which appeared in the decades after the invention of the digital computer in the 1940s, has been emerging as a discipline over recent times, along with the successive and developing formal definitions that have been put forward [1], [2].

It can be defined as the intersection of information science, computer science and healthcare. This field deals with the resources, devices, and methods needed to optimize the acquisition, storage, retrieval, and use of information in health and biomedicine [3].

According to J C Wyatt and J L Y Liu, medical informatics is the study and application of methods to improve the management of patient data, clinical knowledge, population data and other information relevant to patient and community healthcare [2].

In a broad sense, medical informatics in a medical context is the use of informatics and information technologies in health care [4].

Healthcare is continually changing as the underlying science and practice of healthcare is in continuous transformation. Medical Informatics as a discipline is strongly affected by these changes and is in a position to be a key and active contributor to these changes.

Looking at the development of medical informatics, it can be recognised that it has been growing steadily and that today, as a cross-cutting discipline, it constitutes one of the bases of medicine and healthcare. As a result, much is expected of medical informatics to help achieve the health of people around the world, both in contributing to the quality and efficiency of healthcare and biomedical innovation, as well as informatics, health and information science research [5].

In turn, Medical Imaging Informatics is a subset of the supra-mentioned area that studies, among other aspects, image information acquisition, visualization, processing, storage, transmission, security, and knowledge discovery from large biomedical data sets.

Medical imaging has become a fundamental tool in modern healthcare, providing the only in vivo means of studying a given disease. Biomedical informatics is transforming the way one handles and thinks about large amounts of electronic clinical data [6].

Medical Imaging Informatics is a discipline that intersects Medical Imaging and Imaging Informatics to extend the comprehension of disease processes and improve the efficiency and reliability of services regarding medical image usage and exchange in healthcare systems.

Biomedical imaging has revolutionized the practice of medicine with the ability to diagnose diseases by imaging the human body and visualizing pathological cells and specimens in high resolution. In general terms, images are formed through the interaction of electromagnetic waves at various wavelengths with biological tissues. Images formed with high-energy radiation at shorter wavelengths, such as X-rays and gamma rays at one end of the spectrum, are ionizing while at longer wavelengths, such as Magnetic Resonance Imaging (MRI) and Ultrasound, are non-ionising [6].

MRI produces high spatial resolution volumetric images of mainly hydrogen nuclei, using an externally applied magnetic field in conjunction with radiofrequency pulses. It is commonly used in numerous applications including musculoskeletal, cardiovascular, and neurological imaging with excellent soft-tissue contrast. Additionally, functional MRI has evolved into a large subfield of study with applications in areas such as functional connectivity mapping in the brain.

Medical image analysis typically involves the delineation of objects of interest (segmentation) or the description of labels (classification) [2].

To the present, medical image analysis has been constrained by a lack of theoretical understanding of how to optimize, choose and process visual features. However, the emergence of machine learning approaches has provided good results in a wide range of applications. These approaches, attempt to learn the features of interest and optimize parameters based on training examples.

Machine learning involves defining a learning problem to solve a task based on inputs. Recent advances in Graphics Processing Unit (GPU) processing and easy availability of data for training have led to a rapid expansion in neural networks and deep learning for regression and classification. Thus, deep learning methods simultaneously optimize for decisions while identifying and learning the appropriate input features.

Deep learning approaches enable the automation of many aspects of the information extraction and classification process. A variety of methods have been developed to classify tissue regions or whole slide images depending on the context and location of the disease [7].

Medical imaging informatics addresses not only the images themselves but encompasses the associated data to understand the context of the imaging study, to document observations, and to correlate and reach new conclusions about a disease and the course of a medical issue [2].

1.1 CONTEXT AND MOTIVATION

Mild forgetfulness is often a normal part of ageing. But for some people, memory and thinking issues can become more serious as they get older. Cognitive impairment and dementia are increasingly frequent worldwide, with an impact on the quality of life of millions of patients and their families.

The World Health Organization estimates that there are 47.5 million people with dementia worldwide, a number that could reach 75.6 million in 2030 and nearly triple by 2050. In the OECD's "Health at a Glance 2017" report published on 10 November 2017, new data on the prevalence of dementia are presented, placing Portugal as the 4th country with the most cases per thousand inhabitants. According to this report, the estimated number of cases with dementia in Portugal rises to more than 205,000 persons, a number that will rise to 322,000 cases by 2037 [8].

The problem or condition which will be discussed is Mild Cognitive Impairment (MCI) — a condition that can be developed over time, namely, as the years pass by, the inevitability of ageing. Besides that, this condition can also be a cause of a much more problematic disease, which is Alzheimer's disease (AD). But not always do people get worse, they can get better if the diagnosis by the physician is done properly.

MCI is a syndrome defined as cognitive decline greater than that expected for an individual's age and education level, but that does not interfere notably with activities of daily life. It is, thus, distinct from dementia, in which cognitive deficits are more severe and widespread and have a substantial effect on daily function. However, mild cognitive impairment with memory complaints and deficits (amnestic mild cognitive impairment) is consistently shown to have a high risk of progression to dementia, particularly of the Alzheimer type.

Thus, in addition to studying this disease, it is also important to predict its onset in time for possible treatment to reduce the number of people with dementia in the long term.

1.2 OBJECTIVES

This study proposes a pipeline that facilitates the utility of mining predictors for MCI condition based on radiomic features extracted from T1-weighted MRI scans.

The objective of this dissertation is to discover the radiomic predictors, to find out how an individual with MCI will evolve over the years, and to discover which areas of the brain are most affected by this development. The following tasks will be done:

- Segmentation of the several scans, using FreeSurfer software, in the most important areas of the brain for this study;
- Construction of the datasets to be used;
- Development of a Deep Learning model for the analysis of medical images;
- Obtaining the intended features.

The hypothesis is that radiomic features in the postulated region of interest will demonstrate differences that will consequently support delineating patients with MCI that will evolve to AD and those who won't.

1.3 STRUCTURE OF THE DOCUMENT

This work comprises five chapters. This chapter is the first and intends to help the reader understand the context and underlying motivations of the study, along with its proposed objectives. The next chapter is a brief literature review of the topic under study and a revision of the concepts and technologies that lay the foundations for this work.

The third chapter contains all the information about the data used and how it was selected and constructed. Then, chapter four contains all the methods that were used and all the work that was carried out, with an explanation of all the results. Finally, there is a discussion of the results and some of the conclusions are drawn, also exploring future work.

In the end, there is an appendix with all the code used throughout the project.

2 STATE OF THE ART

This chapter reviews the literature and focuses on the theoretical concepts and technologies related to the development of this project. Thus, it is important to focus on various fields such as MCI, MRI, Radiomics and Machine Learning.

2.1 LITERATURE REVIEW

Alzheimer's disease is a neurodegenerative disease characterised by a progressive decline in chronic primary memory and cognitive impairment. The increasing incidence, high disability rate and high cost of treatment have made AD one of the most serious diseases threatening human beings. MCI is a transitional state between normal ageing and AD, which has a higher risk of conversion to AD [9].

To date, the diagnosis of AD or MCI is established after irreversible changes in brain structure. Therefore, the development of new biomarkers is crucial for the early detection and treatment of this disease. Currently, some research studies are showing that radiomic analysis can be a good method for the diagnosis and classification of AD and MCI [9].

Recently, the task of predicting conversion from MCI to AD has received much attention, driven mainly by the emergence of large multicentre studies collecting data from hundreds of patients and controls [10].

Using baseline data, Chupin et al. (2009) [11] automatically segmented the hippocampus and used volume in a k-means classifier to predict conversion from MCI to AD within 18 months. They achieved a classification accuracy of over 64%. Global hippocampal volume may be prone to large inter-individual differences and may not be specific for dementia prediction.

Westman et al. (2011) [12] used predefined cortical thickness regions and subcortical volumes to predict conversion. They combined ADNI data with data from a large European project, AddNeuroMed (Lovestone et al., 2009 [13]), to predict converters in their respective cohorts. For ADNI data, they obtained a low accuracy (58%), while accuracy for AddNeuroMed was higher (70%). It should be noted that the AddNeuroMed data were collected from six different sites, while the ADNI data were collected from over 50 different sites. According to the authors, this may explain the superior accuracy of the predictions on the AddNeuroMed data.

Cuingnet et al. (2011) [14] evaluated the predictive power of ten different structural methods, including hippocampal volume and shape, various VBM (Voxel-Based Morphometry) approaches, and cortical thickness. Using SVM (Support Vector Machine), they obtained accuracies in the range of 58% - 71%, but with relatively low sensitivities. Validation was performed by dividing subjects into training and test sets of equal size, which biased the results towards the random selection process. In addition, the comparison suffered from many image processing flaws that led to different training and test sets for each method evaluated. Thus, the results may deviate from what can be obtained using the original methods.

Wolz et al. (2011) [15] selected baseline scans from the entire MCI population of the ADNI cohort and multiple methods were applied to predict conversion to AD. They used statistical maps to define ROIs for respectively TBM and cortical thickness. Since they included test subjects in their ROI selection step, their results may be biased and accuracy may be overestimated. In addition, they evaluated hippocampal volume and multiple-based learning as predictors. Accuracies in the range of 56%-65% were obtained using the respective methods while combining the methods increased the accuracy to 68%. This study is so far the most comprehensive work evaluating structural methods for predicting conversion to AD in subjects with MCI.

Shen et al. (2018) [16] proposed a decision support model based on deep learning methods to predict the probability of conversion from MCI to AD within one year. Based on 165 samples with MRI scans from the ADNI database, in which all patients with MCI were converted to AD at different conversion periods, they first extracted image features based on the convolutional neural network (CNN) method and then used the support vector machine (SVM) classifier to classify these features. The results showed that the classification accuracy using linear, polynomial and RBF kernel could reach 91.0%, 90.0% and 92.3%. As a result, this study indicated that the decision support model is the potential to be applied in predicting the probability of conversion from MCI to AD within one year.

Das et al. (2019) [17] presented a new interpretable model based on distinct weighting rules. The model was evaluated for 151 subjects from the ADNI cohort (97 AD and 54 CN). The framework was trained in two phases, with the first phase using plasma features to train the interpretable model. Subjects with unclear predictions were propagated to the second phase, where an SVM (Cortes and Vapnik (1995) [18] was trained using invasive Cerebrospinal Fluid (CSF) markers. The evaluation included both Cross-Validation and an independent test dataset. For the test dataset, an area under the Receiver Operating Characteristics Curve (AUC) of 0.81 was achieved.

Choe et al. (2020) [19] examined the utility of the MMSE to predict the progression from MCI to AD. They used a total of 306 individuals with MCI from the ADNI database. They performed standardised clinical and neuropsychological testing at baseline and at 2-year follow-up and logistic regression analysis were performed to examine total MMSE. With this, they underline the importance of assessing orientation and constructing domains to identify subjects at high risk of converting Alzheimer's disease among older people whose memory function is already compromised. In terms of simplicity and ease of interpretation, they conclude that MMSE subscales of memory, orientation, and construct could be useful screening tools to predict conversion to AD from MCI in practical clinical settings.

A few more studies were selected and a summary of them are presented in Table 2.1.

Before ending this chapter, it is also important to highlight the work of Lin et al. (2018) [20] who designed a framework that uses MRI data to predict conversion from MCI to AD by applying CNN and other machine learning algorithms. The results showed that CNN can extract discriminative features of the hippocampus for prediction by learning the morphological changes of the hippocampus between AD and NC, and FreeSurfer provides extra structural brain imaging features to improve the prediction performance as complementary information. Compared with other state-of-the-art methods, the proposed one outperforms others with higher accuracy and AUC, while maintaining a good balance between sensitivity and specificity.

Also, Chaddad et al. (2018) [21] investigated the radiomic features derived from individual subcortical brain regions to identify AD. The results indicated that the hippocampus and amygdala were the brain regions that had the greatest differences between AD and HC and that the "correlation" and "volume" features are the most important for AD diagnosis. In addition, they also suggested that entropy features derived from CNN layers could be used as an effective biomarker for AD.

Although radiomic analysis is already applied to the diagnosis and classification of AD and MCI, there is still a long way to go from these computer-aided diagnostic methods to clinical application. It is believed that AI will be widely used in the future radiomic analysis of AD and will become a powerful auxiliary tool for the diagnosis and clinical treatment of AD.

Table 2.1 - Summary of selected studies

Study authors	Number of subjects	Acquisition	Pre-processing?	Region of interest (ROI) and segmentation	Feature extraction software	Classification or prediction method	Conclusions
Hett et al., 2017 [22]	800 (ADNI)	T1-weighted MRI images	✓	Hippocampus	A bank of 3D Gabor filters	Histogram-based weak classifiers aggregation (accuracy of 91.3% for NC/AD)	Texture-based grading framework and histogram-based weak classifiers aggregation can better discriminate early stages of AD
Luk et al., 2018 [23]	790 (ADNI)	T1-weighted MRI images	✗	Whole brain	Toolbox on West Grid Cluster distributed computing	A binary logistic regression model using texture features, hippocampal occupancy and clinical data as variables (AUC of 0.905, and it predicted MCI conversion with acc. of 76.2%)	Whole-brain S texture analysis has the potential to predict progression from MCI to AD
Sørensen et al., 2016 [24]	503 (ADNI) + 141 (AIBL)	T1-weighted MRI images	✓	Hippocampus (FreeSurfer software)	Combining a texture descriptor and an SVM	SVM - ADNI data: AUC achieved 0.912 in discriminating AD from NC, 0.764 in discriminating MCI from NC, and 0.74 in predicting MCI-to-AD conversion	Hippocampal texture abnormalities may be neuroimaging biomarkers for the diagnosis and prognosis of AD, MCI and NC
Martinez-Torteya et al., 2014 [25]	62 (ADNI)	MP-RAGE images	✗	83 ROIs (automatic whole-brain multi-region segmentation method)	Not specified	Logistic regression model (an average blind-test accuracy and AUC of 0.79)	MRI features about both signal and texture add predictive power for MCI to AD progression
Wu et al., 2018 [26]	457 (ADNI)	T1-weighted MRI images	✓	No ROI	CNN	GoogleNet acquired discrimination accuracies of 97.58% (NC), 67.33% (sMCI) and 84.71% (MCIc); CaffeNet got classification accuracies of 98.71% (NC), 72.04% (sMCI) and 92.35% (MCIc)	The proposed methods had strong ability in classification among sMCI, MCIc and NC, and prediction of MCI conversion

2.2 MILD COGNITIVE IMPAIRMENT

Advancements in healthcare over the last 50 years have extended average life expectancy, resulting in an increase in the number of people over 65 years of age. Many aged people complain of memory loss and have more difficulty doing certain tasks than young people, particularly those that assess memory. These results suggest that memory impairments are a common consequence of the ageing process [27].

"Our brain, like the rest of our body, changes as we grow older". Mild cognitive impairment is a condition in which memory problems and/or other cognitive difficulties (e.g., language, visual-spatial abilities) are noticeable to the affected person and/or others (e.g., family, friends, co-workers), but are not severe enough to interfere with basic life skills [28], [29].

MCI is considered an intermediate stage between the expected cognitive decline of normal ageing and the more severe decline into dementia. It can involve issues with memory, language, thinking and judgement that are greater than the normal age-related changes [28].

Cognitive functioning is typically characterized into one of 5 domains: 1) learning and memory, 2) language, 3) visuospatial, 4) executive, and 5) psychomotor. These domains have a tentative match to their brain location. For a diagnosis of MCI, only one of these areas must be impaired to make a diagnosis, while more than one domain must be impaired to make a diagnosis of dementia [30].

MCI is not considered dementia, but about 10 to 15 per cent of people with MCI can develop dementia each year, including a specific type of dementia known as Alzheimer's disease (AD). AD accounts for between 60 and 80 per cent of dementia cases [31].

The differences between MCI, Alzheimer's disease, and other types of dementia are related to the severity of their symptoms and are represented in Figure 2.1 [32].

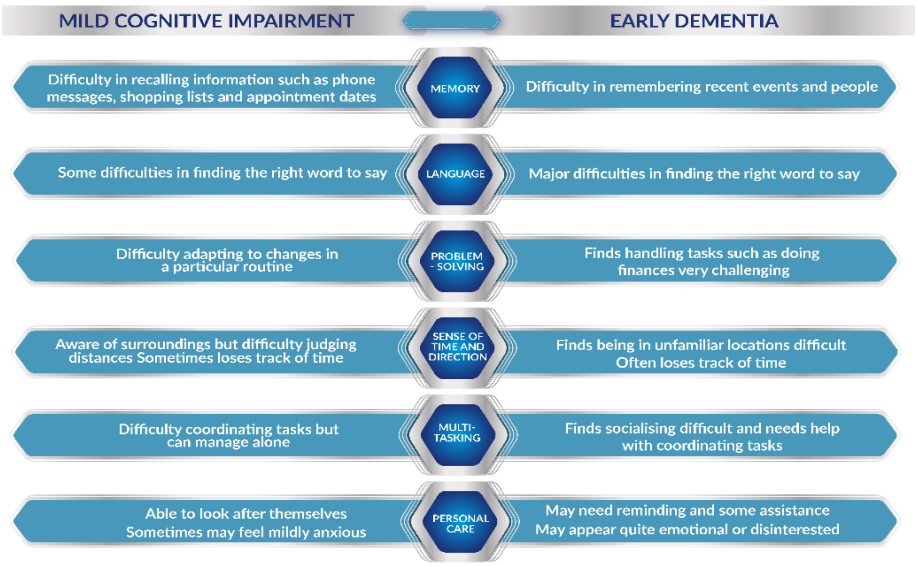


Figure 2.1 - Differences between MCI and dementia. Retrieved from [32]

The typical transition from normal ageing to AD is very subtle, with signs and symptoms emerging very gradually. Everyone can expect to experience cognitive changes as they age. Common changes associated with normal ageing include slower retrieval of information such as names, greater effort to learn and store new information, greater susceptibility to distraction, slower processing of new information, and greater difficulty in multi-tasking [28].

An individual with MCI will score significantly lower than others of the same age on neuropsychological measures of the affected cognitive domains (e.g., memory, language) [28].

Over the last decades, several terms and definitions have been proposed to determine the intermediate stage between normal ageing and dementia. In 1962, V.A. Kral first described two types of age-related cognitive changes in his works. One is "benign senescent forgetfulness" (BSF), which is characterised by a mild, non-progressive decline in memory and presumably involves non-specific histopathological changes in the brain. The second form, "malignant senescent forgetting" (MSF) includes progressive cognitive and behavioural changes that involve specific brain histopathology. The introduction of the term "benign senescent forgetfulness" was the most appropriate to conceive of MCI [33].

This was followed by a National Institute of Mental Health working group in 1986 that proposed the term "Age-Associated Memory Impairment" (AAMI) to refer to memory changes that were felt to be a variant of normal ageing. The shortcomings of AAMI included restricting impairment to the memory domain only and comparing memory function in older adults with the performance of young adults. As such, the AAMI was unable to delineate individuals at risk of developing pathogenic logical conditions from those who underwent the processes of normal ageing. The International Psychogeriatric Association coined the term "age-related cognitive decline" to overcome many of the deficiencies recognised in AAMI. Operational criteria for age-associated cognitive decline referred to a variety of cognitive domains presumed to be in decline in normal ageing and included values adjusted for age and education [34].

In 1995, in an observational study of ageing at the Mayo Clinic, R.C. Petersen and colleagues adopted mild cognitive impairment as an independent diagnostic entity to categorize persons with memory complaints who were not demented, maintained global cognitive function and daily living skills, but scored below the age-adjusted norms on memory tests [33].

In a 1999 article published in the Archives of Neurology, a group of researchers from the Mayo Clinic described their experience with participants with MCI and presented the following diagnostic criteria [35]:

- Memory complaints, preferably corroborated by an informant
- Memory impairment documented according to appropriate reference values
- Essentially normal performance in nonmemory cognitive domains

- Generally preserved activities of daily living
- Not demented

In 2003, Win blad et al. convened a conference of international experts on MCI to review the criteria. From this conference, new, more expansive criteria for MCI were proposed, and these criteria now form the basis for the public-private neuroimaging/biomarker consortium, the Alzheimer Disease Neuroimaging Initiative (ADNI).

These criteria describe the clinical phenotypes of amnestic MCI and non-amnestic MCI with the subtypes of single and multiple domain classifications [36], [37]:

- Amnestic MCI (aMCI) vs. non-amnestic MCI (naMCI): In aMCI, memory is significantly impaired. Other cognitive functions are saved. In naMCI, memory remains intact, but one (single domain) or more (multiple domains) other cognitive skills (e.g. language, visual-spatial skills, executive functioning) are significantly impaired.
- Single-domain vs. multiple-domain MCI: In single-domain MCI, only memory or one other domain of cognition is impaired. In multiple-domain MCI, memory plus one or more other cognitive abilities are affected.

Individuals with amnestic MCI, single or multiple domains, may be at increased risk of Alzheimer's dementia. Individuals with non-amnestic MCI may be at increased risk of other dementias, such as frontotemporal dementia. Having multiple domain MCI appears to increase the risk of future dementia.

Figure 2.2 [38] represents the diagnostic algorithm that can be used to reach a diagnosis of a particular subtype of MCI. This diagnostic process usually begins with a person or an informant who knows the person well, expressing some complaint about the person's cognitive function. When presented with these complaints, the clinician must first establish whether this constitutes normal cognition or suspicion of dementia. This can be done by taking a history and performing a mental status examination, possibly supplemented with neuropsychological testing. If the clinician determines that the patient is neither normal for age nor demented, but has experienced a cognitive decline by history with largely preserved functional activities, then the patient can be described as having MCI [34].

Once the diagnosis of MCI is established, the next task is to identify the clinical subtype. Here, the clinician must first determine whether the memory is impaired, as memory impairment strongly predisposes the individual to Alzheimer's disease. This can be deterred by office memory testing, usually involving an instrument with a delayed memory component, or by more detailed neuropsychological testing. If it is determined that memory is impaired due to age and education, the physician may assume

that this is an amnestic subtype of MCI. If, on the other hand, memory is found to be relatively spared, but the person has impairment in other cognitive domains unrelated to memory, such as language, executive function, or visuospatial skills, this constitutes a non-amnestic subtype of MCI.

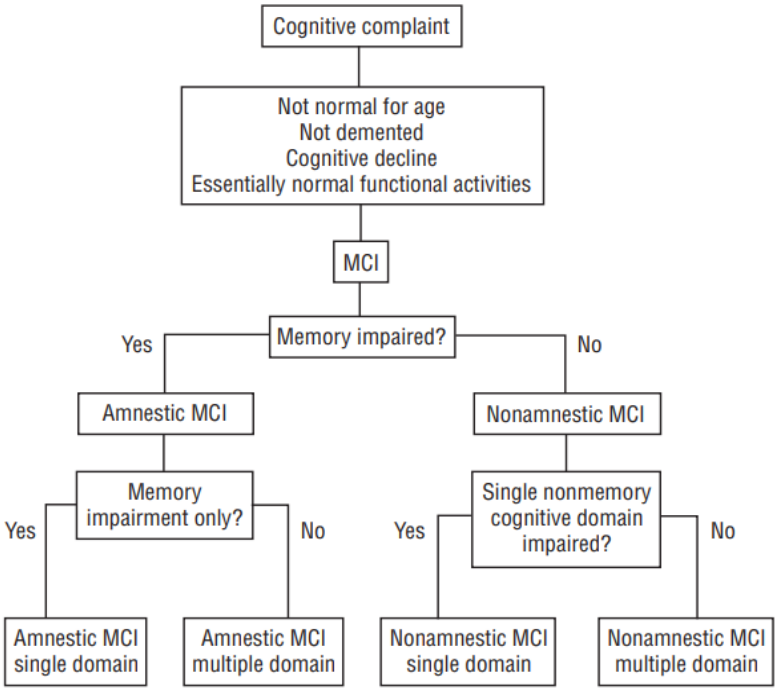


Figure 2.2 - Current diagnostic algorithm for diagnosing and subtyping MCI. Retrieved from [38]

Finally, the clinician must determine whether other cognitive domains are also impaired. This can also be addressed using neuropsychological testing or other relatively brief office instruments. A single-domain diagnosis of amnestic MCI has assumed if the impairment involves only the memory domain, whereas multiple-domain amnestic MCI concerns impairments in the memory domain plus at least one other cognitive domain, such as language, executive function, or visuospatial skills. Similarly, a diagnosis of single-domain MCI non-amnestic is assumed if there is impairment in a single non-memory domain, whereas multiple-domain MCI non-amnestic refers to impairments in multiple non-memory domains [34].

After the clinical characterization of the patient's symptoms has been determined, the next step involves determining the aetiology of the symptoms. Combining clinical syndrome with putative etiologies can be useful in predicting the final type of dementia to which these disorders will progress.

There are several rating scales useful for characterizing individuals along a continuous period from normal ageing to various stages of dementia. While these scales are useful for describing individuals at various levels of involvement, they do not necessarily coincide with the clinically relevant conditions of normal ageing, MCI, and AD [39].

For example, the Clinical Dementia Rating (CDR) is a scale, developed by Hughes et al. (1982), used to associate psychometric and behavioural measures to allow the assessment of several characteristics of the examined patient. This scale describes a continuum from normal (CDR 0) to questionable dementia (CDR 0.5), through mild (CDR 1), moderate (CDR 2), and severe (CDR 3) dementia. Some researchers believe that a CDR of 0.5 is equivalent to MCI [38], [39].

However, it should be noted that the CDR is a severity rating scale and not a diagnostic tool.

Another example is the Global Deterioration Scale (GDS), which includes individuals from GDS 1 (normal) to GDS 2 (normal with subjective memory disturbances), GDS 3 (mild dementia), and GDS 4 to 7 (more severe stages of dementia). Within this classification scale, individuals with MCI could correspond to a GDS of 2 or 3.

These potential relationships are depicted in Figure 2.3 [39].

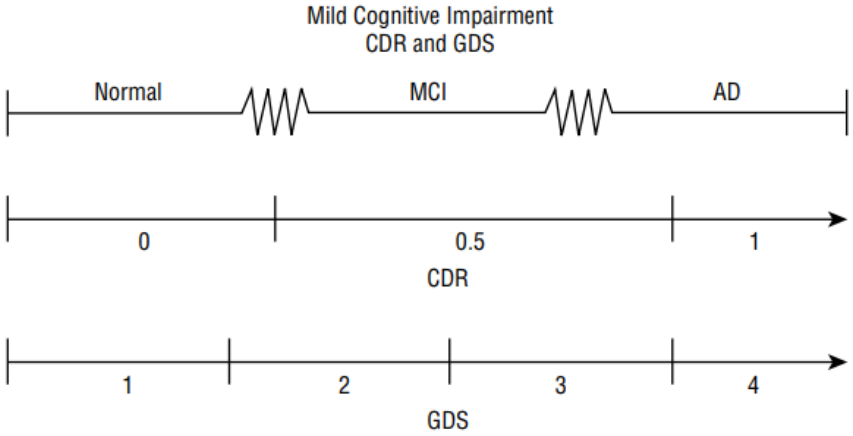


Figure 2.3 - Comparison between the clinical diagnoses and the approximate stages in the classification scales (CDR and GDS). Retrieved from [39]

Another example is the Mini-Mental State Examination (MMSE), developed by Folstein et al. (1975), which is one of the most widely used and most studied tests worldwide to assess cognitive functioning. The score may range from 0 to 30, and its relationship with the CDR is illustrated in Figure 2.4 [40].

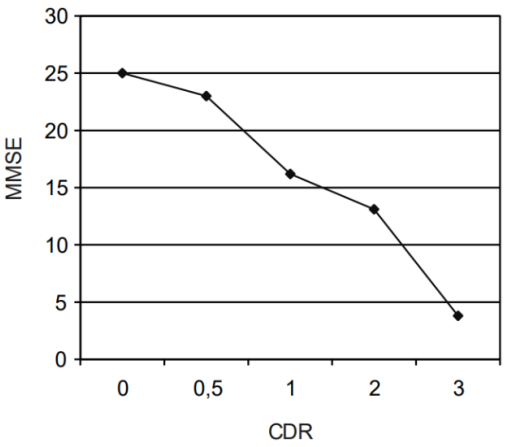


Figure 2.4 - Mean MMSE scores according to classification CDR. Retrieved from [40]

This test is carried out by experienced specialists and consists of a conversation between the specialist and the person in question. In this interview, tasks are defined that are divided into 6 cognitive domains: orientation, repetition, verbal recall, attention and calculation, language and visual construction. An example of this exam, in Portuguese, can be seen in [41].

Recently, the Montreal Cognitive Assessment (MoCA) has been developed. This test represents a brief cognitive screening tool with high sensitivity and specificity for the detection of MCI as currently conceptualised, in patients presenting in the normal range on the MMSE [37].

All patients suspected to have MCI should be subjected to a detailed assessment of physical, neurological, cognitive, psychological and functional status. It is important to identify potentially reversible causes of MCI, such as depression, thyroid disease, vitamin B12 deficiency and folate deficiency. Particular attention should be paid to prescription history [33].

Some studies suggest that up to 10-20% of older adults aged 65 and over have MCI. There is no single cause of MCI, just as there is no single outcome for the disease. MCI symptoms can remain stable for years, progress to Alzheimer's disease or another type of dementia, or improve over time (Figure 2.5 – adapted from [28]). In general, studies estimate that 10-15% of people with MCI progress to Alzheimer's disease each year.

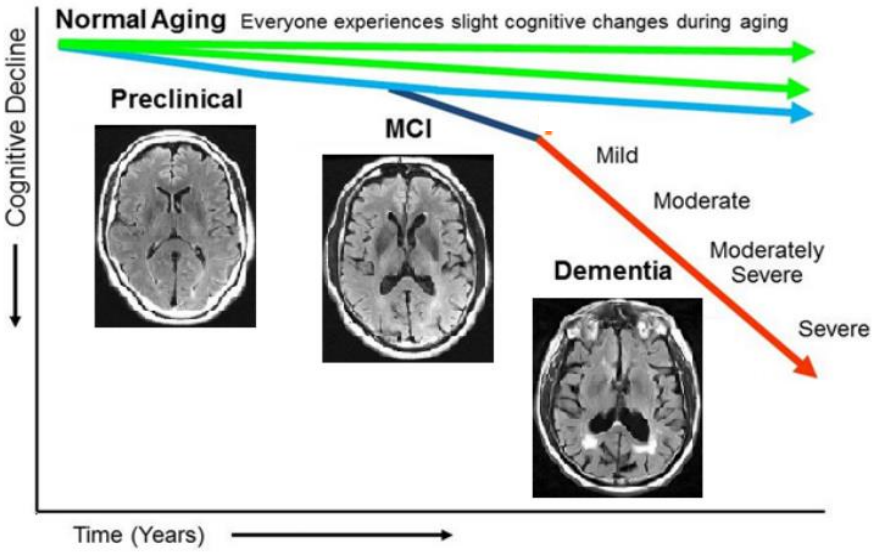


Figure 2.5 - Progression from normal aging to Alzheimer's disease or another dementia. Adapted from [28]

All individuals who clinically present with mild cognitive symptoms may not ultimately meet the same fate. Some may go on to develop AD, while others may progress to another dementia. The heterogeneity of MCI derives from the types of MCI already mentioned above and is depicted in Figure 2.6 [39].

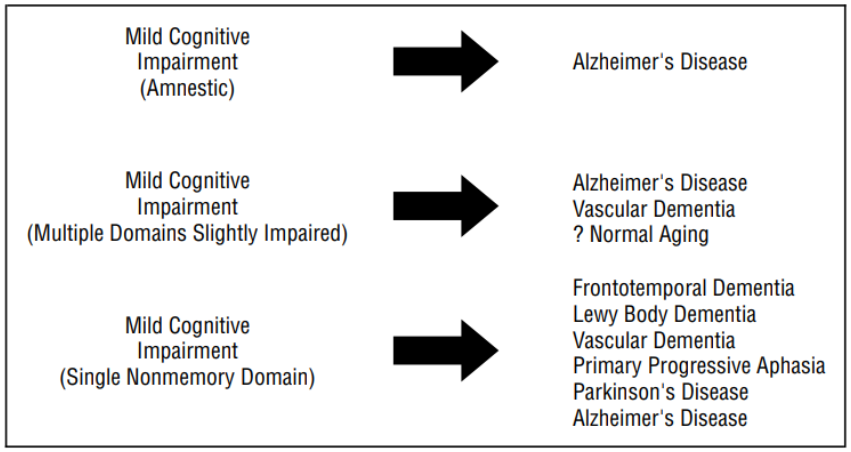


Figure 2.6 - Heterogeneity of the term MCI. Retrieved from [39]

Thus, patients with MCI are an important group to study for both theoretical and practical reasons. From a theoretical point of view, research on mild cognitive impairment will aid in the discovery of earlier clinical symptoms and signs that are likely to lead to Alzheimer's disease. From a practical point of view, it is useful to advise people who have varying degrees of memory impairment about the likelihood of this clinical syndrome progressing to a more severe form of impairment and possibly AD [42].

2.2.1 ADNI

The Alzheimer's Disease Neuroimaging Initiative (ADNI) is a consortium of universities and medical centres in the United States and Canada established to develop standardized imaging techniques and biomarker procedures in normal subjects, with MCI, and with AD [43].

The main goals of ADNI are [44]:

- To detect Alzheimer's disease at the earliest possible stage (pre-dementia) and identify ways to track disease progression with biomarkers.
- To support advances in intervention, prevention and treatment of ADNI by applying new diagnostic methods at the earliest possible stages (when intervention can be most effective)
- To continually administer ADNI's innovative data access policy, which provides all data without embargo to all scientists worldwide.

ADNI began in 2004 under the leadership of Dr Michael Weiner. The initial five-year study (ADNI-1) was extended by two years in 2009 (ADNI-GO), and in 2011 and 2016 (ADNI-2, and ADNI-3, respectively). Each of these stages is described in Table 2.2 – adapted from [44].

Table 2.2 - Different ADNI studies over the years

STUDY CHARACTERISTICS	ADNI-1	ADNI-GO (Grand Opportunities)	ADNI-2	ADNI-3
Primary goal	Develop biomarkers as outcome measures for clinical trials	Examine biomarkers in earlier stages of the disease	Develop biomarkers as predictors of cognitive decline, and as outcome measures	Study the use of tau PET and functional imaging techniques in clinical trials
Duration/start date	5 years/October 2004	2 years/September 2009	5 years/September 2011	5 years/September 2016
Cohort	200 elderly controls 400 MCI 200 AD	Existing ADNI-1 **+** 200 MCI	Existing ADNI-1 and ADNI-GO **+** 150 elderly controls 250 MCI 150 AD	Existing ADNI-1, ADNI-GO, ADNI-2 **+** 133 elderly controls 151 MCI 87 AD

A total of 819 participants were recruited and received a baseline assessment as part of the study. There were 229 normal control subjects, 398 subjects with MCI, and 192 subjects with AD enrolled. The mean age of the 3 groups was equivalent (approximately 75 years old) [43].

Concerning memory complaints, the normal subjects had none, whereas the subjects with MCI and with AD both had complaints. On the Mini-Mental State Examination, the range for normal subjects and subjects with MCI was 24 -30, and for subjects with AD 20 -26; all are inclusive. The CDR score for normal subjects was 0 and for subjects with MCI was 0.5, with a mandatory requirement that the memory box score be 0.5 or higher, and the score for subjects with AD was 0.5 or 1 [43].

This subject population serves as an excellent resource for studying the role of imaging and chemical biomarkers in tracking the AD disease process.

2.2.2 ANATOMICAL AND PHYSIOLOGICAL CONCEPTS

The nervous system represents an organism's communications network. The Nervous System is divided into two fundamental parts: the central nervous system and the peripheral nervous system. The Central Nervous System consists of the brain and spinal cord. The peripheral nervous system is made up of nerves that originate in the brain and spinal cord [45].

The brain is the "central computer" of the body, located inside the cranium, where all the information received converges. The brain represents only 2% of our body mass but consumes more than 20% of our

oxygen. It commands activities such as the control of motor actions, the integration of sensory stimuli and neurological activities such as memory and speech [46].

The brain contains grey matter and white matter. The grey matter is composed of the cell bodies, or neurons, which reside in the cerebral cortex and within the deep nuclei. The white matter is made up of the axons that extend from these cell bodies to various target tissues; axons are myelinated, which means they are sheathed in a fat-containing substance called myelin that speeds the transmission of electrical impulses travelling along the axon [6].

On an anatomical scale, the brain is divided into two hemispheres, the right and left. Each hemisphere controls several functions, for example, the right hemisphere is what gives us the ability to recognise faces and objects. The left side of the brain controls our ability to read and write, as well as identify grammatical rules [47].

Each hemisphere is then divided into four lobes: the frontal, parietal, temporal, and occipital lobes. Each lobe is composed of the cerebral cortex and the connected axons that project to form the white matter tracts. Table 2.3 summarises the different lobes and their primary functions (adapted from [6]).

Table 2.3 - Summary of brain lobe/region and primary functions

Lobe/Region	Function
Frontal lobe (primary motor cortex)	Responsible for physical movements, and the functions of learning, thinking, memory and speech.
Parietal lobe (primary sensory cortex)	Responsible for the perception of touch, pressure, vibration, temperature, and taste.
Temporal lobe (auditory, olfactory cortex)	Responsible for the perception of sounds and smells.
Occipital lobe (visual cortex)	Responsible for the perception of visual stimuli.

Figure 2.7 shows the anatomy of the brain with an indication of the different functional areas [48].

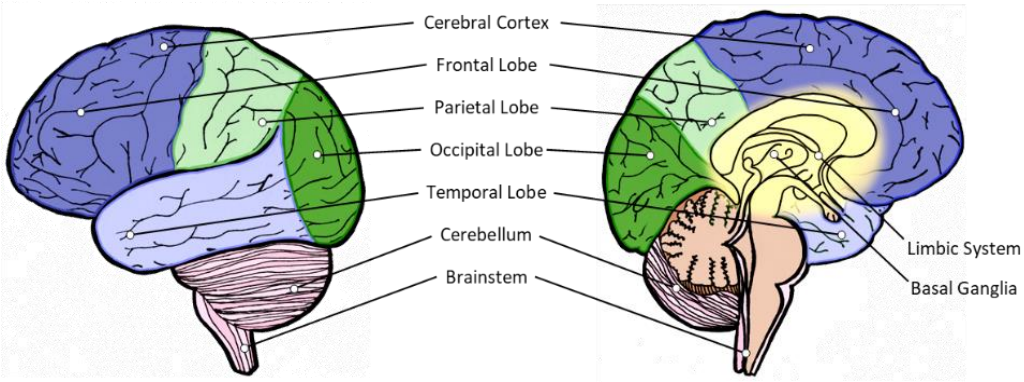


Figure 2.7 - Brain Anatomy. Retrieved from [48]

The brain is one of the body's hardest-working organs. When it's healthy, it works quickly and automatically. But when problems occur, the results can be devastating. Some of the main types of disorders include degenerative diseases of adult life (such as Parkinson's disease and Alzheimer's disease), infectious diseases (such as AIDS), trauma (such as spinal cord and head injury), metabolic diseases (such as Gaucher's disease), neurogenetic diseases (such as Huntington's disease and muscular dystrophy), developmental disorders (such as cerebral palsy), cerebrovascular diseases (such as stroke and vascular dementia), seizure disorders (such as epilepsy) and/or brain tumours [47].

Let's focus on Alzheimer's disease. Changes in the brain can begin a decade or more before symptoms appear. The damage seems to occur initially in the hippocampus and entorhinal cortex, which are parts of the brain that are essential in forming memories. As more neurons die, additional parts of the brain are affected and begin to shrink. In the final stage of Alzheimer's disease, the damage is widespread, and the brain tissue has shrunk significantly.

Figure 2.8 shows an example of brain MRIs from CN, MCI, and AD patients [49].

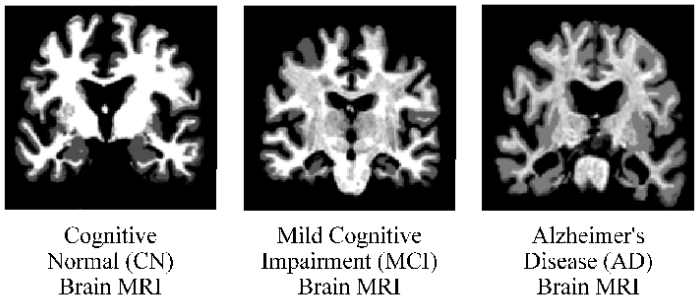


Figure 2.8 - Sample brain MRIs of CN, MCI, and AD patients. Retrieved from [49]

Knowing more about the brain can lead to the development of new treatments for diseases and disorders of the nervous system and improve many areas of human health.

2.3 MAGNETIC RESONANCE IMAGING

Neuroimaging is a branch of medical imaging that focuses on the brain. It is divided into two broad categories [50]:

- Structural imaging, which deals with the structure of the nervous system and the diagnosis of intracranial diseases such as tumours or injuries.
- Functional imaging, which is used to diagnose metabolic diseases (such as Alzheimer's disease), for neurological and cognitive psychology research, and builds brain-computer interfaces.

Ultrasound, Computed Tomography (CT), and Magnetic Resonance Imaging (MRI) are the most commonly used electronic modalities. This chapter will focus on MRI.

MRI is a non-invasive imaging technology that produces detailed 3D anatomical images that can be viewed from different angles, which are created by a strong magnetic field and radio waves [51], [52].

Unlike x-ray and CT scans, MRI does not use radiation. MRI is based on the magnetic properties of a hydrogen atom. The hydrogen nucleus hydrogen is composed of a single proton. When a radiofrequency current is pulsed through the wearer, the protons are stimulated and spin out of balance, fighting against the attraction of the magnetic field [53]. When the radiofrequency field is switched off, MRI sensors can detect the energy released as the protons realign with the magnetic field (Figure 2.9) [54].

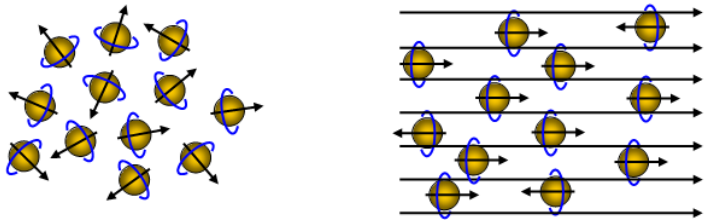
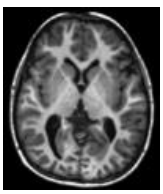
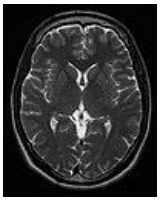


Figure 2.9 - Proton's alignment from natural random state to the direction of magnetic field. Retrieved from [54]

The time for this to happen, as well as the amount of energy released, is variable, depending on the environment and the chemical nature of the molecules, and it is through this that the difference between various types of tissue can be seen.

Based on this, different types of MRI sequences have been developed. The three most common ones are summarised in Table 2.4 (adapted from [55]).

Table 2.4 - Summary of T1, T2 and FLAIR sequences

Sequence	MR sample	Recognition	
T1 (T1 weighted)		**FAT**	Bright
		WATER	Dark
		WM	Light
		CORTEX	Grey
T2 (T2 weighted)		**FAT**	Light
		WATER	Bright
		WM	Dark Grey
		CORTEX	Light Grey

FLAIR (Fluid-Attenuated Inversion Recovery)		**FAT** Light **WATER** Dark **WM** Dark Grey **CORTEX** Light Grey

To obtain an MRI image, the patient is then placed inside a large magnet, where they will remain motionless during the imaging process. Before entering the magnet, patients must be given contrast agents (Gadolinium [56]) intravenously to increase the speed at which the protons realign with the magnetic field. The faster the protons realign, the brighter the image will be.

MRI scanners are particularly suitable for imaging non-bone or soft tissue parts of the body. The brain, spinal cord and nerves, as well as muscles, ligaments and tendons are seen much more clearly with MRI than with regular X-rays or CT scans [57].

In the brain, MRI can differentiate between white and grey matter and can also be used to diagnose aneurysms and tumours. One type of specialised MRI is functional magnetic resonance imaging (fMRI) which is used to look at brain structures and determine which areas of the brain 'activate' (consume more oxygen) during various cognitive tasks. It is used to advance understanding of brain organisation and offers a potential new standard for assessing the neurological status and neurosurgical risk.

The benefits and risks of using MRI are shown in Table 2.5 (adapted from [58]).

Table 2.5 - Benefits and risks of using MRI

Benefits	Risks
non-invasive	employs a strong magnetic field
does not involve exposure to radiation	implanted devices may malfunction or distort images
on soft tissue structures of the body - such as the heart or liver - are more likely to identify and characterise disease (focal lesions and tumours) accurately	gadolinium is eliminated from the body through the kidneys, which means that patients with severe kidney disease may have problems.
detect abnormalities that may be obscured by bone	
gadolinium MRI contrast material is less likely to cause an allergic reaction than the iodine-based contrast materials used for x-rays and CT	

The technician performs the examination, acquires and archives the images, generally in PACS systems (Picture Archiving and Communication System). These images, after being obtained and processed, may later be provided to the doctor and patient through web visualisation systems or saved on external devices. Generally, the format used is an international standard - DICOM (Digital Imaging and

Communications in Medicine). Finally, the images will be interpreted by the Medical Neuroradiologist who will prepare a written report that will accompany the MRI images [59].

2.4 RADIOMICS

Radiomics is a field of research concerned with the extraction of quantitative metrics - so-called radiomic features - in medical images [60].

These radiomic data can be used to discover previously unknown markers and patterns of evolution, progression and response to the treatment of a certain disease.

The radiometric process can be divided into several stages, as depicted in Figure 2.10 [60]: image acquisition and reconstruction, image segmentation and rendering, feature extraction and feature qualification and databases and data sharing for eventual ad hoc informatics analysis [61].

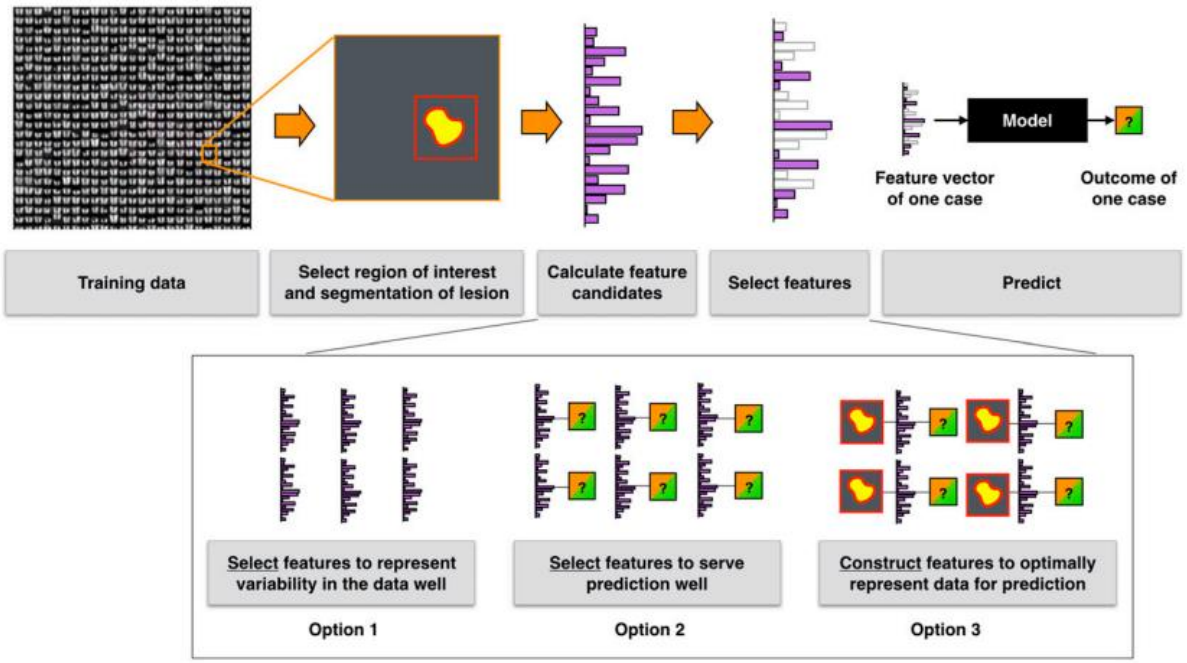


Figure 2.10 - Radiomics workflow. Retrieved from [60]

There are three different classes when it comes to radiomic features: First-order features, shape features and textural features [62].

First-order statistics describe the distribution of voxel intensities within the image region defined by the mask through basic and commonly used metrics. They include mean, median, minimum, maximum, asymmetry, kurtosis, uniformity, entropy, and others.

Shape Features (2D/3D) include descriptors of the two and three-dimensional size and shape of the

Region of Interest (ROI). These features are independent of the grey level intensity distribution in the ROI and are therefore only computed on the image and non-derived mask.

The textural features are the most complex and provide information about the intensities and geometry of the voxel. Many values can be calculated in different subclasses of textural features such as Grey Level Co-occurrence Matrix (GLCM) feature values, Grey Level Size Zone Matrix (GLSZM) features, Grey Level Run Length Matrix (GLRLM) features, Neighbouring Grey Tone Difference Matrix (NGTDM) features and Grey Level Dependency Matrix (GLDM) features. These subclasses are defined in the following points [60]:

- GLCM is a second-order grey-level histogram that captures spatial relationships of pairs of pixels or voxels with predefined grey-level intensities, in different directions, and with a predefined distance between the pixels or voxels.
- GLRLM provides information about the spatial distribution of runs of consecutive pixels with the same grey level, in one or more directions, in 2 or 3 dimensions.
- GLSZM counts of the number of groups (so-called zones) of interconnected neighbouring pixels or voxels with the same grey level form the basis for the matrix. A more homogeneous texture will result in a wider and flatter matrix.
- NGTDM quantifies the sum of the differences between the grey level of a pixel or voxel and the average grey level of its neighbouring pixels or voxels within a predefined distance.
- GLDM quantifies grey-level dependencies in an image. A grey-level dependency is defined as the number of voxels connected at a certain distance that depends on the central voxel.

To extract radiomic features, it is necessary to mention the filters that must be applied to the images. The reason behind this is that the radiomics will be extracted not only from the original image but will also be extracted from the images originated by applying the filters. Examples of filters are: Original, Wavelet, LoG, Square, SquareRoot, Logarithm, Exponential, Gradient, LBP2D, LBP3D, among others [63].

Pyradiomics is a python package for the extraction of Radiomics features from medical images. The purpose is to establish a reference standard for Radiomic Analysis, and to provide a tested and maintained open-source platform for easy and reproducible extraction of Radiomic features [64].

After the radiomic features have been selected, they are used to predict target variables in the present or variables in the future. The target may be scalar (e.g. survival in months), as would be predicted by a regression model, or categorical (e.g. response status or receiver positivity), as would be predicted by a classification model [60]. The explanation of how these models work will be examined in the next section.

2.5 MACHINE LEARNING

John McCarthy (2004) defines Artificial Intelligence (AI) as "the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to be limited to biologically observable methods" [65].

A significant difference between humans and computers is that humans learn from experience, while computers need to be programmed to follow instructions. However, today, with Machine Learning, it is possible to make machines also learn from experiences [66].

Over the past decade, Machine Learning (ML) has become one of the keystones of information technology, and according to *Yousef Shaheen M (2021)*, it can be defined as "an Artificial Intelligence application that employs algorithms to discover patterns in massive volumes of data" [67].

In other words, ML is an evolving branch of computational algorithms that are designed to emulate human intelligence by learning from the surrounding environment [68]. They are considered the working horse in the new era of so-called big data.

ML techniques have become popular in recent years and range from pattern recognition, computer vision, spacecraft engineering, finance, entertainment and computational biology to biomedical and medical applications [69].

ML then involves defining a learning problem to solve a task based on inputs. Thus, the objective is the prediction of an output variable y based on a vector x of input variables. To achieve this, the input and output are assumed to approximately follow a functional relationship "$y = f(x)$" called the predictive model [70]. The four main learning algorithms include supervised, unsupervised, semi-supervised and reinforcement learning [71].

Figure 2.11 (adapted from [72]) explains three of them.

The term "supervised learning" comes from the fact that their algorithms need an example from a training set for learning. This situation is similar to that of a teacher supervising the learning process of a student. Such methods are used, when possible, outcomes and correct answers are already given for labelled training sets. Thus, the predictive model is obtained with the training data comprising examples where both x and y are known. Supervised learning is used to solve classification and regression problems. Classification is represented as a supervised learning problem that seeks to identify the group of subclasses to which an object belongs. Therefore, it requires the prediction of a class label. Regression, on the other hand, is a supervised learning problem that involves the prediction of a numerical label [66].

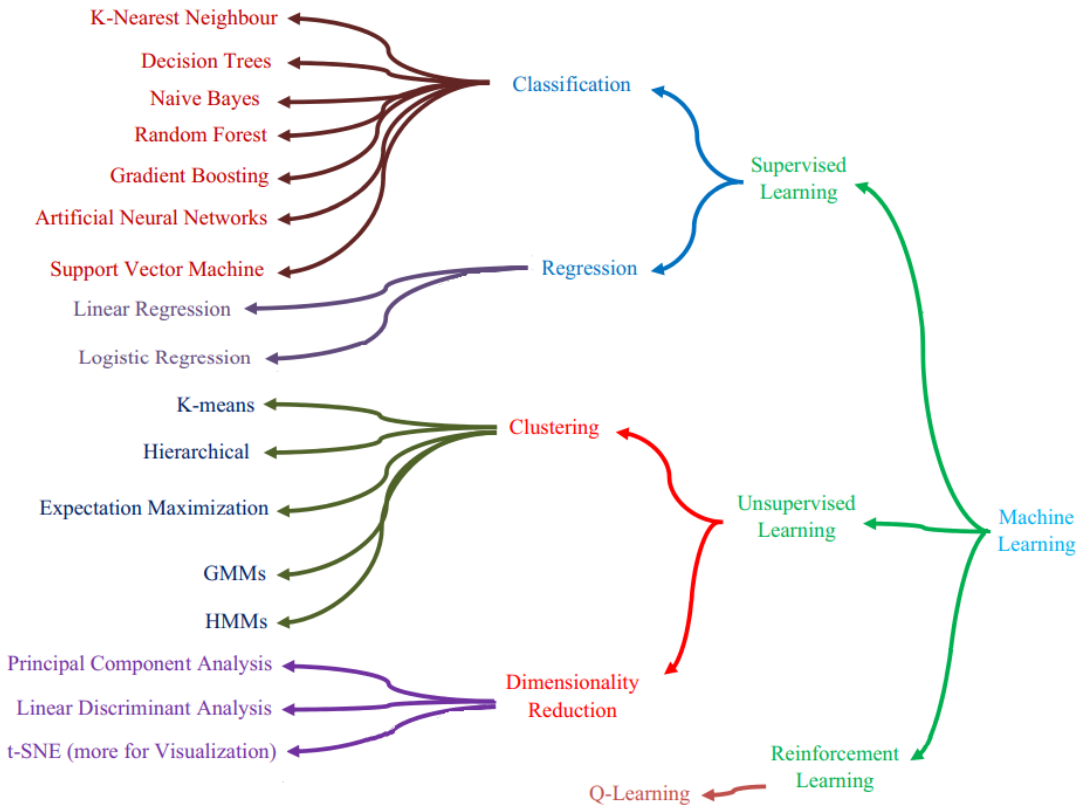


Figure 2.11 - ML algorithms related to supervised and unsupervised learning. Adapted from [72]

When you need to handle a large amount of unlabelled data and extract valuable information from it, “unsupervised learning” is the way to go. Unsupervised learning is closer to genuine AI. With unsupervised algorithms, a machine can identify complex processes without guidance, i.e. it starts learning from unlabelled data. Thus, it does not have an "instructor" to correct the model. Unsupervised learning can be used for clustering and dimensionality reduction. Clustering is a fundamental task of data mining. It is the process of separating data into meaningful sub-classes, called clusters, based on the similarity between objects in the data. Dimensionality reduction is a technique used when the number of features, or dimensions, in a given dataset is too high. It reduces the number of data entries to a manageable size while preserving the integrity of the data set as much as possible [66].

The term "semi-supervised learning" is due to both the use of labelled and unlabelled data in the learning process to overcome the drawbacks of the above-mentioned methods. The aim is to effectively use all the available data, not just the labelled information. Therefore, semi-supervised learning is a hybrid technique that uses a supervised learning algorithm to be trained on a labelled training set, then applies an unsupervised learning method to generate new labelled examples and adds them to the initial labelled training set. In general terms, the labelled part can be used to aid the learning of the unlabelled part [66].

At last, reinforcement learning is a form of behavioural training. This ML method allows machines to learn their optimal behaviour, i.e. how to act in a specific situation based on previous experience. In addition, these systems collect feedback (e.g. "this action was good, this action was bad") to improve performance. In other words, the machine will gradually learn from its experience, trial, and error by repeating the procedure several times. As a result of a series of successful decisions, the process will be "reinforced" because it addresses the problem better [66], [73].

2.5.1 ARTIFICIAL NEURAL NETWORK

A biological neural network is composed of billions of different types of neurons, the cells of the nervous system responsible for conducting nerve impulses. Neurons are connected with several dendrites and an element leaving them called axons. Dendrites are a structure that appears from the cell body, while the axon comes from the cell body, and is composed of nerve fibres. Each neuron can communicate with any other neuron by transmitting an impulse called synapses. The neuron receives signals through the dendrites of the axons, a phenomenon called synapse, which performs computation and generates a signal that is delivered through the axon [74].

Neurons are typically classified into three types based on their function. Sensory neurons respond to stimuli such as touch, sound, or light that affect the cells of the sensory organs, and they send signals to the spinal cord or brain. Motor neurons receive signals from the brain and spinal cord to control everything from muscle contractions to glandular output. Interneurons connect neurons to other neurons within the same region of the brain or spinal cord [75].

Figure 2.12 [76] shows the anatomy of a biological neuron.

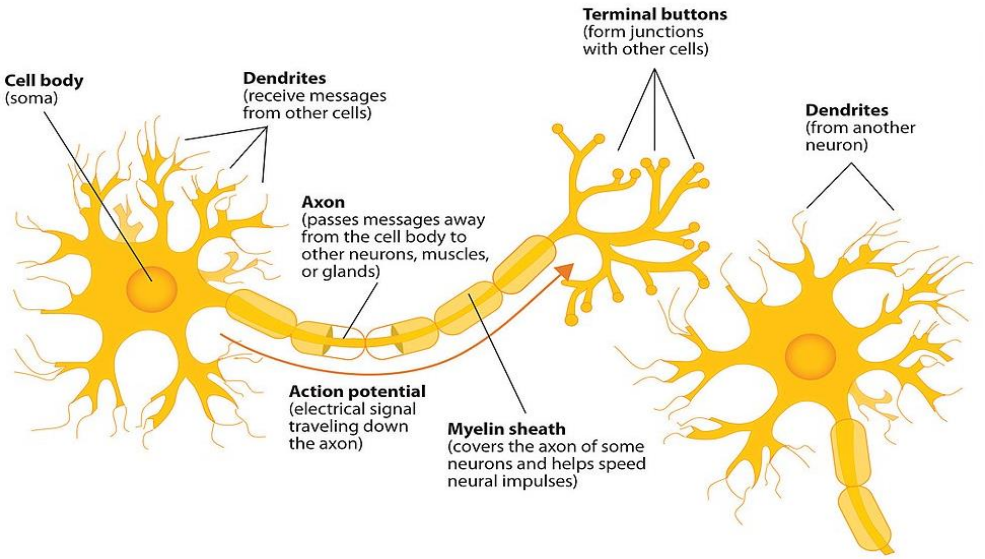


Figure 2.12 - Diagram of the components of a neuron. Retrieved from [76]

The artificial neural network is based on the biological neural network, also having artificial neurons. The connection between the artificial neurons symbolises the axon and dendrites, while each neuron or node performs the computation and transmits it to the next node [74].

One of the first definitions of ANN was given in "Neural Network Primer: Part I" (1989) by Dr Robert Hecht-Nielsen, as quoted below [77]: *" ... a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."*

It was Warren McCulloch and Walter Pitts who introduced the first artificial neural network in 1943. A handwritten problem can be an illustration of a neural network application. This problem consists of the ability of a computer to take a sequence of handwritten numbers (Figure 2.13 [78]) and detect which digits the sequence contains.

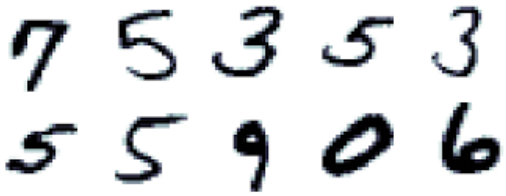


Figure 2.13 - Handwritten digits sample. Retrieved from [78]

By the human eye, it is easy to identify which numbers have the sequence but writing a computer program to do so becomes difficult. Neural networks develop a system that takes a training sample and automatically infers rules for recognising handwritten digits. Each time the training samples increase, the network learns more effectively, and consequently the accuracy increases. Over the years, researchers have found that if the network had more hidden layers, i.e., more layers between the input layer and the output layer, the network would be able to identify more features, and eventually could have more accuracy [74]. Although this type of network allows the learning of a larger number of features, it has the disadvantage of increased energy consumption and time calculations [79]. These sorts of neural networks are known as Deep Learning.

2.5.2 DEEP LEARNING

Deep Learning, also known as Deep Neural Networks (DNNs), is a subfield of machine learning and has become one of the most studied fields in AI. Although machine learning as yet become obsolete, it became essential to use neural nets as a pivotal resource to improve and reach other results in the learning mechanisms and it is now possible since there is the required computational power [74], [80].

Figure 2.14 [81] exhibits the relationship between AI and DL.

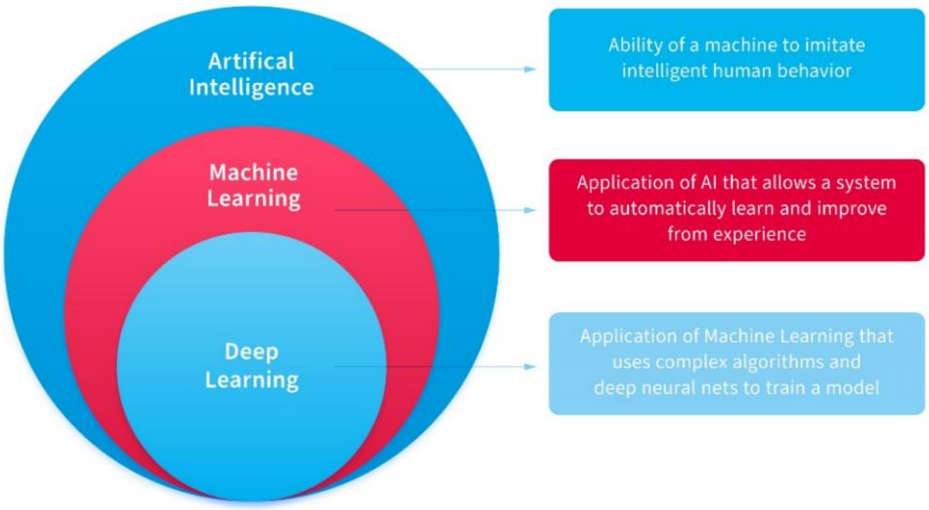


Figure 2.14 - Artificial Intelligence and its subfields . Retrieved from [81]

DL does not yet have a true definition, but it was defined early on as a "neural network of more than two hidden layers" (Figure 2.15) [82]. However, DL is much more than a neural network of more than two hidden layers. Deep learning uses neural networks, making it the closest analogue to humans in the way it learns and handles data. These neural networks allow deep learning networks to work with large amounts of data. When comparing deep learning with machine learning, the first not only become better at analysing data but can correct itself when high prediction accuracy is at stake [71], [80].

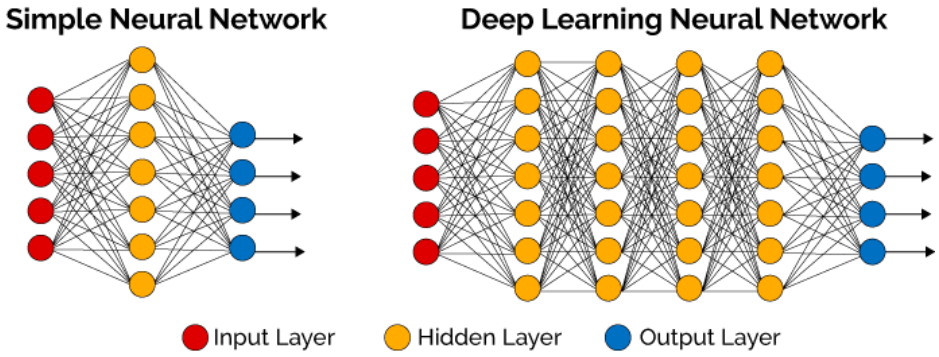


Figure 2.15 - Comparison between a Neural Network and a Deep Neural Network. Retrieved from [82]

Although DL is a relatively new field, Schmidhuber [83] built a thorough work reviewing the whole history of the developments in this field that led to the current state, while Arel et al. [84] emphasize the novelties of the last decade, both exhibiting all the promising results that this field may present [74].

As ML, also DL relies on advanced knowledge of mathematics, however, when creating the algorithms in machine learning, all the available frameworks and libraries offer an easy way to work with it without knowing the mathematics behind the neurons [85].

However, it becomes crucial to understand some core concepts when dealing with deep networks such as [85]:

- Layers: Being a fundamental architectural unit in deep neural networks, layers can be of various types and are changed by the activation function applied to them.
- Activation Functions: An activation function propagates the output of a layer to the adjacent layer, up to the output layer. Each neurone in the hidden layers will have the activation function defined for that layer.
- Loss Functions: Loss functions display how a neural net is doing when trained, quantifying how optimal the model is. The model will only be optimal if the parameters are well-chosen to minimize error.
- Optimizers: Optimizers are used to minimise the output of the loss function.
- Hyperparameters: Parameters that are tuned to make networks faster and well trained. The hyperparameters control the optimisation functions, model selection, and updaters which are used to update weights and minimise the error.

Layers

A layer in a deep learning model is a network structure or topology in the model architecture, which takes information from the previous layers and then passes the information to the next layer. There are several famous layers in deep learning, namely the convolutional layer and the maximum clustering layer in the convolutional neural network [86].

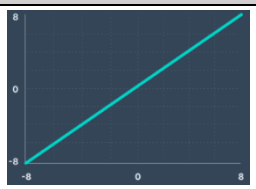
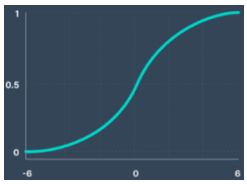
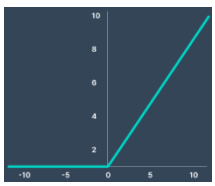
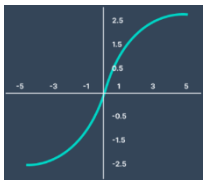
Activation Functions

In neural networks, the activation function implements the decision about the final output combined with the different weights. In human biology, the activation of the neural network happens because there is a potential difference and it will be activated if the value of the neuron is large enough to pass to the next neuron [80].

Activation functions are an extremely important element because they decide whether a neuron should be activated or not. That is, whether the information the neuron is receiving is relevant to the input or should be ignored [87].

There are several types of activation functions, and four will be discussed: the Linear, the Sigmoid, the Rectified Linear (ReLU) and the Hyperbolic Tangent (TanH), explained in Table 2.6 (adapted from [80], [88]).

Table 2.6 - Activation Functions

Activation	Equation	Plot	Characteristics
Linear	$f(x) = mx + b$		Makes gradient-based learning faster.
Sigmoid	$\sigma(z) = \frac{1}{1+e^{-z}}$		For Binary classification problems; Gradient vanishing.
ReLU	$f(x) = max(0, x)$		Used as a hidden unit. Leads to large and consistent gradients.
TanH	$f(x) = tanh(x)$		Mean value equals zero, faster than sigmoid but still with gradient vanishing.

SoftMax, another trigger function, can transform any score into probabilities. It is commonly used as the output layer for multi-classification tasks. It normalizes the results of the previous tasks, so their probabilities sum to one, and the output depicts the probability of each class [89].

Loss Functions and Optimizers

Output's evaluation gives the DL model the ability to understand whether the learning is adequate or not [80]. After the parameters are set, the learning algorithm tends to predict the target value for each training example and produces its loss function values. The loss function represents the measurement of the difference between the target value and the predicted value. It, therefore, produces a value for incorrect predictions. The learning algorithm will then search for the set of weights and bias parameters that can minimise the value of the loss function.

There are loss functions more suitable than others for certain models, however, cross-entropy is one of the most widely used. Besides quantifying the difference between two probability distributions, it is used in multi-classification tasks, although it is mainly based on binary cross-entropy [85].

The loss function is metaphorically a "mountain" and a visual example is shown in Figure 2.16 [90]. The main objective is to find the lowest place (global minimum).

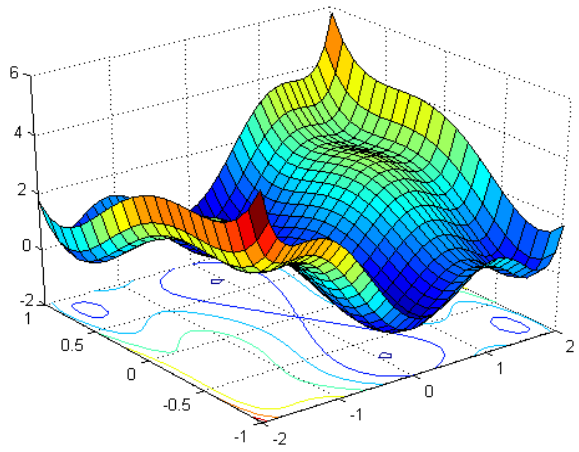


Figure 2.16 - Loss-function example graph. Retrieved from [90]

To minimise the value of the loss function, an optimiser should be used. Stochastic gradient descent (SGD) [91], which follows the same principle as gradient descent by finding local minima or maxima per iteration, Adaptive Moment Estimation (Adam) [92] and Root Mean Square Propagation (RMSprop) [93] are three of the most widely used optimisers. They all have different ways of choosing the direction, speed and jumps to achieve the global minimum [94].

Like loss functions, these optimisers will not be further explained in this master thesis, since a good explanation would encompass an explanation of statistics and differentiable functions.

Hyperparameters

Hyperparameters can be thought of as the tuning knobs of your model, i.e. a parameter whose value is used to control the learning process [95]. The reason is that neural networks are notoriously difficult to set up and many parameters need to be set. Also, individual models can be very slow to train.

Hyperparameters can be classified as model hyperparameters, which cannot be inferred while fitting the machine to the training set because they refer to the model selection task, or algorithm hyperparameters, which in principle do not influence model performance but affect the speed and quality of the learning process [85]. Examples of model hyperparameters are a number of hidden layers and units and dropout.

Layer size is defined by the number of neurons in a given layer [85]. Hidden layers are the layers between the input layer and the output layer. Many hidden units within a layer with regularization techniques can increase accuracy. A smaller number of units may cause underfitting [96].

Dropout is a regularization technique to avoid overfitting (increase validation accuracy), thus increasing the power of generalization [96]:

- Generally, use a small dropout value of 20%-50% of neurons with 20% providing a good starting point.
- It is likely to get better performance when dropout is used in a larger network, giving the model a greater opportunity to learn independent representations.

In a broad sense, dropout is a technique in which randomly selected neurons are ignored during training, as seen in Figure 2.17 [97].

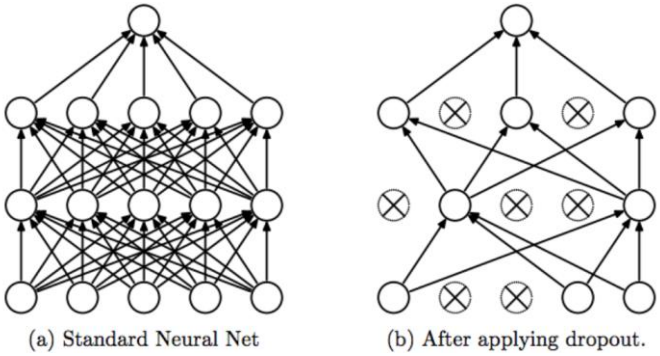


Figure 2.17 - Dropout: a simple way to prevent neural networks from overfitting. Retrieved from [97]

Examples of algorithm hyperparameters are the learning rate, the momentum, the number of epochs or the batch size.

The learning rate defines how quickly a network updates its parameters. As depicted in Figure 2.18 [95], a low learning rate slows down the learning process but converges smoothly. A higher learning rate speeds up learning but may not converge.

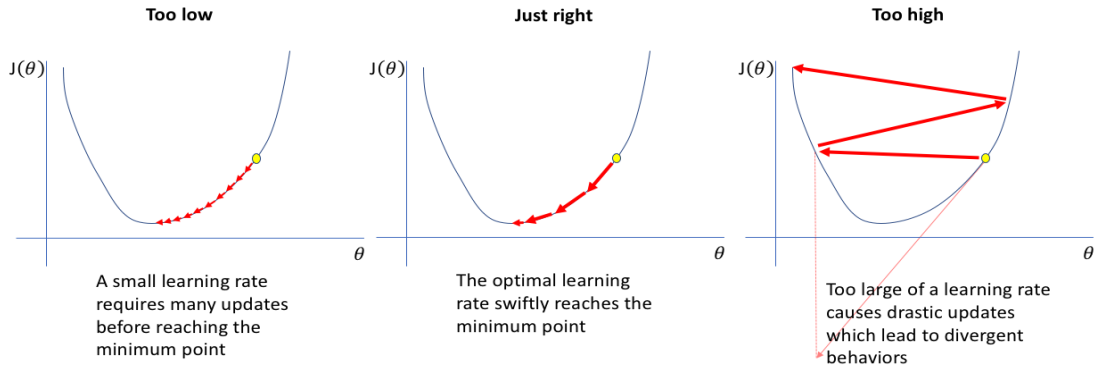


Figure 2.18 - Learning rate process. Retrieved from [95]

Momentum helps to know the direction of the next step with the knowledge of the previous steps. It helps to avoid oscillations. A typical choice of moment is between 0.5 to 0.9 [96]. Generically, it can be thought of as the mass of a ball that is rolling over the surface of the loss function. The heavier the ball, the faster it falls. But if it is too heavy, it can get stuck or overshoot the target [95].

The number of epochs is the number of times the whole training data is shown to the network during training. Increasing the number of epochs until validation accuracy starts to decrease, even when training accuracy is increasing (overfitting), is a possible adjustment [96].

The batch size is the number of subsamples given to the network after which the parameter update happens [85], [99]. A good default for the batch size might be 32.

2.5.3 MULTI-LAYER PERCEPTRON

A multilayer perceptron is a feed-forward artificial neural network that consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. An MLP is fully connected: given a pair of adjacent layers, each node in the left layer is connected to each node in the right layer. In addition to the input layer nodes, each node is a neuron and each layer of neurons involves a non-linear activation function. In addition, MLPs use a technique called backward error propagation (or simply backward propulsion). Figure 2.19 (adapted from [71]) shows the contents of an MLP with two hidden layers [71].

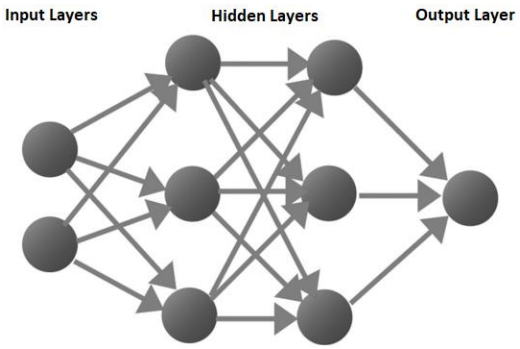


Figure 2.19 - An Example of an MLP. Adapted from [71]

MLPs are suitable for classification prediction problems where inputs are assigned to a class or label. They are also suitable for regression prediction problems where a valued real quantity is predicted given a set of inputs [98].

2.5.4 XGBOOST

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solves many data science problems in a fast and accurate way. The same code runs on a major distributed environment and can solve problems beyond billions of examples [99].

The name XGBoost (Figure 2.20 [100]), though, actually refers to the engineering goal to push the limit of computational resources for boosted tree algorithms. The reasons why many people use XGBoost are [101]:

- Execution speed: XGBoost was almost always faster than the other benchmarked implementations of R or Python Spark and is faster when compared to the other algorithms.
- Model performance: XGBoost dominates structured or tabular datasets on predictive modelling problems of classification and regression.

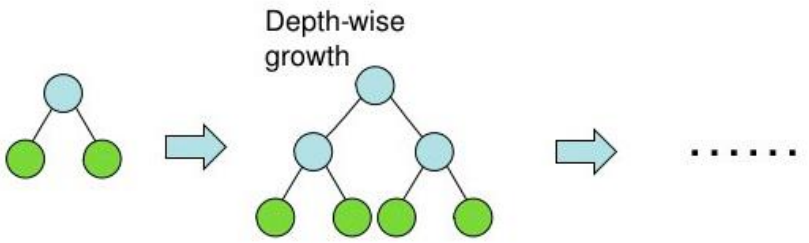


Figure 2.20 - XGBoost architecture. Retrieved from [100]

Therefore, XGBoost is a faster algorithm when compared to other algorithms because of its parallel and distributed computing. XGBoost is developed with both deep considerations in terms of systems optimization and principles in machine learning. The main advantages are briefly described in Figure 2.21 [101].

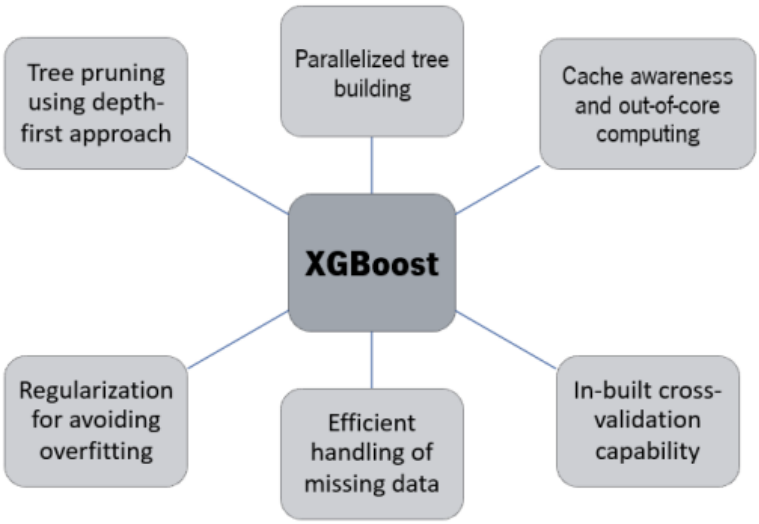


Figure 2.21 - Main advantages of XGBoost algorithm. Retrieved from [101]

XGBClassifier is a scikit-learn API-compatible class for classification. In addition, to build more robust models with XGBoost, one should always perform k-fold cross-validation. In this way, it is ensured that the original training dataset is used for both training and validation. Also, each entry is used for validation just once.

XGBoost provides a way to examine the importance of each feature in the original dataset within the model - *plot_importance()*. It involves counting the number of times each feature is split across all boosting trees in the model. Then is visualized the result as a bar graph, with the features ordered according to how many times they appear.

Another method is SHAP values (*SHapley Additive exPlanations*). It is a method based on cooperative game theory and is used to increase transparency and interpretability of machine learning models. Consider a cooperative game with the same number of players as the name of features. SHAP will disclose the individual contribution of each player (or feature) to the output of the model, for each example or observation [102].

Although SHAP shows the contribution or the importance of each feature on the prediction of the model, it does not evaluate the quality of the prediction itself. There is a big difference between both importance measures: feature importance is based on the decrease in model performance while SHAP is based on the magnitude of feature attributions.

2.5.5 CROSS-VALIDATION

Cross-validation is a statistical method for evaluating and comparing learning algorithms by dividing data into two parts: one for learning or training the model, and one for validating the model. In typical cross-validation, the training and validation sets must be crossed in consecutive rounds so that each data point has a chance to validate it. The basic form of cross-validation is k-fold cross-validation.

In k-fold cross-validation (Figure 2.22 [103]), the data is first divided into k equal (or nearly equal) segments or folds. Then perform k training and validation iterations so that in each iteration a different fold of the data is kept for validation, while the remaining k-1 folds are used for learning [104].

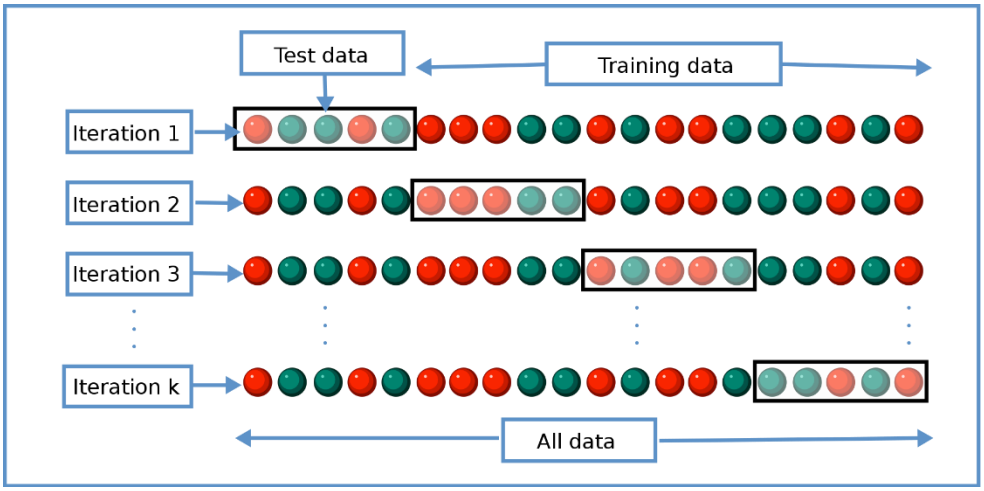


Figure 2.22 - K-fold cross validation diagram. Retrieved from [103]

2.5.6 CONFUSION MATRIX AND EVALUATION METRICS

One of the most powerful analytical tools in machine learning and data science is the confusion matrix. A confusion matrix is a predictive analytics tool. Specifically, it is a table that displays and compares actual values with the model's predicted values. Within the context of machine learning, a confusion matrix is utilized as a metric to analyse how a machine learning classifier performed on a dataset. A confusion matrix generates a visualization of metrics like precision, accuracy, specificity, and recall [105].

The reason that the confusion matrix is particularly useful is that, unlike other types of classification metrics such as simple accuracy, the confusion matrix generates a more complete picture of how a model performed. Only using a metric like accuracy can lead to a situation where the model is completely and consistently misidentifying one class, but it goes unnoticed because on average performance is good. Meanwhile, the confusion matrix gives a comparison of different values like False Negatives, True Negatives, False Positives, and True Positives [106].

Figure 2.23 [105] shows the components of a confusion matrix and the metrics that are possible to calculate.

		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Figure 2.23 - Components of confusion matrix. Retrieved from [105]

Recall is the number of genuinely positive examples divided by the number of false-negative examples and total positive examples. In other words, recall is representative of the proportion of true positive examples that a machine learning model has classified. This value may also be referred to as the "hit rate", and a related value is "sensitivity", which describes the likelihood of recall, or the rate of genuine positive predictions [105].

Like recall, precision is a value that tracks a model's performance in terms of positive example classification. Unlike recall though, precision is concerned with how many of the examples the model labelled positive were truly positive. To calculate this, the number of true positive examples is divided by the number of false-positive examples plus true positives[105].

While recall and precision are values that track positive examples and the true positive rate, specificity quantifies the true negative rate or the number of examples the model defined as negative that was truly negative. This is calculated by taking the number of examples classified as negative and dividing them by the number of false-positive examples combined with the true negative examples [105].

While one could spend time manually calculating metrics such as precision, recall, and specificity, these metrics are so commonly used that most machine-learning libraries have methods of displaying them. For example, Scikit-learn [107] for Python has a function that generates a confusion matrix.

2.6 FREESURFER

FreeSurfer is a software package for the analysis and visualization of structural, functional and diffusion neuroimaging data from cross-sectional and longitudinal studies. It is developed by the Laboratory for Computational Neuroimaging at the Athinoula A. Martinos Center for Biomedical Imaging.

It includes complete MR imaging data that involves skull-stripping, bias field correction, registration, and anatomical segmentation as well as cortical surface reconstruction, registration, and parcellation. FreeSurfer also includes fMRI and diffusion tractography toolboxes, a robust visualization interface, utilities for statistical group analysis, and much more [108].

FreeSurfer's "recon-all" performs all, or any part of the cortical reconstruction process, i.e. it segments the whole brain into its different parts.

Segmentation is a volume whose values represent an index of an anatomical structure or label to which the corresponding voxel in the main anatomical volume belongs. The structures are listed in a lookup table. This table also specifies the colours in which the structures should appear and is more commonly called the colour table [109] - a plot is represented in Table 2.7.

The development of FreeSurfer enabled a suite of automated image analysis tools to be created that were robust, accurate and relatively easy to use. This included automated geometric accurate topology correction, surface-based inter-subject alignment and whole-brain segmentation [110].

Table 2.7 – FreeSurfer Color LUT: Atlas

#No.	Label Name	R-G-B		#No.	Label Name	R-G-B
1	Left-Cerebral-Exterior	70-130-180		53	Right-Hippocampus	220-216-20
2	Left-Cerebral-White-Matter	245-245-245		54	Right-Amygdala	103-255-25
3	Left-Cerebral-Cortex	205-62-78		...	...	...
...	...	...		1006	ctx-lh-entorhinal	220-20-10
17	Left-Hippocampus	220 216 20		1007	ctx-lh-fusiform	180-220-140
18	Left-Amygdala	103 255 255		1008	ctx-lh-inferiorparietal	220-60-220
...	...	...		...	...	...

2.7 WORKING ENVIRONMENT

To make this thesis possible and create the DL model to ease the researchers' work, an IDE was needed to provide a place where code could be written and tested. However, instead of a common IDE, Jupyter Notebook was used. Jupyter Notebook (Figure 2.24) is an open-source and browser-based tool that supports workflows, code, data and visualization. It can handle text, code and output of said code in the same file which can be then exported to be shared in an editable way or even in formats like pdf.

Figure 2.24 - Jupyter Notebooks in a Virtual Environment. Retrieved from [111]

The files created by Jupyter Notebooks are called notebooks. The notebooks also allow long experiments to be split into smaller pieces that are executed independently giving the possibility of running only a part of the code that did not work instead of losing valuable time running all the code again [111].

Instead of the plain version of Jupyter Notebooks, JupyterLab was used in this thesis, the next-generation user interface for Jupyter Notebooks. It offers all the familiar building blocks of the classic Jupyter and some more in a dominant user interface [112].

The main libraries and packages used were [111]:

- Matplotlib: a software library for creating graphics and general data visualizations, made for and from the Python programming language.

- NumPy: a package for the Python language that supports arrays and multidimensional arrays, having a wide collection of mathematical functions to work with these structures.
- Scikit Learn: an open-source machine learning library for the Python programming language.
- PyRadiomics: an open-source Python package for the extraction of radiomic features from medical imaging.
- Pandas: a software library created for the Python language for data manipulation and analysis. It offers structures and operations for manipulating numerical tables and time series.
- Tensorflow: an open-source library for developing and creating Machine Learning models.
- Keras: an open-source neural network library written in Python. Keras is capable of running on top of TensorFlow (as explained above).
- NiBabel: a Python package that provides easy read and writes access to some of the most common neuroimaging file formats such as NIfTI.

3 DATA ACQUISITION AND SELECTION

As stated before, the goal of this work is not only to discover radiomics predictors of MCI but also to find out which areas of the brain are most affected by this development. As such, one of the most important materials used was, in fact, the MRI studies themselves.

The full datapipeline is available at: github.com/paulojorge99/ [113].

3.1 ADNI: DATA ACQUISITION

The dataset that was used was obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI – 2.2.1). The Image and Data Archive (IDA) is a secure online resource for archiving, exploring and sharing neuroscience data [114]. The dataset was extracted through ADNI advanced search (Figure 3.1).

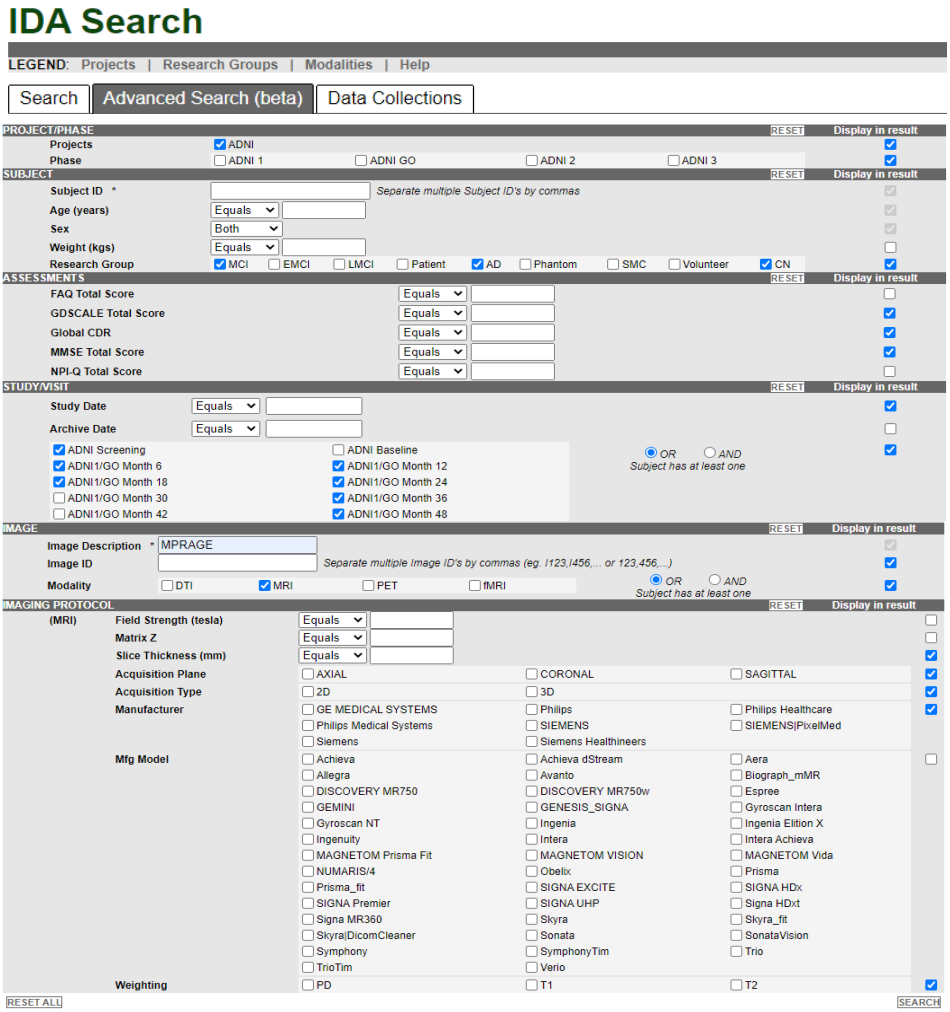


Figure 3.1 - ADNI advanced search. Adapted from [143]

All data were extracted from ADNI1/GO. These longitudinal MRI scans were obtained 6, 12, 18, 24, 36 and 48 months after the first scan. All the MRI scans were acquired using MPRAGE.

MPRAGE – three-dimension (3D) magnetization-prepared rapid gradient-echo – is designed for rapid acquisition with T1-weighted dominance. This sequence is one of the most popular sequences for structural brain imaging in clinical and research settings. The sequence captures high tissue contrast and provides high spatial resolution with whole brain coverage in a short scan time [115].

All scans were composed of 256 slices with an isotropic voxel resolution of 1 mm3, in NIfTI format.

Part of one of the studies contained in the dataset can be visualized in Figure 3.2 where some slices of that study can be seen.

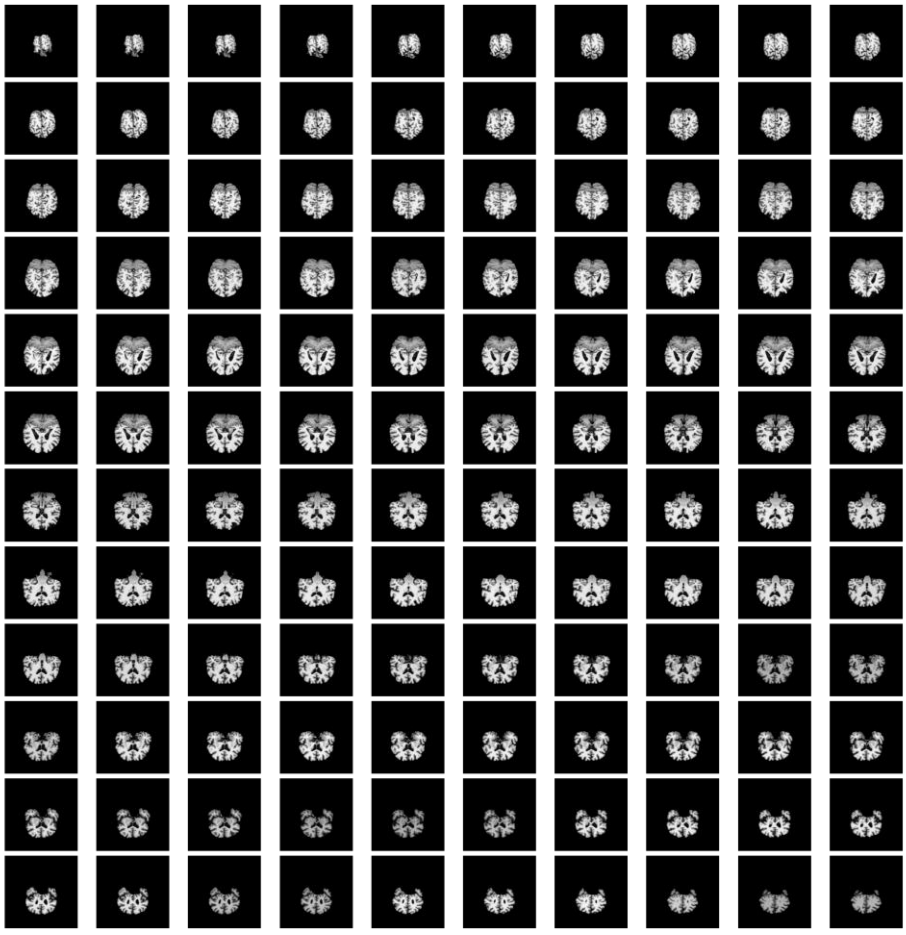


Figure 3.2 - Visualization of some slices in one of the studies

3.2 DATA SELECTION

Initially, 3602 exams from 736 patients were obtained from ADNI.

Figure 3.3 represents all the data processing that was carried out from the time the data was taken from ADNI to the time it was ready.

The pre-processing process involved was removal of non-brain tissue using the FSL, Brain Extraction Tool, followed by bias correction and non-uniform intensity corrected image.

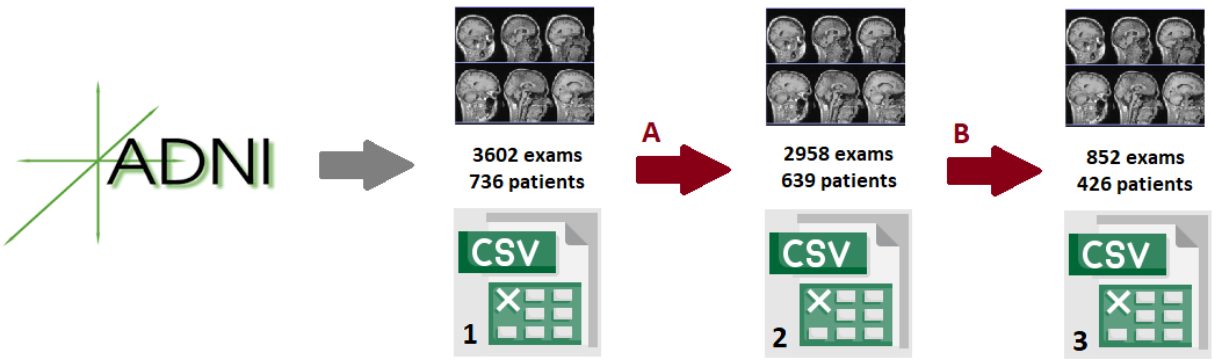


Figure 3.3 - Dataset creation process

The first step (Figure 3.3A) was to remove the repeated exams (with the same exam date). Then, all the patients who had only one exam performed were also eliminated, as they would not be of interest to the study. At the end of this process, there are 639 patients. All the processing described can be seen in "APPENDIX I: Data Selection".

Therefore, a new CSV file was created considering each line a single patient and all their exams in different columns (Figure 3.3B). The columns 'visit_0', 'visit_6', 'visit_12', 'visit_18', 'visit_24', 'visit_36' and 'visit_48' were created and filled only with 0 and 1, based on whether or not there was an exam on that date. The purpose of this was to study the number of exams in each time base to later choose which exams to use for the study. All the processing described can be seen in Figure 3.4

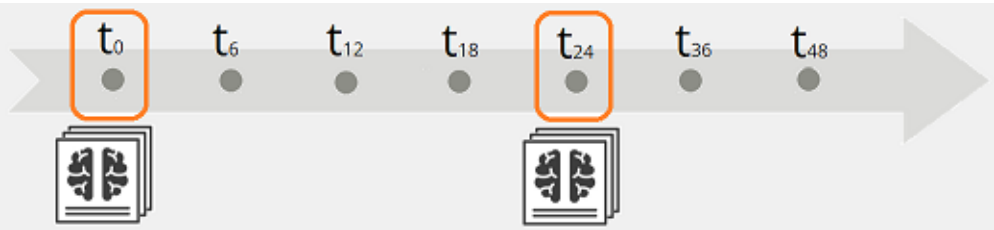


Figure 3.4 - Timeline for the selection of exams

Once all the existing exams had been checked, it was decided to use only the initial exams (month 0) and the exams performed 24 months later (month 24) in the study because this time base contained the largest number of exams with all the intended data (MMSE and CDR).

After eliminating all other exams and all patients without exams at the end of the 2nd year with all the intended data (Figure 3.3B), 852 tests were obtained from 426 patients.

3.3 DATA DESCRIPTION AND CLINIC INFORMATION

Once all the initial data processing was completed, a CSV was obtained with the data of all 426 patients, i.e. 852 exams. The Table 3.1 summarises all the data obtained in the dataset considered.

Table 3.1 - Data description

Patients:	426
Exams:	852 (two per patient - month 0 and month 24)
Sequence:	MPRAGE
Protocols:	Acquisition Plane=SAGITTAL; Slice Thickness=1.2; Acquisition Type=3D; Manufacturer=GE MEDICAL SYSTEMS; Weighting=T1
	Acquisition Plane=SAGITTAL; Slice Thickness=1.2; Acquisition Type=3D; Manufacturer=SIEMENS; Weighting=T1
Age Range: (month 0)	55.2 – 91.0 (years old)
Average Age:	75.3 (years old)
#Male:	241
#Female:	185

As an example, Figure 3.5 represents the distribution of patients by age.

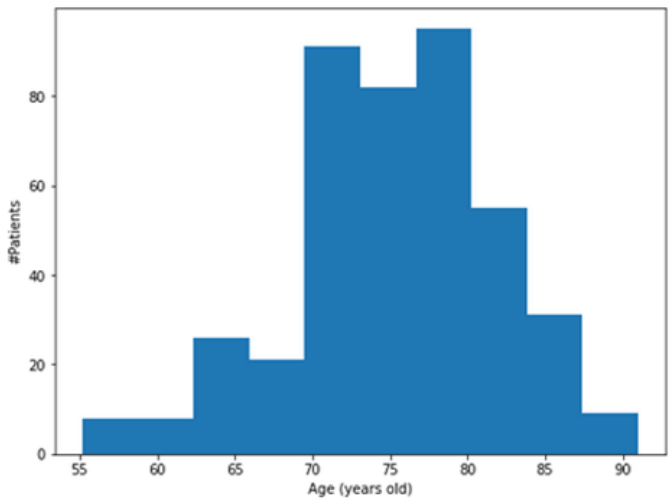


Figure 3.5 - Distribution of patients by age

4 MINING MRI PREDICTORS

The following subsections describe the approaches used throughout the work to obtain the predicted results. The 852 scans of the 426 patients are then used. However, only the initial 426 exams from each patient will be used. The exams of month 24 were used only to classify the transition of each initial exam.

Then, in the first stage, the segmentation of all 426 exams was performed. Once this segmentation is done, all radiomics features are extracted both from the original scans and from each desired zone. After the feature extraction, the datasets that will be used as input to the DL models are created. With the application of these models, the desired results are obtained.

The described pipeline is represented in Figure 4.1.

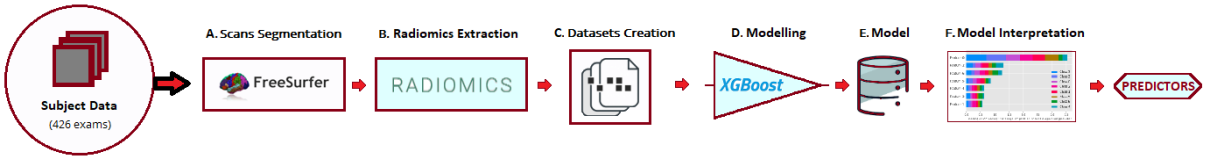


Figure 4.1 - Project Pipeline

The full datapipeline is available at: github.com/paulojorge99/ [113].

4.1 SCANS SEGMENTATION

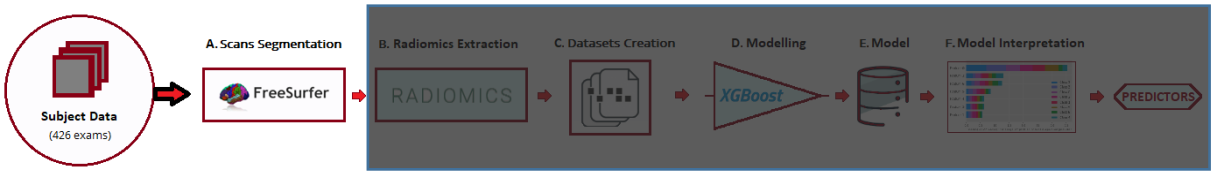


Figure 4.2 - Project Pipeline: Segmentation

FreeSurfer [108] – version 7 – was used for the segmentation of all 426 scans. The scheme in Figure 4.3 represents the whole process carried out to segment the exams.

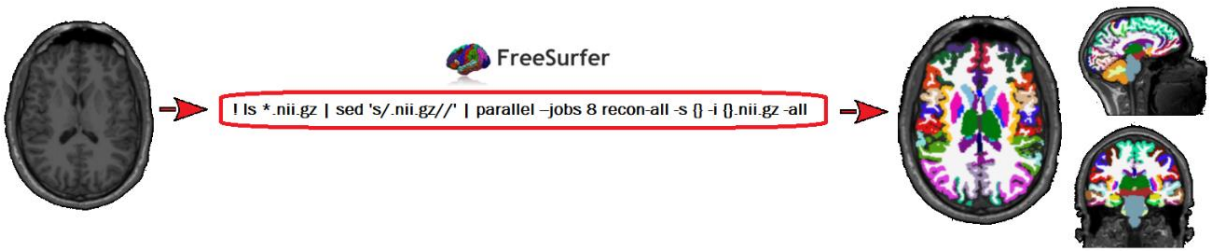


Figure 4.3 – FreeSurfer: Segmentation process

The concept of parallel computing was important in this project. As there were 426 scans to segment, and each exam takes an average of 6.6 hours – with 8 CPUs – (Figure 4.4), it would take about 4 months to segment everything. However, by splitting the segmentation process across different computers and servers, with several CPUs each, in about 15 days all 426 images were segmented.

```
Started at Thu Feb 10 20:32:51 UTC 2022
Ended   at Fri Feb 11 03:09:07 UTC 2022
#@#%# recon-all-run-time-hours 6.604
recon-all-s ADNI_002_S_0295_MR_MP-RAGE__br_raw_20060418193713091_1_S13408_I13722_generate finished without error at Fri Feb 11
03:09:07 UTC 2022
```

Figure 4.4 - "Recon -all" command execution time per exam

The atlas used for the segmentation was already presented above, in Table 2.7.

After segmenting all the images, the masks of the different areas of the brain and all the information (areas and volumes) of each area are obtained.

In this particular case, it will only work with the hippocampus, the entorhinal (entorhinal cortex) and the lateraloccipital. The first two are potential zones of interest in the area, and the last one will act as a control.

As an example, Figure 4.5 show a segmented scan of the aforementioned areas of interest. The software for the visualisation of scans and respective segmentations was 3D Slicer [116].

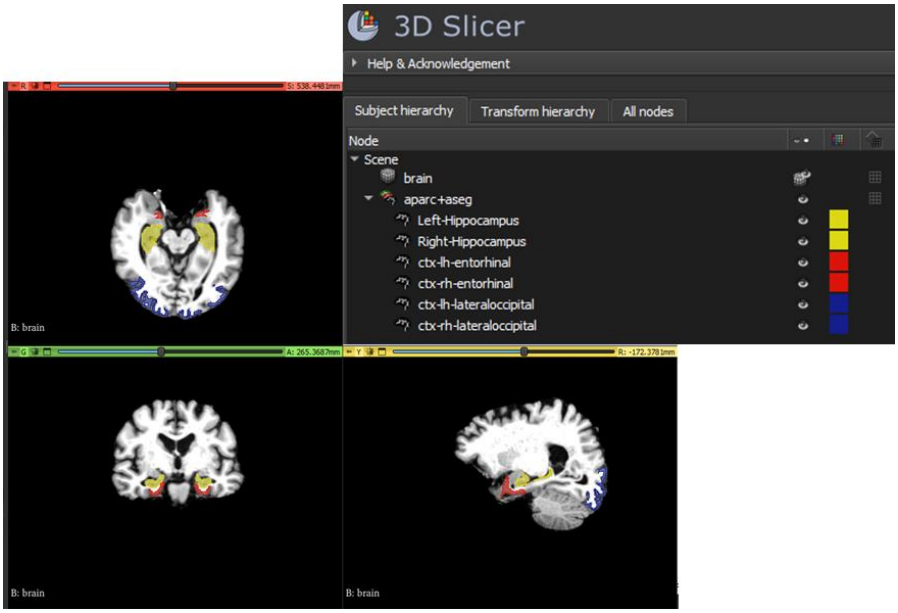


Figure 4.5 - Hippocampus, Entorhinal and LateralOccipital in 3D Slicer

Finally, to obtain not only the masks of each region but also the area and volume information, the script in "APPENDIX II: Scans Segmentation" was run.

Processing the different CSV files obtained results in a file as shown in Figure 4.6.

	Paciente	Left-Lateral-Ventricle_volume	Left-Inf-Lat-Vent_volume	Left-Cerebellum-White-Matter_volume	Left-Cerebellum-Cortex_volume	Right-Pallidum_volume	Right-Hippocampus_volume	Right-Amyg
1	002_S_0295	0.24668912908133656	0.15135807942164403	0.49619140800854766	0.4884190771326864	0.5685144529820709	0.5973694120178599	0.537702
2	002_S_0413	0.12400416114249005	0.16591928251121077	0.1859757378584249	0.2849180853408491	0.34970728137577756	0.589622895260638	0.518630
3	002_S_0559	0.15855433344392703	0.23221197292366455	0.4004213761352246	0.4259161551870378	0.39402671057446037	0.6427457098283932	0.748739
4	002_S_0685	0.2115084405893806	0.069634605875633	0.48449251787247066	0.42267363035884825	0.3821807537504574	0.5904836193447738	0.3596112
5	002_S_0729	0.06392006588034997	0.04537161758938772	0.03851810056603665	0.35089394840010946	0.31462678375411635	0.4855290763354672	0.2564568
6	002_S_0782	0.5076236941903948	0.27592968337056045	0.3485897104989886	0.632586205745046	0.5868551042810101	0.6239442681155521	0.4612066
7	002_S_0938	0.05813042035734539	0.07178298010196253	0.1395883479294345	0.15853976183638438	0.34467618002195394	0.1906503846360751	0.036168
8	002_S_1018	0.0867772691643215	0.04960016368565534	0.3255640857878594	0.5494025873411296	0.4847237467983901	0.5465597934262199	0.3146079
9	002_S_1070	0.19707794739882333	0.08886767037801158	0.5926037083500905	0.7060522677637598	0.4967069154774973 5	0.8434289095701758	0.570649

Figure 4.6 - Areas and Volumes of each region per patient

In addition to the masks of each brain region, FreeSurfer also provides as output the mask of the whole brain. This will be the mask of the whole brain used throughout the project. The dimensions of each scan or mask are 256x256x256.

4.2 RADIOMICS EXTRACTION

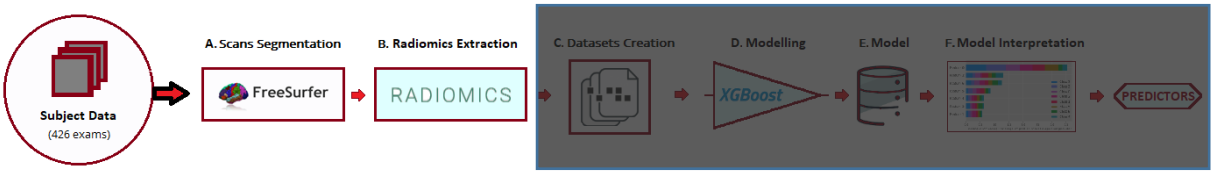


Figure 4.7 - Project Pipeline: Radiomics Extraction

The extraction of radiomic features has been implemented in Python using the open-source package PyRadiomics, as mentioned before. This process, when done through PyRadiomics, contains three fundamental steps: (I) loading the scans and the respective masks, (II) customizing the extraction by applying filters, (III) extraction of the radiomic features, and returning the features.

Before extracting the radiomics, it is necessary first to define in a ".yaml" file (Figure 4.9B) the features classes and filters to be used – all these concepts are explained in subchapter *2.4 Radiomics*.

In this case (Figure 4.8), the parameters used were as follows:

params.yaml

```yaml
setting:
  #force2D: true
  #force2Ddimension: 0
  minimumROIDimensions: 2

imageType:
  Original: {}
  Wavelet: {}
  LoG:
    sigma: [1.0, 2.0, 3.0, 4.0, 5.0]
  Square: {}
  SquareRoot: {}
  Logarithm: {}
  Exponential: {}
  Gradient: {}
  LBP2D: {}
  LBP3D: {}
```

Figure 4.8 - Parameters used to extract radiomics

Starting with the complete scans (the whole brain), the extraction of the radiomics features (Figure 4.9E) was done with pyradiomics in a single command line (Figure 4.9D): ***!pyradiomics*** *paths.csv* ***--param*** *params.yaml* ***--jobs 7 -o*** *features.csv* ***-f*** *csv*. For this, it was necessary to have a csv file with the paths of each exam and mask (Figure 4.9A/C). This process can be seen in the figure below.

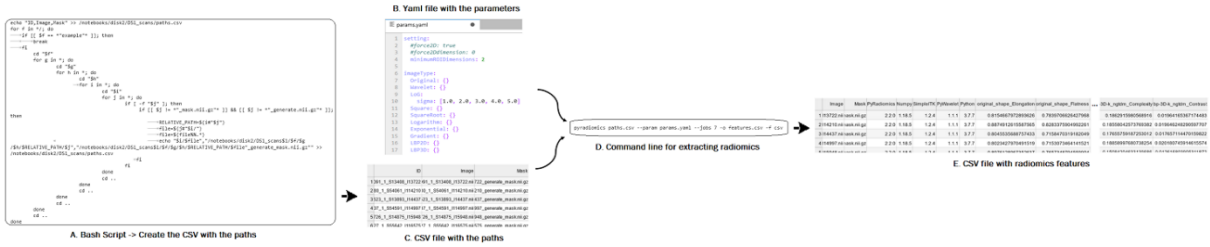


Figure 4.9 - Features radiomics extraction - Whole Brain

Moving on to each of the three regions already presented (hippocampus, entorhinal and lateraloccipital), the extraction of the features radiomics is done similarly, using PyRadiomics, but this time with a Python script. This script can be seen in appendix III.

Using the hippocampus as an example (Figure 4.10), the following results are obtained:

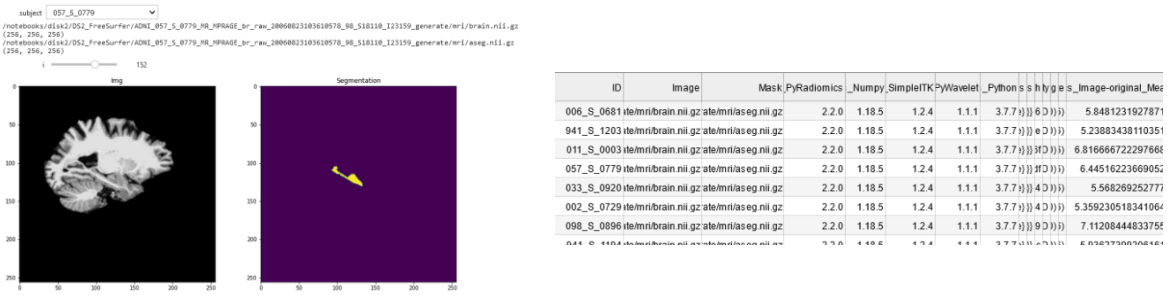


Figure 4.10 - Hippocampus and its features

4.2.1 PROCESSING THE EXTRACTED FEATURES

After extracting all 2178 radiomics, they are processed. It starts with the radiomics extracted from the whole brain and then for each of the segmented regions.

Once the radiomics have been read, the first step is to remove the non-numerical radiomics, as they are of no interest to the study. After having the file with only the 2004 numerical radiomics, the next step was to add to each patient/line the respective personal information (age, MMSE, CDR and sex - at month 0 and month 24).

With all this data it is now possible to apply the classification function below to classify each patient as being normal (CN), having MCI or AD:

```
1.  def classificacao(MMSE,CDR):
2.
3.      if MMSE > 24 and MMSE <=30 and CDR == 0.0:
4.          return "CN"
5.
6.      if MMSE > 24 and MMSE <=30 and CDR == 0.5:
7.          return "MCI"
8.
9.      if MMSE > 1 and MMSE <=24 and CDR == 0.5:
10.          return "AD"
11.
12.     if MMSE > 1 and MMSE <=30 and CDR > 0.5:
13.         return "AD"
14.
15.     else:
16.         return null
```

With the classifications done, it is then possible to create the column "Transition", where with the classification of month 0 and month 24 by the patient, it is possible to assign a transition to the exams of month 0. For example, if a certain patient is considered without dementia at month 0 and if MCI is detected at month 24, then the transition assigned to the month 0 exam is "CN-MCI".

After this, and only useful for running the model, it is important to associate each transition created with a number. Thus, a new column is created where the transition "CN-MCI" is associated to 0, "CN-AD" to 1, "MCI-AD" to 2, "CN-CN" to 3, "MCI-MCI" to 4, and "AD-AD" to 5.

Once this is done, as mentioned above, all the scans from month 24 and all the scans that do not fit into any of the above transitions are then deleted, leaving 406 scans from 406 patients.

Finally, two more columns are created. The "Class" column, where each test is assigned its initial classification, i.e. whether that test belongs to a patient without dementia, with MCI or with AD. The "Class_ID" column contains the numeric values associated with each classification, i.e. "CN" at 0, "MCI" at 1 and "AD" at 2.

The number of examinations by transitions and by classes is presented in Table 4.1.

Table 4.1 - Number of exams by transitions and by classes

Transitions	#
CN-MCI	16
CN-AD	1
MCI-AD	88
CN-CN	125
MCI-MCI	101
AD-AD	75

Classes	#
CN	142
MCI	189
AD	75

As shown in the table above, the CN-AD transition only presents one exam, which does not make sense to be present when using the model. Therefore, it was decided to remove this transition, leaving only the 4 remaining transitions for model training.

Last but not least, all the data were normalized. After all this processing, the different datasets to train the model, with the 405 exams of the 405 patients, will be created.

All this processing already explained is in APPENDIX IV: Processing of the Extracted Radiomics (Particular case example: Whole Brain) and represented in Figure 4.11.

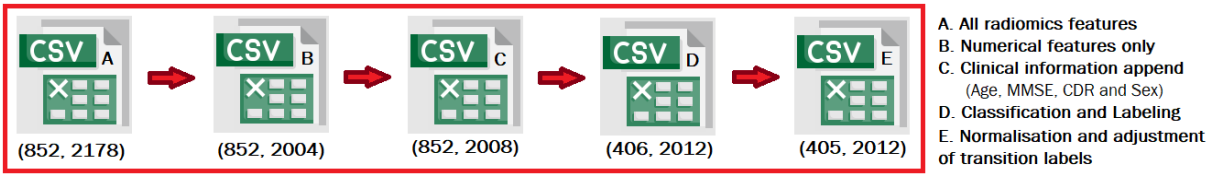


Figure 4.11 - Processing of the extracted features

4.3 DATASETS

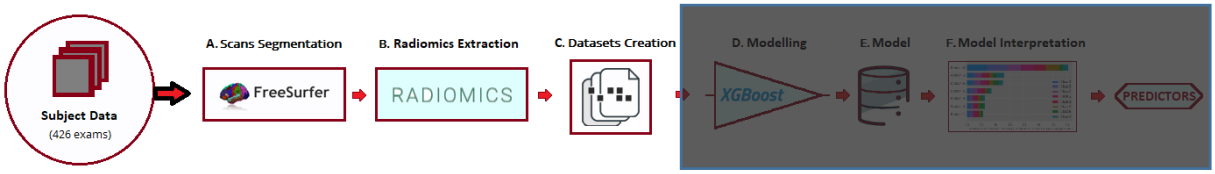


Figure 4.12 - Project Pipeline: Datasets

After processing all the data and radiomics features, one then has the datasets ready for model training. Four different datasets are then created, as shown in Figure 4.13.

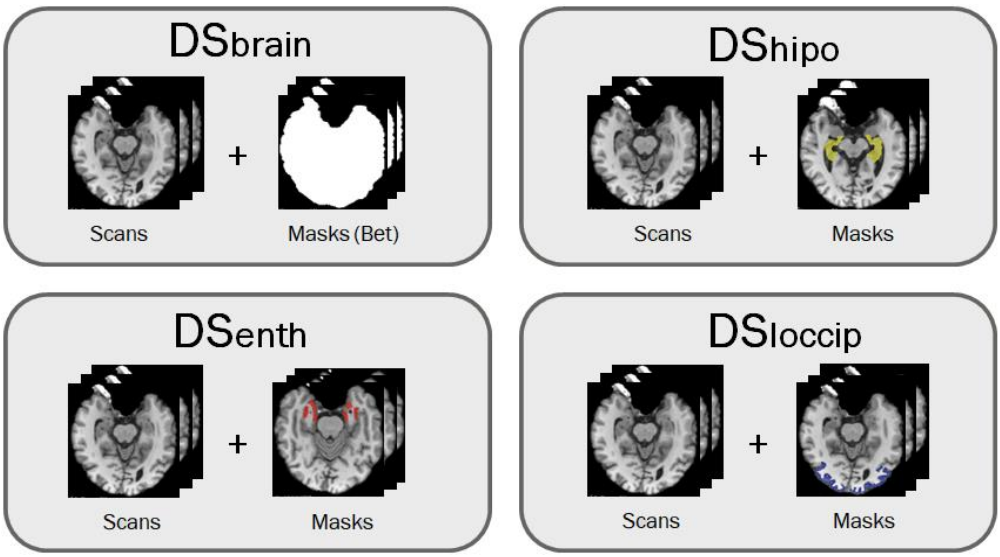


Figure 4.13 – ADNI: Datasets description

4.4 MODELLING AND MODEL INTERPRETATION

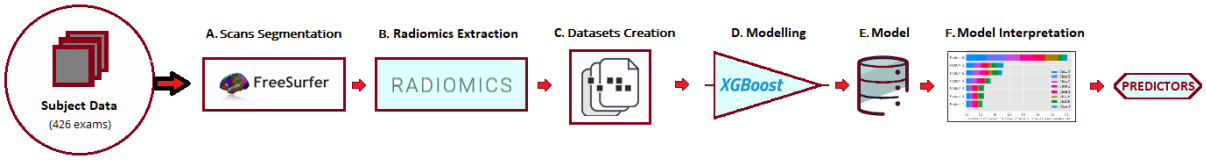


Figure 4.14 - Project Pipeline: Modelling and Model Interpretation

The first step in creating the DL model was to prepare the dataset to be compatible with the model. The 4 datasets mentioned above were then created. Thus, this subchapter will be divided into four subsections.

The model used was the XGBoost. This model has already been explained in 2.5.4. The primary reasons one should use this algorithm are its accuracy, efficiency and feasibility. It is a linear model and a tree-learning algorithm that does parallel computations on a single machine. It also has extra features for doing cross-validation and computing feature importance.

To build more robust models with XGBoost, one should always perform k-fold cross-validation. In this way, ensures that the original training dataset is used for both training and validation. Also, each entry is used for validation just once.

XGBoost provides a way to examine the importance of each feature in the original dataset within the model. It involves counting the number of times each feature is split across all boosting trees in the model. The result is then displayed as a bar chart, with the features ordered according to how many times they appear. XGBoost has a *plot_importance()* function that helps us to achieve this task. Then one can visualize the features that have been given the highest importance score among all the features. In addition, it is also possible to obtain the SHAP values.

Thus, XGBoost provides a way to do the feature selection.

4.4.1 DATASET 1: DSBRAIN

We will start with the first dataset that was created (Figure 4.15). This is made up of the whole brain exams and their respective masks.

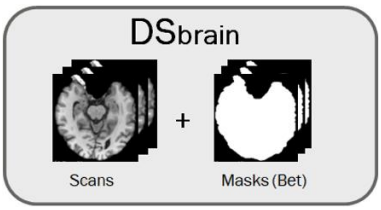


Figure 4.15 – Dsbrain (Whole Brain)

Before moving on to the model, it is always important to check all the data in our dataset. Therefore, we started by running data analysis functions (Figure 4.16 to Figure 4.19):

```
# Import dataset
CSV_FILE = "A3.DS_Brain.csv"
df=pd.read_csv(CSV_FILE)
```

Figure 4.16 - Import dataset: DSbrain

```
df.head()
```

	ID	Sex	Age	MMSE	CDR	Transition_Label	Transition	Class_Label	Class	original_shape_Elongation	...	lbp-3 k_glszm_SmallAreaHighGrayLevelEmpha
0	002_S_0295	1	84.9	28	0.0	2	CN-CN	0	CN	0.364516	...	0.171
1	002_S_0413	0	76.4	29	0.0	2	CN-CN	0	CN	0.324836	...	0.245
2	002_S_0559	1	79.4	30	0.0	2	CN-CN	0	CN	0.663188	...	0.113
3	002_S_0685	0	89.7	30	0.0	2	CN-CN	0	CN	0.356748	...	0.311
4	002_S_0729	0	65.3	27	0.5	1	MCI-AD	1	MCI	0.453070	...	0.235

5 rows × 2012 columns

Figure 4.17 – DSBrain: Function to see the first 5 lines of the dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 405 entries, 0 to 404
Columns: 2012 entries, ID to lbp-3D-k_ngtdm_Strength
dtypes: float64(2005), int64(4), object(3)
memory usage: 6.2+ MB
```

```
df.describe()
```

	Sex	Age	MMSE	CDR	Transition_Label	Class_Label	original_shape_Elongation	original_shape_Flatness	original_shape_LeastAxisLength	original_shape_MajorAxisLength	...	lbp-3 k_glszm_SmallAreaHighGrayLevelEmpha
count	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	...	405.000
mean	0.555556	75.415802	27.125926	0.377778	2.323457	0.837037	0.505093	0.456682	0.345788	0.324552	...	0.381
std	0.497519	6.524493	2.598828	0.313476	1.124246	0.712763	0.190909	0.176718	0.159573	0.153321	...	0.225
min	0.000000	55.200000	20.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000
25%	0.000000	71.700000	25.000000	0.000000	1.000000	0.000000	0.376717	0.323868	0.241236	0.234120	...	0.205
50%	1.000000	75.600000	28.000000	0.500000	2.000000	1.000000	0.477074	0.455551	0.330465	0.316339	...	0.301
75%	1.000000	79.800000	29.000000	0.500000	3.000000	1.000000	0.620688	0.576107	0.428650	0.400785	...	0.567
max	1.000000	91.000000	30.000000	1.000000	4.000000	2.000000	1.000000	1.000000	1.000000	1.000000	...	1.000

8 rows × 2009 columns

Figure 4.18 - DSBrain: Description of data

```
seaborn.distplot(df['Age'],kde=False,color='darkred',bins=30)
```

```
<AxesSubplot:xlabel='Age'>
```

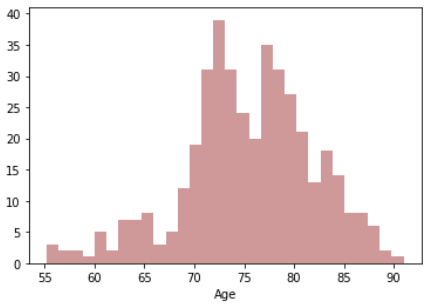


Figure 4.19 - DSBrain: Age distribution

After checking all the data, it is then necessary to define the variables to be used. To do that, the columns "ID", "Age", "MMSE", "CDR", "Transition", "Class_Label" and "Class" were removed. The Y variable (decision variable) was defined as the "*Transition_Label*" column, which contains the numbers previously associated to each transition. Finally, we eliminated this column and then defined the variable X, which is all the remaining dataset (just all the extracted features).

After this, an internal data structure is built that is used by XGBoost, which is optimised for both memory efficiency and training speed - DMatrix.

Being XGBoost the model to be used, it is necessary to define its parameters. As a base, the following parameters were used: *'objective':'binary:logistic'; 'max_depth': 4; 'alpha': 0; 'learning_rate': 0.05; 'n_estimators':1000; 'gamma':0.2; 'min_child_weight':3; 'lambda':0.9; 'subsample':0.8; 'colsample_bytree':0.7; 'reg_lambda':1.0; 'reg_alpha':0.0*.

The *cross_val_score()* function will be used to perform the evaluation, taking the dataset and cross-validation configuration and returning a list of scores calculated for each fold. The function run evaluates the model using 10-fold cross-validation. The average classification accuracy is then reported.

In this case, it is seen that the model achieved an estimated classification accuracy of about $0.30 \ +/- \ 0.05$.

To perform a model improvement, a technique was applied, the Random Search, which is a technique where random combinations of the hyperparameters are used to find the best solution for the built model. This technique is very useful when we have many parameters to try and the training time is very long. A fixed number of combinations of hyperparameters is chosen randomly, so not every single combination is evaluated.

Therefore, the new estimated parameters to be used by the model are: *'subsample': 1.0, 'reg_lambda': 1.0, 'reg_alpha': 0.1, 'min_child_weight': 3, 'max_depth': 5, 'learning_rate': 0.01, 'gamma': 0.2 and 'colsample_bytree': 1.0*, and the estimated classification accuracy increased of about $0.31 +/- 0.04$.

The following is a tabular summary of the number of correct and incorrect predictions made by the model, confusion matrix (Figure 4.20).

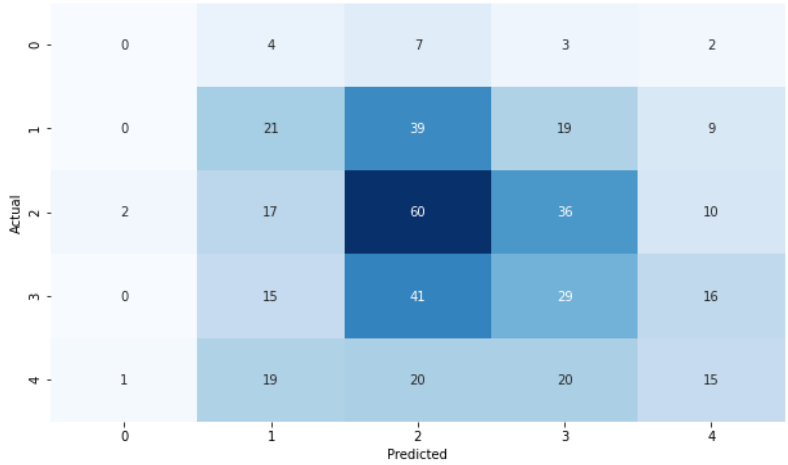


Figure 4.20 - DSBrain: Confusion Matrix : 'CN-MCI': 0, 'MCI-AD':1, 'CN-CN':2, 'MCI-MCI': 3, 'AD-AD':4

Based on the results obtained, and as mentioned above, we can see the Feature Importance graph (Figure 4.21). In this particular case, it was decided to choose the 15 most important features.

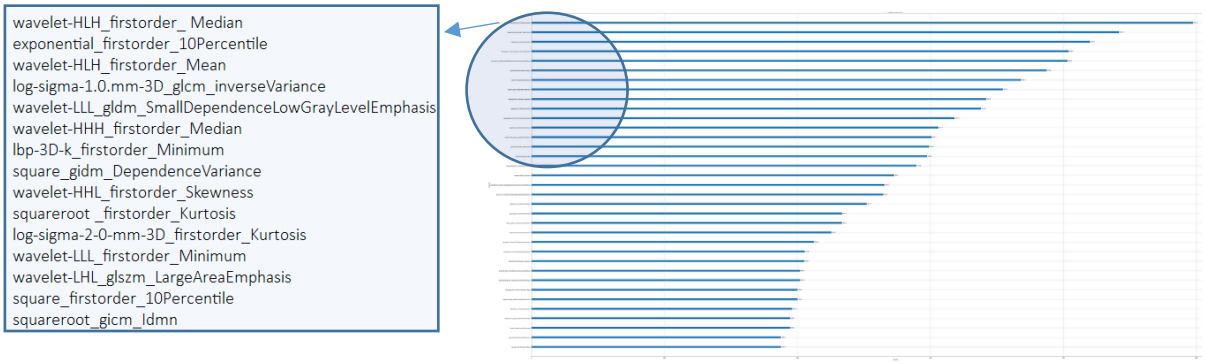


Figure 4.21 - DSBrain: Feature Importance

In addition to this graph, it is also important to present the ShapValues plot (Figure 4.22), which shows which features contributed the most to each class.

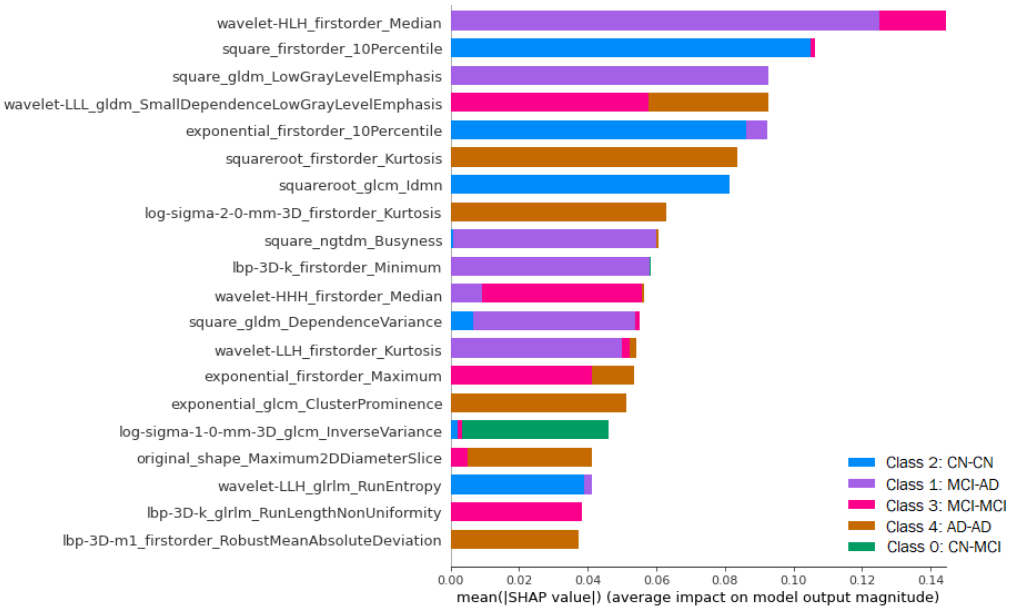


Figure 4.22 - DSBrain: ShapValues

Besides presenting the graph of the overall ShapValues, it is also important to represent the ShapValues only for the class on which the study focuses most - class 1: MCI-AD (Figure 4.23). In this representation, it is possible to see not only the important features of the intended class but also if each one of them contributes positively or negatively to the obtained result.

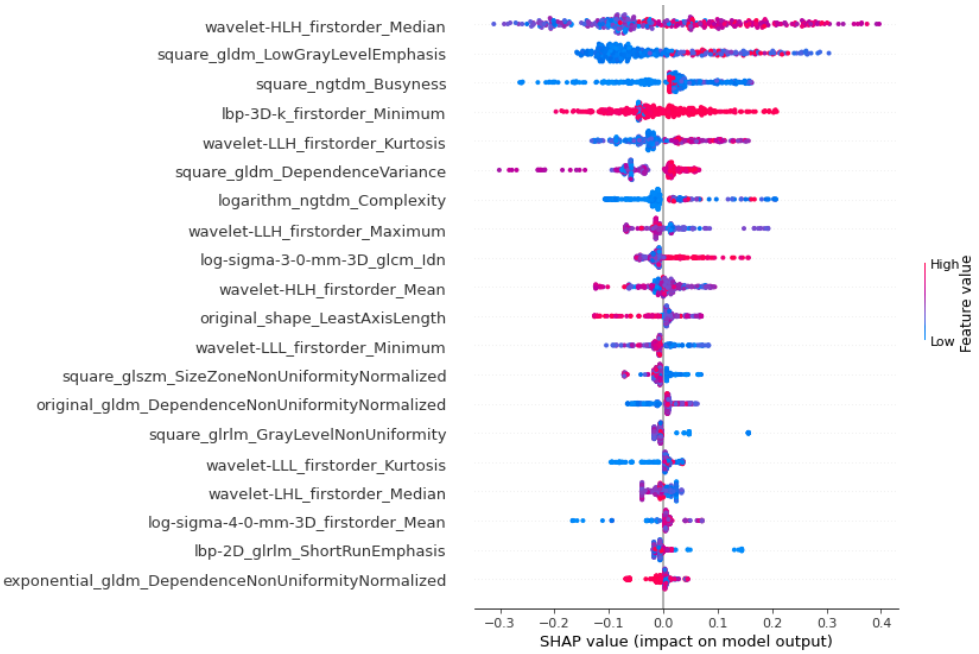


Figure 4.23 - DSBrain: ShapValues: Transition MCI-AD

All these graphs and results obtained will be analysed and commented on in the next subchapter.

4.4.2 DATASET 2: DSHIPO

Moving on to the next dataset (Figure 4.24), in this case, the Hippocampus, everything will be described similarly.

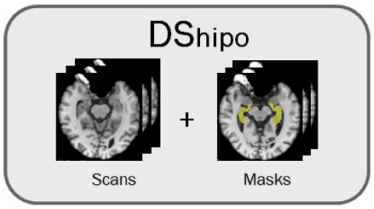


Figure 4.24 – Dshipo: Hippocampus

Before moving on to the model, it is always important to check all the data in our dataset. Therefore, we started by running data analysis functions (Figure 4.25 to Figure 4.29):

```
# Import dataset
CSV_FILE = "B2.DS_Hipo.csv"
df=pd.read_csv(CSV_FILE)
```

Figure 4.25 - Import dataset: DShipo

```
df.head()
```

	ID	Sex	Age	MMSE	CDR	Transition_Label	Transition	Class_Label	Class	original_shape_Elongation	...	lbp-3 k_glszm_SmallAreaHighGrayLevelEmpha
0	006_S_0681	0	77.1	30	0.0	2	CN-CN	0	CN	0.615742	...	0.445
1	941_S_1203	1	83.4	30	0.0	2	CN-CN	0	CN	0.637859	...	0.679
2	011_S_0003	1	81.3	20	1.0	4	AD-AD	2	AD	0.728098	...	0.575
3	057_S_0779	1	79.6	28	0.0	0	CN-MCI	0	CN	0.611092	...	0.617
4	033_S_0920	0	80.1	30	0.0	2	CN-CN	0	CN	0.582458	...	0.393

5 rows × 2017 columns

Figure 4.26 – DShipo: Function to see the first 5 lines of the dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 405 entries, 0 to 404
Columns: 2017 entries, ID to lbp-3D-k_ngtdm_Strength
dtypes: float64(2010), int64(4), object(3)
memory usage: 6.2+ MB
```

```
df.describe()
```

	Sex	Age	MMSE	CDR	Transition_Label	Class_Label	original_shape_Elongation	original_shape_Flatness	original_shape_LeastAxisLength	original_shape_MajorAxisLength	...	lbp-3 k_glszm_SmallAreaHighGrayLevelEmpha
count	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	...	405.0000
mean	0.555556	75.415802	27.125926	0.377778	2.323457	0.837037	0.589029	0.320199	0.300380	0.373691	...	0.5561
std	0.497519	6.524493	2.598828	0.313476	1.124246	0.712763	0.123789	0.140308	0.149601	0.164915	...	0.1712
min	0.000000	55.200000	20.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0000
25%	0.000000	71.700000	25.000000	0.000000	1.000000	0.000000	0.513301	0.226485	0.198398	0.253996	...	0.4559
50%	1.000000	75.600000	28.000000	0.500000	2.000000	1.000000	0.591935	0.311260	0.293936	0.358193	...	0.5614
75%	1.000000	79.800000	29.000000	0.500000	3.000000	1.000000	0.670771	0.393472	0.380017	0.476838	...	0.6636
max	1.000000	91.000000	30.000000	1.000000	4.000000	2.000000	1.000000	1.000000	1.000000	1.000000	...	1.0000

8 rows × 2014 columns

Figure 4.27 - DShipo: Description of final data

```
df['Transition_Label'].hist(bins=30,color='darkred',alpha=0.7)
#"CN-MCI":0, "MCI-AD":1, "CN-CN":2, "MCI-MCI":3, "AD-AD":4
```

```
<AxesSubplot:>
```

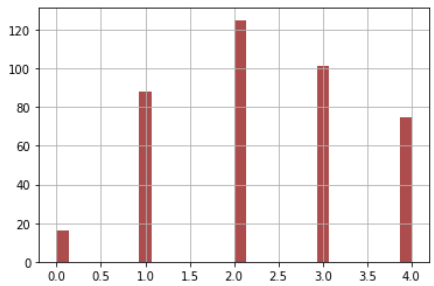


Figure 4.28 – DShipo: Label's distribution

```
seaborn.distplot(df['Age'],kde=False,color='darkred',bins=30)
```

```
<AxesSubplot:xlabel='Age'>
```

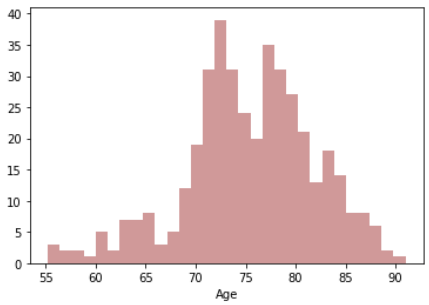


Figure 4.29 – DShipo: Age distribution

After checking all the data, it is then necessary to define the variables to be used. To do that, the columns "ID", "Age", "MMSE", "CDR", "Transition", "Class_Label" and "Class" were removed.

The Y variable (decision variable) was defined as the "*Transition_Label*" column, which contains the numbers previously associated to each transition. Finally, we eliminated this column and then defined the variable X, which is all the remaining dataset (just all the extracted features).

After this, an internal data structure is built that is used by XGBoost, which is optimised for both memory efficiency and training speed - DMatrix.

Being XGBoost the model to be used, it is necessary to define its parameters. As a base, the following parameters were used: *'objective':'multi:softprob'; 'max_depth': 5; 'learning_rate': 0.001;*

'n_estimators':1000; 'gamma':0.2; 'min_child_weight':3; 'subsample':1.0; 'colsample_bytree':1.0; 'reg_lambda':1.0; 'reg_alpha':0.1.

The *cross_val_score()* function will be used to perform the evaluation, taking the dataset and cross-validation configuration and returning a list of scores calculated for each fold. The function run evaluates the model using 10-fold cross-validation. The average classification accuracy is then reported.

In this case, it is seen that the model achieved an estimated classification accuracy of about $0.41 \ +/- \ 0.07$.

To perform model improvement, Random Search was applied, as in the previous case.

Therefore, the new estimated parameters to be used by the model are: *'subsample': 0.8, 'reg_lambda': 0.7, 'reg_alpha': 0.0, 'min_child_weight': 4, 'max_depth': 5, 'learning_rate': 0.005, 'gamma': 0.5 and 'colsample_bytree': 1.0*, and the estimated classification accuracy increased of about $0.46 +/- 0.09$.

The following is a tabular summary of the number of correct and incorrect predictions made by the model, confusion matrix (Figure 4.30).

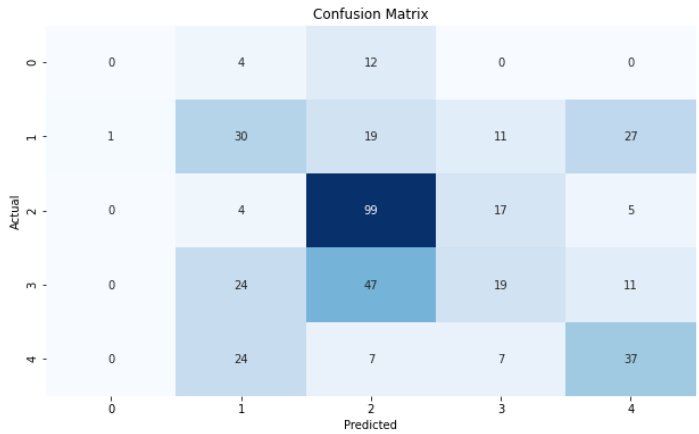


Figure 4.30 - DShipo: Confusion Matrix : 'CN-MCI': 0, 'MCI-AD':1, 'CN-CN':2, 'MCI-MCI': 3, 'AD-AD':4

Based on the results obtained, and as mentioned above, we can see the Feature Importance graph (Figure 4.31). In this particular case, it was decided to choose the 15 most important features.

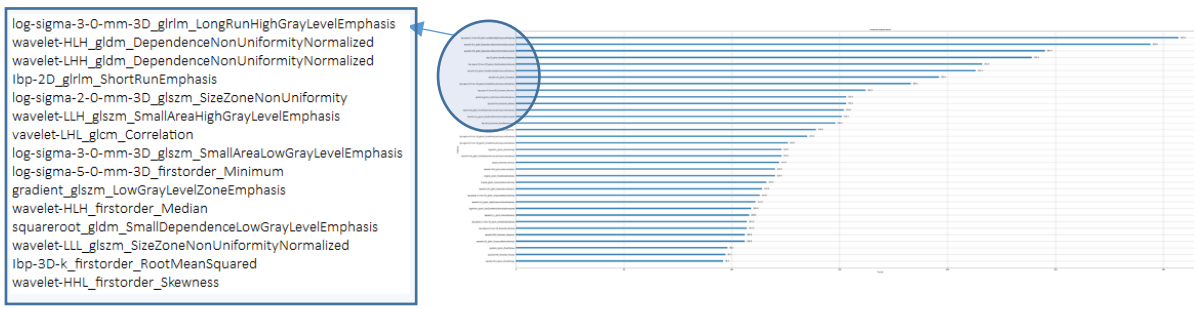


Figure 4.31 – Dshipo: Feature Importance

In addition to this graph, it is also important to present the ShapValues plot (Figure 4.32), which shows which features contributed the most to each class.

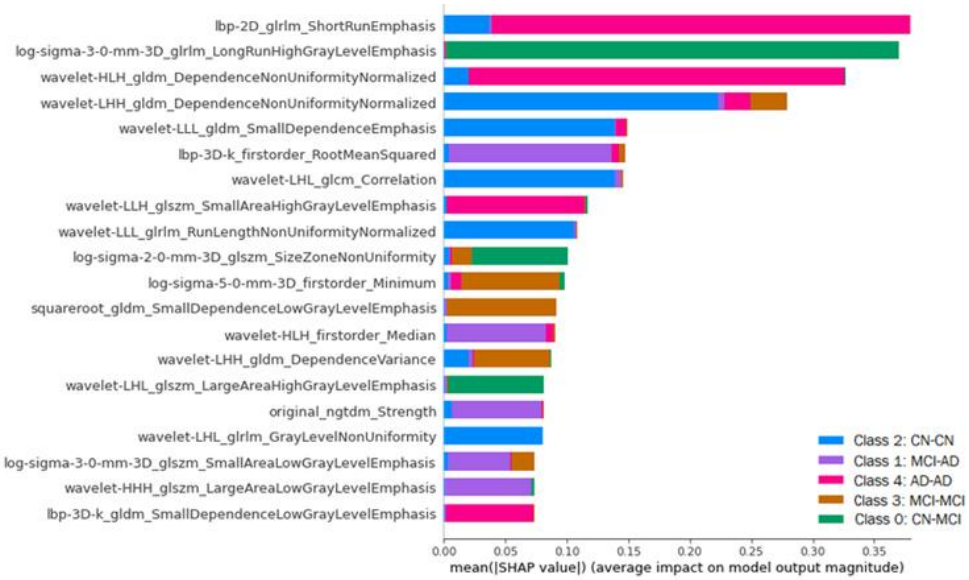


Figure 4.32 - DShipo: ShapValues

Besides presenting the graph of the overall ShapValues, it is also important to represent the ShapValues only for the class on which the study focuses most - class 1: MCI-AD (Figure 4.33). In this representation, it is possible to see not only the important features of the intended class but also if each one of them contributes positively or negatively to the obtained result.

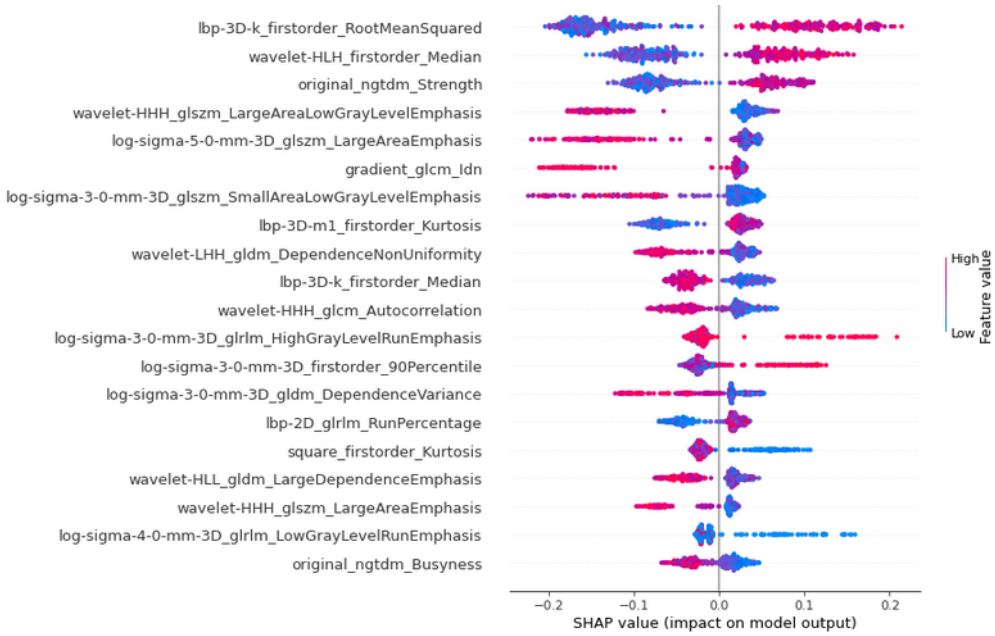


Figure 4.33 – Dshipo: ShapValues: Transition MCI-AD

All these graphs and results obtained will be analysed and commented on in the next subchapter.

4.4.3 DATASET 3: DSENTH

Moving on to the next dataset (Figure 4.34), in this case, the entorhinal, everything will be described similarly.

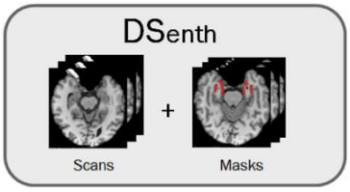


Figure 4.34 – Dsenth: Entorhinal

Before moving on to the model, it is always important to check all the data in our dataset. Therefore, we started by running data analysis functions (Figure 4.35 to Figure 4.39):

```
# Import dataset
CSV_FILE = "C2.DS_Entorh.csv"
df=pd.read_csv(CSV_FILE)
```

Figure 4.35 - Import dataset: DSenth

```
df.head()
```

	ID	Sex	Age	MMSE	CDR	Transition_Label	Transition	Class_Label	Class	original_shape_Elongation	...	lbp-3 k_glszm_SmallAreaHighGrayLevelEmpha
0	006_S_0681	0	77.1	30	0.0	2	CN-CN	0	CN	0.732243	...	0.675
1	941_S_1203	1	83.4	30	0.0	2	CN-CN	0	CN	0.838797	...	0.697
2	011_S_0003	1	81.3	20	1.0	4	AD-AD	2	AD	0.584932	...	0.376
3	057_S_0779	1	79.6	28	0.0	0	CN-MCI	0	CN	0.727133	...	0.366
4	033_S_0920	0	80.1	30	0.0	2	CN-CN	0	CN	0.537634	...	0.531

5 rows × 2030 columns

Figure 4.36 – DSenth: Function to see the first 5 lines of the dataset

```
df['Transition_Label'].hist(bins=30,color='darkred',alpha=0.7)
#"CN-MCI":0, "MCI-AD":1, "CN-CN":2, "MCI-MCI":3, "AD-AD":4
```

```
<AxesSubplot:>
```

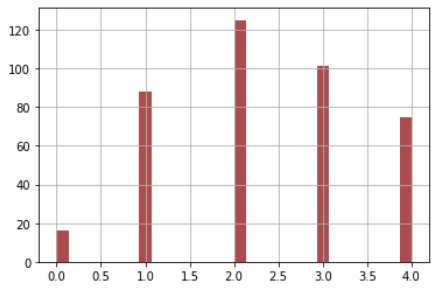


Figure 4.37 – DSenth: Label's distribution

```
seaborn.distplot(df['Age'],kde=False,color='darkred',bins=30)
```

```
<AxesSubplot:xlabel='Age'>
```

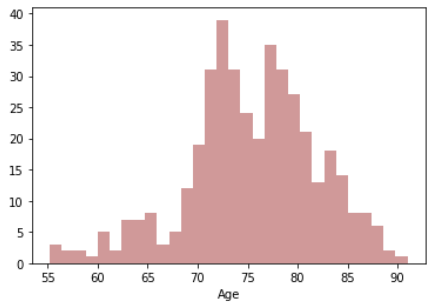


Figure 4.38 – DSenth: Age distribution

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 405 entries, 0 to 404
Columns: 2030 entries, ID to lbp-3D-k_ngtdm_Strength
dtypes: float64(2023), int64(4), object(3)
memory usage: 6.3+ MB
```

```
df.describe()
```

	Sex	Age	MMSE	CDR	Transition_Label	Class_Label	original_shape_Elongation	original_shape_Flatness	original_shape_LeastAxisLength	original_shape_MajorAxisLength	...	lbp-3 k_glszm_SmallAreaHighGrayLevelEmpha
count	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	...	405.0000
mean	0.555556	75.415802	27.125926	0.377778	2.323457	0.837037	0.567270	0.401846	0.370019	0.273345	...	0.5757
std	0.497519	6.524493	2.598828	0.313476	1.124246	0.712763	0.159620	0.172889	0.163179	0.125245	...	0.1600
min	0.000000	55.200000	20.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0000
25%	0.000000	71.700000	25.000000	0.000000	1.000000	0.000000	0.470692	0.285463	0.263713	0.189724	...	0.4778
50%	1.000000	75.600000	28.000000	0.500000	2.000000	1.000000	0.572697	0.400742	0.361201	0.258785	...	0.5771
75%	1.000000	79.800000	29.000000	0.500000	3.000000	1.000000	0.676420	0.510563	0.481697	0.346521	...	0.6640
max	1.000000	91.000000	30.000000	1.000000	4.000000	2.000000	1.000000	1.000000	1.000000	1.000000	...	1.0000

8 rows × 2027 columns

Figure 4.39 - DSenth: Description of final data

After checking all the data, it is then necessary to define the variables to be used. To do that, the columns "ID", "Age", "MMSE", "CDR", "Transition", "Class_Label" and "Class" were removed.

The Y variable (decision variable) was defined as the "*Transition_Label*" column, which contains the numbers previously associated to each transition. Finally, we eliminated this column and then defined the variable X, which is all the remaining dataset (just all the extracted features).

After this, an internal data structure is built that is used by XGBoost, which is optimised for both memory efficiency and training speed - DMatrix.

Being XGBoost the model to be used, it is necessary to define its parameters. As a base, the following parameters were used: *'objective':'multi:softprob'; 'max_depth': 5; 'learning_rate': 0.001; 'n_estimators':1000; 'min_child_weight':3; 'subsample':1.0; 'colsample_bytree':1.0; 'reg_lambda':1.0; 'reg_alpha':0.1*.

The *cross_val_score()* function will be used to perform the evaluation, taking the dataset and cross-validation configuration and returning a list of scores calculated for each fold. The function run evaluates the model using 10-fold cross-validation. The average classification accuracy is then reported.

In this case, it is seen that the model achieved an estimated classification accuracy of about $0.39 \ +/- \ 0.07$.

To perform model improvement, Random Search was applied, as in the previous cases.

Therefore, the new estimated parameters to be used by the model are: *'subsample': 1.0, 'reg_lambda': 0.7, 'reg_alpha': 0.1, 'min_child_weight': 4, 'max_depth': 5, 'learning_rate': 0.005, 'gamma': 0.1 and 'colsample_bytree': 1.0*, and the estimated classification accuracy increased of about $0.39 +/- 0.05$.

The following is a tabular summary of the number of correct and incorrect predictions made by the model, confusion matrix (Figure 4.40).

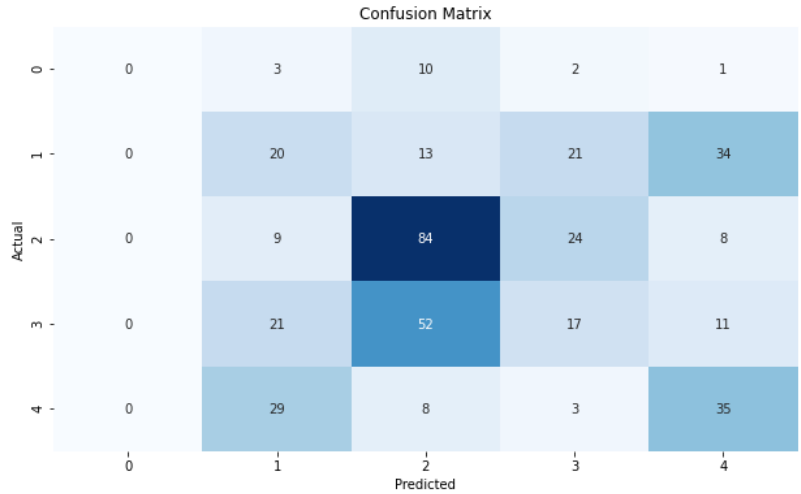


Figure 4.40 - DSenth: Confusion Matrix : 'CN-MCI': 0, 'MCI-AD':1, 'CN-CN':2, 'MCI-MCI': 3, 'AD-AD':4

Based on the results obtained, and as mentioned above, we can see the Feature Importance graph (Figure 4.41). In this particular case, it was decided to choose the 15 most important features.

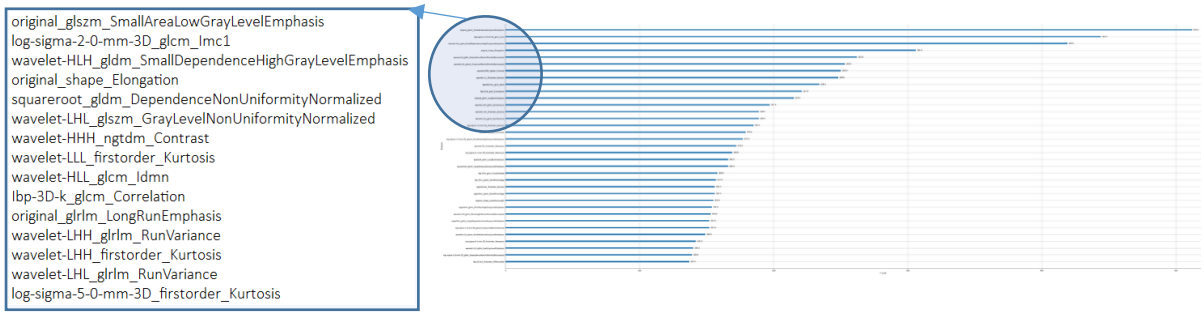


Figure 4.41 – DSenth: Feature Importance

In addition to this graph, it is also important to present the ShapValues plot (Figure 4.42), which shows which features contributed the most to each class.

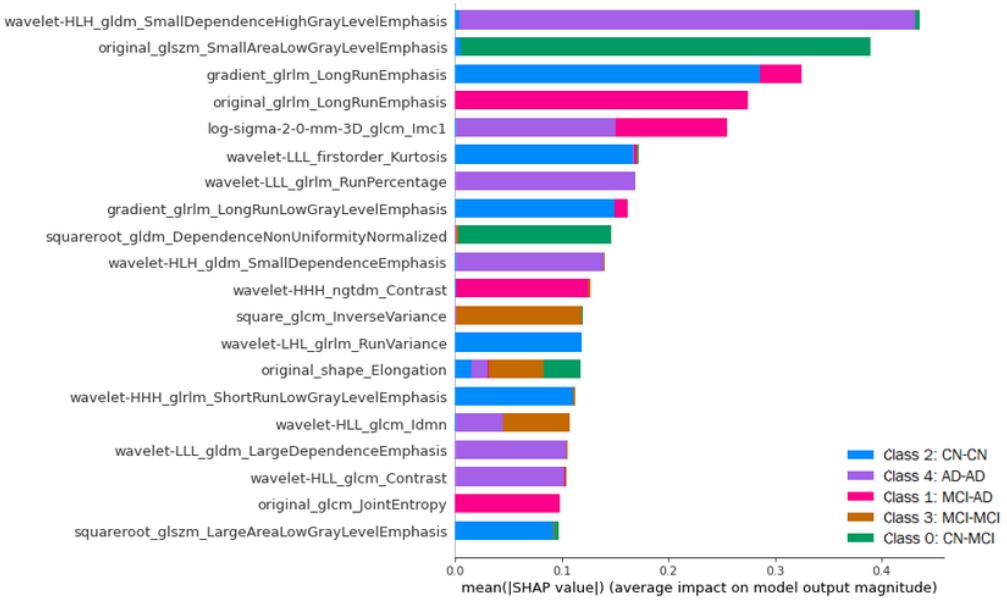


Figure 4.42 - DSenth: ShapValues

Besides presenting the graph of the overall ShapValues, it is also important to represent the ShapValues only for the class on which the study focuses most - class 1: MCI-AD (Figure 4.43). In this representation, it is possible to see not only the important features of the intended class but also if each one of them contributes positively or negatively to the obtained result.

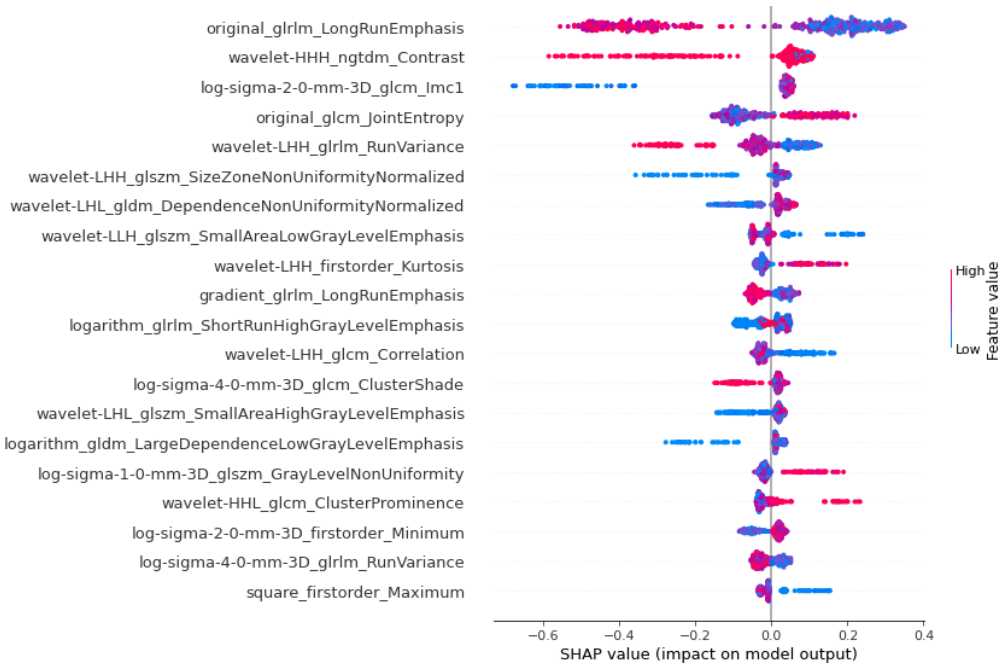


Figure 4.43 – DSenth: ShapValues: Transition MCI-AD

All these graphs and results obtained will be analysed and commented on in the next subchapter.

4.4.4 DATASET 4: DSLOCCIP

Finally, in the last dataset (Figure 4.44), the lateraloccipital, everything will be described in a similar way.

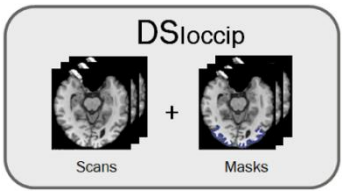


Figure 4.44 – Dsloccip: Lateraloccipital

Before moving on to the model, it is always important to check all the data in our dataset. Therefore, we started by running data analysis functions (Figure 4.45 to Figure 4.49):

```
# Import dataset
CSV_FILE = "D2.DS_Occip.csv"
df=pd.read_csv(CSV_FILE)
```

Figure 4.45 - Import dataset: DSloccip

```
df.head()
```

	ID	Sex	Age	MMSE	CDR	Transition_Label	Transition	Class_Label	Class	original_shape_Elongation	...	lbp-3 k_glszm_SmallAreaHighGrayLevelEmpha
0	006_S_0681	0	77.1	30	0.0	2	CN-CN	0	CN	0.251573	...	0.4081
1	941_S_1203	1	83.4	30	0.0	2	CN-CN	0	CN	0.481976	...	0.547(
2	011_S_0003	1	81.3	20	1.0	4	AD-AD	2	AD	0.548329	...	0.4037
3	057_S_0779	1	79.6	28	0.0	0	CN-MCI	0	CN	0.451810	...	0.6881
4	033_S_0920	0	80.1	30	0.0	2	CN-CN	0	CN	0.279128	...	0.5852

5 rows × 2028 columns

Figure 4.46 – DSloccip: Function to see the first 5 lines of the dataset

```
df['Transition_Label'].hist(bins=30,color='darkred',alpha=0.7)
#"CN-MCI":0, "MCI-AD":1, "CN-CN":2, "MCI-MCI":3, "AD-AD":4
```

```
<AxesSubplot:>
```

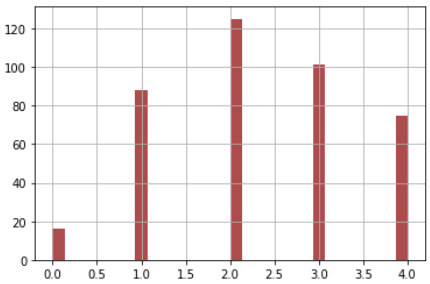


Figure 4.47 – DSloccip: Label's distribution

```
seaborn.distplot(df['Age'],kde=False,color='darkred',bins=30)
```

```
<AxesSubplot:xlabel='Age'>
```

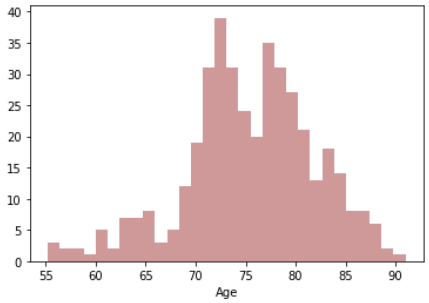


Figure 4.48 – DSloccip: Age distribution

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 405 entries, 0 to 404
Columns: 2028 entries, ID to lbp-3D-k_ngtdm_Strength
dtypes: float64(2021), int64(4), object(3)
memory usage: 6.3+ MB
```

```
df.describe()
```

	Sex	Age	MMSE	CDR	Transition_Label	Class_Label	original_shape_Elongation	original_shape_Flatness	original_shape_LeastAxisLength	original_shape_MajorAxisLength	...	lbp-3D k_glszm_SmallAreaHighGrayLevelEmphas
count	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	405.000000	...	405.0000(
mean	0.555556	75.415802	27.125926	0.377778	2.323457	0.837037	0.389359	0.677440	0.570675	0.273855	...	0.5313(
std	0.497519	6.524493	2.598828	0.313476	1.124246	0.712763	0.145657	0.105930	0.150340	0.097121	...	0.1683(
min	0.000000	55.200000	20.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0000(
25%	0.000000	71.700000	25.000000	0.000000	1.000000	0.000000	0.282039	0.616269	0.465403	0.211189	...	0.4292(
50%	1.000000	75.600000	28.000000	0.500000	2.000000	1.000000	0.380470	0.689762	0.587583	0.267523	...	0.5345(
75%	1.000000	79.800000	29.000000	0.500000	3.000000	1.000000	0.487788	0.748680	0.674378	0.329050	...	0.6488(
max	1.000000	91.000000	30.000000	1.000000	4.000000	2.000000	1.000000	1.000000	1.000000	1.000000	...	1.0000(

8 rows × 2025 columns

Figure 4.49 - DSloccip: Description of final data

After checking all the data, it is then necessary to define the variables to be used. To do that, the columns "ID", "Age", "MMSE", "CDR", "Transition", "Class_Label" and "Class" were removed.

The Y variable (decision variable) was defined as the "*Transition_Label*" column, which contains the numbers previously associated to each transition. Finally, we eliminated this column and then defined the variable X, which is all the remaining dataset (just all the extracted features).

After this, an internal data structure is built that is used by XGBoost, which is optimised for both memory efficiency and training speed - DMatrix.

Being XGBoost the model to be used, it is necessary to define its parameters. As a base, the following parameters were used: *'objective':'multi:softprob'; 'max_depth': 5; 'learning_rate': 0.001;*

'n_estimators':1000; 'min_child_weight':3; 'subsample':1.0; 'colsample_bytree':1.0; 'reg_lambda':1.0; 'reg_alpha':0.1.

The *cross_val_score()* function will be used to perform the evaluation, taking the dataset and cross-validation configuration and returning a list of scores calculated for each fold. The function run evaluates the model using 10-fold cross-validation. The average classification accuracy is then reported.

In this case, it is seen that the model achieved an estimated classification accuracy of about $0.27 \ +/- \ 0.10$.

The following is a tabular summary of the number of correct and incorrect predictions made by the model, confusion matrix (Figure 4.50).

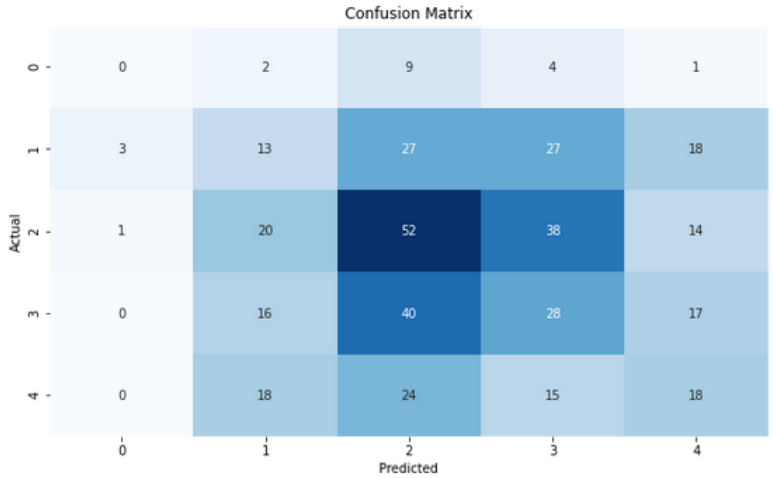


Figure 4.50 - DSloccip: Confusion Matrix : 'CN-MCI': 0, 'MCI-AD':1, 'CN-CN':2, 'MCI-MCI': 3, 'AD-AD':4

Based on the results obtained, and as mentioned above, we can see the Feature Importance graph (Figure 4.51). In this particular case, it was decided to choose the 15 most important features.

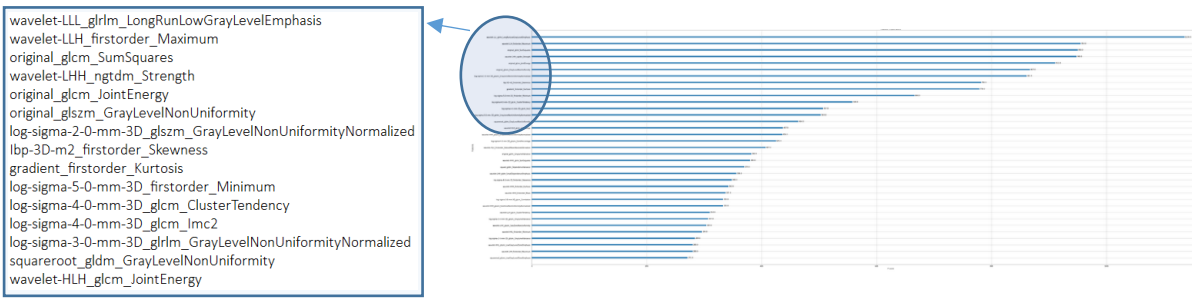


Figure 4.51 – DSloccip: Feature Importance

In addition to this graph, it is also important to present the ShapValues plot (Figure 4.52), which shows which features contributed the most to each class.

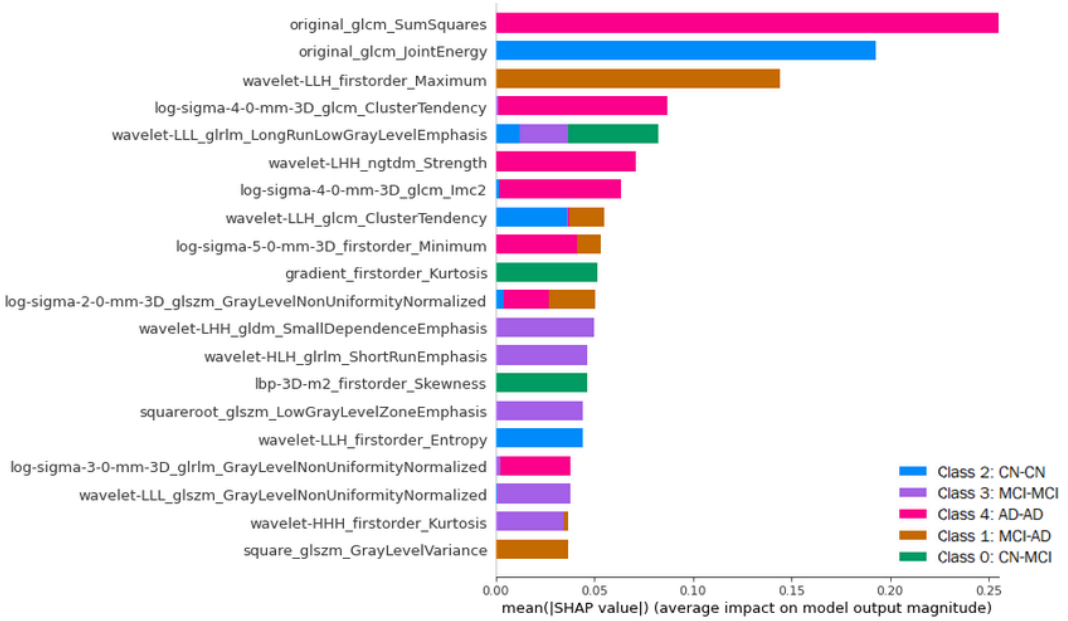


Figure 4.52 - DSloccip: ShapValues

Besides presenting the graph of the overall ShapValues, it is also important to represent the ShapValues only for the class on which the study focuses most - class 1: MCI-AD (Figure 4.53). In this representation, it is possible to see not only the important features of the intended class but also if each one of them contributes positively or negatively to the obtained result.

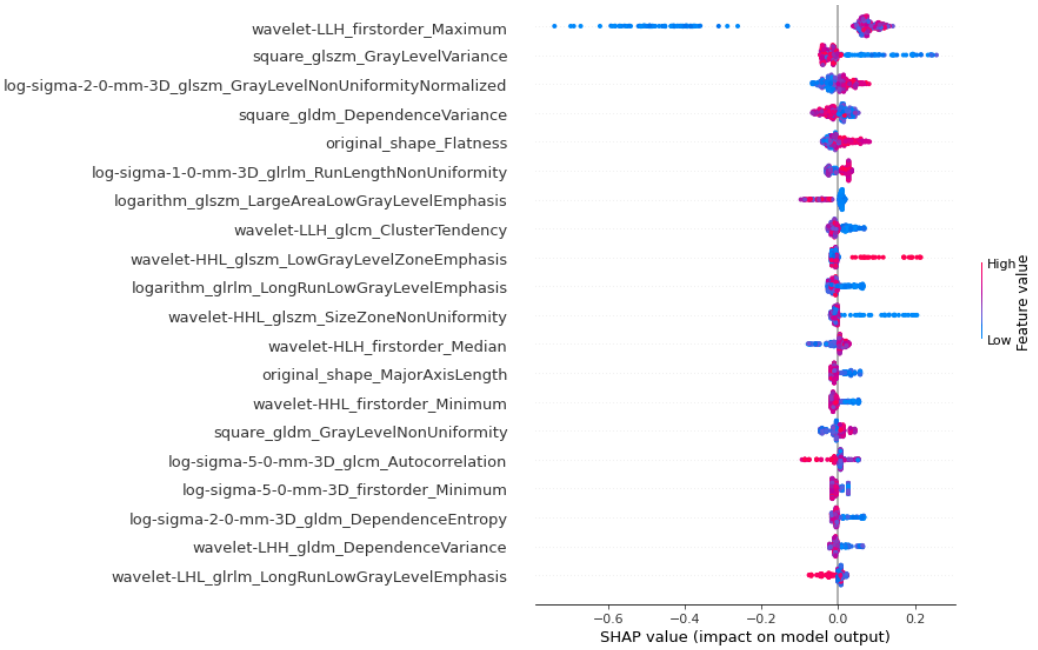


Figure 4.53 – DSloccip: ShapValues: Transition MCI-AD

All these graphs and results obtained will be analysed and commented on in the next subchapter.

All this processing already explained throughout the chapter (in the 4 different cases) is presented in APPENDIX V: Modelling and Model Interpretation.

4.5 RESULTS AND DISCUSSION

Thus, and summarising in Table 4.2 the results obtained in the 4 cases presented above, the following accuracy values are achieved:

Table 4.2 - Summary of the results obtained

Region Test	Whole Brain	Hippocampus	Entorhinal	LateralOccipital
Accuracy	0.31 +/- 0.04	0.46 +/- 0.09	0.39 +/- 0.05	0.27 +/- 0.10

The results obtained will be analysed, by dataset/region [63][117].

4.5.1 DATASET 1: DSBRAIN (WHOLE BRAIN)

Analysing Figure 4.21, the first eight radiomic features (setting a minimum F score of 700) that can be considered candidate predictors for the model are present in Table 4.3.

Table 4.3 - DSbrain: Feature Importance

Filter	Feature	F Score
wavelet-HLH	firstorder_Median	995.0
exponential	firstorder_10Percentile	884.0
wavelet-HLH	firstorder_Mean	840.0
log-sigma-1-0-mm-3D	glcm_InverseVariance	808.0
wavelet-LLL	gldm_SmallDependenceLowGrayLevelEmphasis	806.0
wavelet-HHH	firstorder_Median	775.0
lbp-3D-k	firstorder_Minimum	736.0
square	gldm_DependenceVariance	709.0

In the first case, when the *wavelet* filter is applied, this produces 8 decompositions per level (all possible combinations of applying a High or Low pass filter in each of the three dimensions).

After the filter is applied, *firstorder statistics* describe the distribution of voxel intensities within the image region defined by the mask through commonly used and basic metrics, where:

- X is a set of Np voxels included in the ROI;
- $P(i)$ is the firstorder histogram with Ng discrete intensity levels, where Ng is the number of non-zero bins, equally spaced from 0 with a width defined in the binWidth parameter;

- $p(i)$ is the normalized firstorder histogram and is equal to $\frac{P(i)}{Np}$.

In this first case, the applied feature is *firstorder_Median*, i.e. it calculated the median grey level intensity within the ROI.

In the second case, when the *exponential* filter is applied, it computes the exponential of the original image and the resulting values are rescaled on the range of the initial original image and is defined as:

$$f(x) = e^{cx}, where\ c = \frac{\log(\max(|x|))}{\max(|x|)},$$

$$where\ x\ and\ f(x)\ are\ the\ original\ and\ filtered\ intensity, respectively.$$

After the filter application, the applied feature is *firstorder_10Percentile*, i.e. it calculated the 10th percentile of X.

The third case is similar to the first. The filter applied is the same, just in a different type of decomposition. Figure 4.54 [118] shows a schematic representation of a 3D Wavelet decomposition.

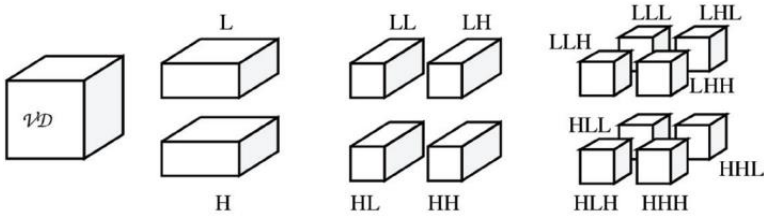


Figure 4.54 - A schematic representation of a 3D Wavelet decomposition. Retrieved from [118]

Once the filter is applied, the *firstorder_Mean* feature is used, i.e.it refers to the average grey level intensity within the ROI.

Moving on to the next case, when the *log-sigma* filter is applied, it works as an edge enhancement filter. Emphasizes areas of grey level change, where sigma (σ) defines how coarse the emphasised texture should be. In short, it applies a Laplacian Gaussian filter to the input image and yields a derived image for each sigma value specified. The Gaussian kernel is used to smooth the image and is defined as:

$$G(x, y, z, \sigma) = \frac{1}{(\sigma\sqrt{2\pi})^3} e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}$$

A low sigma emphasis on fine textures (change over a short distance), where a high sigma value emphasises coarse textures (grey level change over a large distance).

After the filter is applied, *Gray Level Co-occurrence Matrix (GLCM)* of size $Ng \times Ng$ describes the second-order joint probability function of an image region constrained by the mask and is defined as

$P(i,j|\delta,\theta)$. The $(i,j)^{th}$ element of this matrix represents the number of times the combination of levels i and j occur in two pixels in the image, that are separated by a distance of δ pixels along angle θ. The distance δ from the centre voxel is defined as the distance according to the infinity norm. In this particular case, the *glcm_InverseVariance* is defined as:

$$inverse\ variance = \sum_{k=1}^{N_g-1} \frac{p_{x-y}(k)}{k^2},\ p_{x-y}(k) = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j),$$

$where\ |i-j| = k\ (k = 0,1,\dots,N_g-1)\ and\ N_g is\ the\ number\ of\ discret\ intensity\ levels\ in\ the\ image.$

In the 5th case, the wavelet filter applied has already been explained, however, it is in a different decomposition. After the filter is applied, a *Gray Level Dependence Matrix (GLDM)* quantifies grey-level dependencies in an image. A grey level dependency is defined as the number of connected voxels within distance δ that are dependent on the centre voxel. A neighbouring voxel with grey level j is considered dependent on the centre voxel with grey level i if $|i-j| \leq \alpha$. In a grey-level dependence matrix $P(i,j)$ the $(i,j)^{th}$ element describes the number of times a voxel with grey level i with j dependent voxels in its neighbourhood appear in the image. In this case, the *gldm_SmallDependenceLowGrayLevelEmphasis* is defined as:

$$SDLGLE = \frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}\frac{P(i,j)}{i^2 j^2}}{N_z},$$

$where\ N_g is\ the\ number\ of\ discret\ intensity\ levels\ in\ the\ image, N_d\ is\ the\ number\ of\ discret\ dependency\ sizes\ in\ the\ image\ and\ N_z is\ the\ number\ of\ dependency\ zones\ in\ the\ image.$

In the next case, with an FScore of 775, the filter used was a different Wavelet decomposition applied to the same feature already explained above (*firstorder_Median*).

Turning to the second-to-last case, when the *lbp-3D* filter is applied, it computes and returns the Local Binary Pattern (LBP) in 3D using spherical harmonics. LBP is only calculated for voxels segmented in the mask. Once the filter is applied, the *firstorder_Minimum* feature is used, i.e. it calculated the minimum of X.

In the latter case, when the filter square is applied, it takes the square of the image intensities and linearly scales them back to the original range. Negative values in the original image will be made negative again after application of a filter.

After applying the filter, a *Grey Level Dependency Matrix (GLDM)* quantifies the grey-level dependencies in an image, as explained above. In this case, the *gldm_DependenceVariance* measures the variance in dependence size in the image and can be defined as:

$$DV = \sum_{i=1}^{N_g}\sum_{j=1}^{N_d} p(i,j)(j-\mu)^2 \,, where\ \mu = \sum_{i=1}^{N_g}\sum_{j=1}^{N_d} jp(i,j)$$

Considering the ShapValues graph in Figure 4.22, the most important features for each transition (classes 0 and 1), in order of importance, are:

a) CLASS 0 (CN-MCI):

i. log-sigma-1-0-mm-3D_glcm_InverseVariance (already fully explained above)

This information can be interpreted as when *log-sigma-1-0-mm-3D_glcm_InverseVariance* are found in the scan, then indicated that the patient will transit from CN to MCI. It is important to note the most important features for class 2 (CN-CN), as this is where the initial condition has been maintained. Therefore, these features will no longer matter much for the transition.

b) CLASS 1 (MCI-AD) - Figure 4.23:

i. wavelet-HLH_firstorder_Median (already fully explained above)

This information can be interpreted as when *wavelet-HLH_firstorder_Median* is found in the scan, then indicated that the patient will transit from MCI to AD. It is important to note the most important features for class 3 (MCI-MCI), as this is where the initial condition has been maintained. Therefore, these features will no longer matter much for the transition.

ii. square_gldm_LowGrayLevelEmphasis

The filter used has already been explained above. After the filter is applied, a *gldm_LowGrayLevelEmphasis* measures the distribution of low grey-level values, with a higher value indicating a greater concentration of low grey-level values in the image, and can be defined as:

$$LGLE = \frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}\frac{P(i,j)}{i^2}}{N_z}$$

where N_g is the number of discret intensity levels in the image, N_d is the number of discret dependency sizes in the image and N_z is the number of dependency zones in the image.

iii. square_ngtdm_Busyness

The filter used has already been explained above. After the filter is applied, a *Neighbouring Gray Tone Difference Matrix (NGTDM)* quantifies the difference between a grey value and the average grey value of its neighbours within distance δ. The sum of absolute differences for grey level i is stored in the matrix. In this particular case, *ngtdm_Busyness* measures the change from a pixel to its neighbour. A high value for busyness indicates a 'busy' image, with rapid changes of intensity between pixels and its neighbourhood.

iv. lbp-3D-k_firstorder_Minimum (already fully explained above)

v. wavelet-LLH_firstorder_Kurtosis

The filter used has already been explained above. After the filter is applied, the *firstorder_Kurtosis* is a measure of the 'peakedness' of the distribution of values in the image ROI. A higher kurtosis implies that the mass of the distribution is concentrated towards the tail(s) rather than towards the mean. A lower kurtosis implies the reverse: that the mass of the distribution is concentrated towards a spike near the Mean value.

vi. square_gldm_DependenceVariance (already fully explained above)

4.5.2 DATASET 2: DSHIPO (HIPPOCAMPUS)

Analysing Figure 4.31, the first nine radiomic features (setting a minimum F score of 260) that can be considered candidate predictors for the model are present in Table 4.4.

Table 4.4 - DShipo: Feature Importance

Filter	Feature	F Score
log-sigma-3-0-mm-3D	glrlm_LongRunHighGrayLevelEmphasis	307.0
wavelet-HLH	gldm_DependenceNonUniformityNormalized	294.0
wavelet-LHH	gldm_DependenceNonUniformityNormalized	245.0
lbp-2D	glrlm_ShortRunEmphasis	239.0
log-sigma-2-0-mm-3D	glszm_SizeZoneNonUniformity	216.0
wavelet-LLH	glszm_SmallAreaHighGrayLevelEmphasis	213.0
wavelet-LHL	glcm_Correlation	196.0
log-sigma-3-0-mm-3D	glszm_SmallAreaLowGrayLevelEmphasis	183.0
log-sigma-5-0-mm-3D	firstorder_Minimum	162.0

The *Gray Level Run Length Matrix (GLRLM)* quantifies grey level runs, which are defined as the length in number of pixels, of consecutive pixels that have the same grey level value. In a grey-level run length matrix $P(i,j|\theta)$, the $(i,j)^{th}$ element describes the number of runs with grey level i and length j occur in the image (ROI) along angle θ. In a more particular way, the *glrlm_LongRunHighGrayLevelEmphasis* measures the joint distribution of long-run lengths with higher grey-level values and *glrlm_ShortRunEmphasis* is a measure of the distribution of short-run lengths, with a greater value indicative of shorter run lengths and more fine textural textures.

The *gldm_DependenceNonUniformityNormalized* measures the similarity of dependence throughout the image, with a lower value indicating more homogeneity among dependencies in the image.

The *Gray Level Size Zone Matrix (GLSZM)* quantifies grey level zones in an image. A grey-level zone is defined as the number of connected voxels that share the same grey-level intensity. A voxel is considered connected if the distance is 1. In a grey-level size zone matrix $P(i,j)$ the $(i,j)^{th}$ element equals the number of zones with grey level i and size j appear in image. Contrary to GLCM and GLRLM, the GLSZM is rotation independent, with only one matrix calculated for all directions in the ROI.

In more detail, *glszm_SizeZoneNonUniformity* measures the variability of size zone volumes in the image, with a lower value indicating more homogeneity in size zone volume, whereas *glszm_SmallAreaHighGrayLevelEmphasis* and *glszm_SmallAreaLowGrayLevelEmphasis* measures the proportion in the image of the joint distribution of smaller size zones with higher/lower grey-level values.

The *glcm_Correlation* is a value between 0 (uncorrelated) and 1 (perfectly correlated) showing the linear dependency of grey-level values to their respective voxels in the GLCM.

Considering the ShapValues graph in Figure 4.32, the most important features for each transition (classes 0 and 1), in order of importance, are:

a) CLASS 0 (CN-MCI):
   i. log-sigma-3-0-mm-3D_glrlm_LongRunHighGrayLevelEmphasis
   ii. log-sigma-2-0-mm-3D_glszm_SizeZoneNonUniformity
   iii. wavelet-LHL_glszm_LargeAreaHighGrayLevelEmphasis

b) CLASS 1 (MCI-AD) - Figure 4.33:
   i. lbp-3D-k_firstorder_RootMeanSquared

The *firstorder_RootMeanSquared* feature is the square root of the mean of all the squared intensity values. It is another measure of the magnitude of the image values.

ii. wavelet-HLH_firstorder_Median

iii. original_ngtdm_Strength

This feature is applied to the original scan. *A Neighbouring Gray Tone Difference Matrix (NGTDM)* quantifies the difference between a grey value and the average grey value of its neighbours within distance δ. The sum of absolute differences for grey level i is stored in the matrix. In this case, *ngtdm_Strength* is a measure of the primitives in an image. Its value is high when the primitives are easily defined and visible, i.e. an image with a slow change in intensity but more large coarse differences in grey level intensities.

iv. wavelet-HHH_glszm_LargeAreaLowGrayLevelEmphasis

v. log-sigma-5-0-mm-3D_glszm_LargeAreaEmphasis

The *glszm_LargeAreaEmphasis* is a measure of the distribution of large area size zones, with a greater value indicative of larger size zones and more coarse textures.

vi. gradient_glcm_Idn

When the *gradient* filter is applied, this computes and returns the Gradient Magnitude in the image. After the filter is applied, *glcm_Idn* - inverse difference normalized - is a measure of the local homogeneity of an image. It normalizes the difference between the neighbouring intensity values by dividing over the total number of discrete intensity values.

vii. log-sigma-3-0-mm-3D_glszm_SmallAreaLowGrayLevelEmphasis

For example, this feature has less importance for the transition than the previous ones since it not only contributes to this transition but also to class 3 (MCI-MCI), where a patient with MCI does not change his condition.

4.5.3 DATASET 3: DSENTH (ENTORHINAL)

Analysing Figure 4.41, the first six radiomic features (setting a minimum F score of 250) that can be considered candidate predictors for the model are present in Table 4.5.

Table 4.5 - DSenth: Feature Importance

Filter	Feature	F Score
original	glszm_SmallAreaLowGrayLevelEmphasis	512.0
log-sigma-2-0-mm-3D	glcm_Imc1	444.0
wavelet-HLH	gldm_SmallDependenceHighGrayLevelEmphasis	419.0
original	shape_Elongation	306.0
squareroot	gldm_DependenceNonUniformityNormalized	262.0
wavelet-LHL	glszm_GrayLevelNonUniformityNormalized	253.0

The *Shape_Elongation* shows the relationship between the two largest principal components in the ROI shape. For computational reasons, this feature is defined as the inverse of true elongation:

$$elongation = \sqrt{\frac{\lambda_{minor}}{\lambda_{major}}}$$

Here, λ_{major} and λ_{minor} are the lengths of the largest and second-largest principal component axes. The values range between 1 (where the cross-section through the first and second largest principal moments is circle-like (non-elongated)) and 0 (where the object is a maximally elongated: i.e. a 1-dimensional line). The principal component analysis is performed using the physical coordinates of the voxel centres defining the ROI. It, therefore, takes spacing into account but does not make use of the shape mesh.

Considering the ShapValues graph in Figure 4.42, the most important features for each transition (classes 0 and 1), in order of importance, are:

a) CLASS 0 (CN-MCI):
   i. original_glszm_SmallAreaLowGrayLevelEmphasis
   ii. squareroot_gldm_DependenceNonUniformityNormalized

b) CLASS 1 (MCI-AD) - Figure 4.43:
   i. original_glrlm_LongRunEmphasis

A Gray Level Run Length Matrix (GLRLM) quantifies grey level runs, which are defined as the length in number of pixels, of consecutive pixels that have the same grey level value. In a grey-level run length matrix $P(i,j|\theta)$, the $(i,j)^{th}$ element describes the number of runs with grey level i and length j occur in the image (ROI) along angle θ.

In this case, *glrlm_LongRunEmphasis* is a measure of the distribution of long run lengths, with a greater value indicative of longer run lengths and more coarse structural textures.

ii. wavelet-HHH_ngtdm_Contrast

The *ngtdm_Contrast* feature is a measure of the spatial intensity change but is also dependent on the overall grey level dynamic range. Contrast is high when both the dynamic range and the spatial change rate are high, i.e. an image with a large range of grey levels, with large changes between voxels and their neighbourhood.

iii. log-sigma-2-0-mm-3D_glcm_Imc1

iv. original_glcm_JointEntropy

v. wavelet-LHH_glrlm_RunVariance

4.5.4 DATASET 4: DSLOCC (LATERALOCCIPITAL)

Analysing Figure 4.51, the first seven radiomic features (setting a minimum F score of 800) that can be considered candidate predictors for the model are present in Table 4.6.

Table 4.6 - DSlocc: Feature Importance

Filter	Feature	F Score
wavelet-LLL	glrlm_LongRunLowGrayLevelEmphasis	1136.0
wavelet-LLH	firstorder_Maximum	955.0
original	glcm_SumSquares	950.0
wavelet-LHH	ngtdm_Strength	948.0
original	glcm_JointEnergy	911.0
original	glszm_GrayLevelNonUniformity	867.0
log-sigma-2-0-mm-3D	glszm_GrayLevelNonUniformityNormalized	861.0

Considering the ShapValues graph in Figure 4.52, the most important features for each transition (classes 0 and 1), in order of importance, are:

a) CLASS 0 (CN-MCI):

i. gradient_firstorder_Kurtosis

ii. lbp-3D-m2_firstorder_Skewness

The *firstorder_Skewness* feature measures the asymmetry of the distribution of values about the Mean value. Depending on where the tail is elongated and the mass of the distribution is concentrated, this value can be positive or negative.

b) CLASS 1 (MCI-AD) - Figure 4.53:

i. wavelet-LLH_firstorder_Maximum
ii. square_glszm_GrayLevelVariance
iii. log-sigma-2-0-mm-3D_glszm_GrayLevelNonUniformityNormalized
iv. square_gldm_DependenceVariance

5 CONCLUSION

This master thesis used a variety of technologies and frameworks, for which different implementation stages were performed. The following section, section 5.1, presents conclusion notes on the distinct chapters presented, while section 5.2 comprises some ideas for proceeding with this work.

5.1 CONCLUSIONS

This study proposed a pipeline that facilitates the utility of mining predictors for MCI condition based on radiomic features extracted from T1-weighted MRI scans. Most routine clinical scans include T1-weighted images, and the results obtained from this study expand the possibility of the role of machine learning-based classification and prediction techniques in clinical practice.

In addition to wanting to discover the radiomic predictors, to find out how an individual with MCI will evolve over the years, the goal of this dissertation was also to find out which areas of the brain are most affected by this development.

The results obtained were as expected, as stated in [119], where it was found that the areas that are most affected by Alzheimer's are the Hippocampus and the Entorhinal. In this project, it was noted that the accuracy values of the Hippocampus (46%) are higher than the accuracy values of the Entorhinal (39%), but even so, these two are higher than the Lateraloccipital (27%), which was used as a control, as it was previously known that this was not a zone of interest. Besides these 3 regions, it was also important to study the features that contribute to the transitions of interest in a full brain scan (31%).

Thus, one can conclude that the most important zone, also for MCI, is the hippocampus and, in this case, feature "*lbp-3D-k_firstorder_RootMeanSquared"* was the one that contributed the most to the transition from MCI to AD, while, for example, feature "*squareroot_gldm_ SmallDependenceLowGrayLevelEmphasis"* was the one that contributed the most to the MCI stage (MCI-MCI).

The construction of models from radiomics features through machine learning algorithms and their subsequent interpretation allows the mining of predictors. All these features need to be studied and analysed to help in the decision-making, helping medical staff to understand if a "normal" patient will develop MCI or AD, or if a patient who has MCI will evolve into AD in the future.

5.2 PROSPECT FOR FUTURE WORK

For future work, the objective would be to use Black-Box techniques (like Occlusion [120]) applied to the complete scans (whole brain) to obtain the most important areas, i.e., the areas in which the model was based to obtain the classification obtained in order to compare it with the results obtained in this dissertation. Besides this, it would also be important to segment in more regions and besides segmenting the scans of "month 0" (as it was done here), also segment the scans of "month 24" to obtain other essential information like, for example, the difference between the volumes of each segmented region.

Last but not least, it would also be essential to understand how each feature, individually, affects the transition or class to which it belongs.

REFERENCES

[1] J. P. Deshazo, D. L. Lavallie, and F. M. Wolf, "Publication trends in the medical informatics literature: 20 years of 'medical Informatics' in MeSH," *BMC Med. Inform. Decis. Mak.*, vol. 9, no. 1, 2009, doi: 10.1186/1472-6947-9-7.

[2] J. C. Wyatt and J. L. Y. Liu, "Basic concepts in medical informatics," *J. Epidemiol. Community Health*, vol. 56, no. 11, pp. 808–812, Nov. 2002, doi: 10.1136/JECH.56.11.808.

[3] "What is Health Informatics? | Michigan Technological University." https://www.mtu.edu/health-informatics/what-is/ (accessed Dec. 22, 2021).

[4] "Medical Informatics - an overview | ScienceDirect Topics." https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/medical-informatics (accessed Dec. 22, 2021).

[5] R. Haux, "Medical informatics: Past, present, future," *Int. J. Med. Inform.*, vol. 79, no. 9, pp. 599–610, Sep. 2010, doi: 10.1016/j.ijmedinf.2010.06.003.

[6] A. A. T. Bui and R. K. Taira, *Medical Imaging Informatics*. Springer US, 2010.

[7] A. S. Panayides *et al.*, "AI in Medical Imaging Informatics: Current Challenges and Future Directions," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 7. Institute of Electrical and Electronics Engineers Inc., pp. 1837–1857, Jul. 01, 2020, doi: 10.1109/JBHI.2020.2991043.

[8] "Prevalência da Demência | Associação Alzheimer Portugal." https://alzheimerportugal.org/pt/text-0-9-106-349-prevalencia-da-demencia (accessed Dec. 22, 2021).

[9] Q. Feng and Z. Ding, "MRI Radiomics Classification and Prediction in Alzheimer's Disease and Mild Cognitive Impairment: A Review," *Curr. Alzheimer Res.*, vol. 17, no. 3, pp. 297–309, Mar. 2020, doi: 10.2174/1567205017666200303105016.

[10] S. F. Eskildsen, P. Coupé, D. García-Lorenzo, V. Fonov, J. C. Pruessner, and D. L. Collins, "Prediction of Alzheimer's disease in subjects with mild cognitive impairment from the ADNI cohort using patterns of cortical thinning," *Neuroimage*, vol. 65, pp. 511–521, Jan. 2013, doi: 10.1016/j.neuroimage.2012.09.058.

[11] M. Chupin *et al.*, "Fully Automatic Hippocampus Segmentation and Classification in Alzheimer's Disease and Mild Cognitive Impairment Applied on Data from ADNI," *Hippocampus*, vol. 19, no. 6, p. 579, Jun. 2009, doi: 10.1002/HIPO.20626.

[12] E. Westman *et al.*, "AddNeuroMed and ADNI: similar patterns of Alzheimer's atrophy and automated MRI classification accuracy in Europe and North America," *Neuroimage*, vol. 58, no. 3, pp. 818–828, Oct. 2011, doi: 10.1016/J.NEUROIMAGE.2011.06.065.

[13] S. Lovestone *et al.*, "AddNeuroMed–the European collaboration for the discovery of novel biomarkers for

Alzheimer's disease," *Ann. N. Y. Acad. Sci.*, vol. 1180, pp. 36–46, 2009, doi: 10.1111/J.1749-6632.2009.05064.X.

[14] R. Cuingnet *et al.*, "Automatic classification of patients with Alzheimer's disease from structural MRI: a comparison of ten methods using the ADNI database," *Neuroimage*, vol. 56, no. 2, pp. 766–781, May 2011, doi: 10.1016/J.NEUROIMAGE.2010.06.013.

[15] R. Wolz *et al.*, "Multi-Method Analysis of MRI Images in Early Diagnostics of Alzheimer's Disease," *PLoS One*, vol. 6, no. 10, p. e25446, Oct. 2011, doi: 10.1371/JOURNAL.PONE.0025446.

[16] T. Shen, J. Jiang, Y. Li, P. Wu, C. Zuo, and Z. Yan, "Decision Supporting Model for One-year Conversion Probability from MCI to AD using CNN and SVM," *Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Annu. Int. Conf.*, vol. 2018, pp. 738–741, Oct. 2018, doi: 10.1109/EMBC.2018.8512398.

[17] D. Das, J. Ito, T. Kadowaki, and K. Tsuda, "An interpretable machine learning model for diagnosis of Alzheimer's disease," *PeerJ*, vol. 2019, no. 3, p. e6543, Mar. 2019, doi: 10.7717/PEERJ.6543/SUPP-6.

[18] C. Cortes, V. Vapnik, and L. Saitta, "Support-vector networks," *Mach. Learn. 1995 203*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[19] Y. M. Choe, B. C. Lee, I. G. Choi, G. H. Suh, D. Y. Lee, and J. W. Kim, "Mmse subscale scores as useful predictors of ad conversion in mild cognitive impairment," *Neuropsychiatr. Dis. Treat.*, vol. 16, pp. 1767–1775, 2020, doi: 10.2147/NDT.S263702.

[20] W. Lin *et al.*, "Convolutional Neural Networks-Based MRI Image Analysis for the Alzheimer's Disease Prediction From Mild Cognitive Impairment," *Front. Neurosci.*, vol. 12, no. NOV, Nov. 2018, doi: 10.3389/FNINS.2018.00777.

[21] A. Chaddad, C. Desrosiers, and T. Niazi, "Deep radiomic analysis of MRI related to Alzheimer's disease," *IEEE Access*, vol. 6, pp. 58213–58221, 2018, doi: 10.1109/ACCESS.2018.2871977.

[22] K. Hett, V.-T. Ta, J. Manjón, and P. Coupé, "Adaptive fusion of texture-based grading: Application to Alzheimer's disease detection," Sep. 2017, Accessed: Dec. 15, 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01582003.

[23] C. C. Luk *et al.*, "Alzheimer's disease: 3-Dimensional MRI texture for prediction of conversion from mild cognitive impairment," *Alzheimer's Dement. Diagnosis, Assess. Dis. Monit.*, vol. 10, pp. 755–763, Jan. 2018, doi: 10.1016/j.dadm.2018.09.002.

[24] L. Sørensen *et al.*, "Early detection of Alzheimer's disease using MRI hippocampal texture," *Hum. Brain Mapp.*, vol. 37, no. 3, pp. 1148–1161, Mar. 2016, doi: 10.1002/HBM.23091.

[25] A. Martinez-Torteya, J. Rodriguez-Rojas, J. M. Celaya-Padilla, J. I. Galván-Tejada, V. Treviño, and J. Tamez-Peña, "Magnetization-prepared rapid acquisition with gradient echo magnetic resonance imaging signal and texture features for the prediction of mild cognitive impairment to Alzheimer's disease progression," *J. Med. imaging (Bellingham, Wash.)*, vol. 1, no. 3, p. 031005, Sep. 2014, doi: 10.1117/1.JMI.1.3.031005.

[26] C. Wu, S. Guo, Y. Hong, B. Xiao, Y. Wu, and Q. Zhang, "Discrimination and conversion prediction of mild cognitive impairment using convolutional neural networks," *Quant. Imaging Med. Surg.*, vol. 8, no. 10, p. 992, Nov. 2018, doi: 10.21037/QIMS.2018.10.17.

[27] C. DeCarli, "Mild cognitive impairment: prevalence, prognosis, aetiology, and treatment," *Lancet. Neurol.*, vol. 2, no. 1, pp. 15–21, Jan. 2003, doi: 10.1016/S1474-4422(03)00262-X.

[28] "Mild Cognitive Impairment | Lewis Katz School of Medicine at Temple University." https://medicine.temple.edu/departments-centers/research-centers/alzheimers-center-temple/stay-informed/mild-cognitive (accessed Dec. 30, 2021).

[29] "Mild Cognitive Impairment | Memory and Aging Center." https://memory.ucsf.edu/dementia/mild-cognitive-impairment (accessed Dec. 30, 2021).

[30] D. S. Knopman and R. C. Petersen, "Mild cognitive impairment and mild dementia: A clinical perspective," *Mayo Clinic Proceedings*, vol. 89, no. 10. Elsevier Ltd, pp. 1452–1459, Oct. 01, 2014, doi: 10.1016/j.mayocp.2014.06.019.

[31] S. Gauthier *et al.*, "Mild cognitive impairment," *Lancet (London, England)*, vol. 367, no. 9518, pp. 1262–1270, Apr. 2006, doi: 10.1016/S0140-6736(06)68542-5.

[32] "Difference between Mild Cognitive Impairment and Dementia? | NHS." https://www.nestlehealthscience.com/mci/what-is-the-difference-between-mild-cognitive-impairment-and-dementia (accessed Dec. 30, 2021).

[33] M. Janelidze and N. Botchorishvili, "Mild Cognitive Impairment," *Alzheimer's Dis. - 21st Century Chall.*, Jul. 2018, doi: 10.5772/INTECHOPEN.75509.

[34] R. C. Petersen and S. Negash, "Mild cognitive impairment: An overview," *CNS Spectrums*, vol. 13, no. 1. MBL Communications, pp. 45–53, 2008, doi: 10.1017/S1092852900016151.

[35] F. Portet *et al.*, "Mild cognitive impairment (MCI) in medical practice: A critical review of the concept and new diagnostic procedure. Report of the MCI Working Group of the European Consortium on Alzheimer's Disease," *Journal of Neurology, Neurosurgery and Psychiatry*, vol. 77, no. 6. pp. 714–718, Jun. 2006, doi: 10.1136/jnnp.2005.085332.

[36] R. C. Petersen *et al.*, "Mild Cognitive Impairment," 2009.

[37] E. Mariani, R. Monastero, and P. Mecocci, "Mild Cognitive Impairment: A Systematic Review," IOS Press, 2007. doi: 10.3233/jad-2007-12104.

[38] R. C. Petersen, "Mild cognitive impairment as a diagnostic entity," *J. Intern. Med.*, vol. 256, no. 3, pp. 183–194, Sep. 2004, doi: 10.1111/J.1365-2796.2004.01388.X.

[39] R. C. Petersen *et al.*, "Current concepts in mild cognitive impairment," *Arch. Neurol.*, vol. 58, no. 12, pp. 1985–1992, 2001, doi: 10.1001/ARCHNEUR.58.12.1985.

[40] M. B. M. Macedo Montaño and L. R. Ramos, "[Validity of the Portuguese version of Clinical Dementia Rating]," *Rev. Saude Publica*, vol. 39, no. 6, pp. 912–917, Dec. 2005, doi: 10.1590/S0034-

89102005000600007.

[41] "Mini Mental State Examination (MMSE) 1. Orientação (1 ponto por cada resposta correcta)."

[42] R. C. Petersen, "Mild Cognitive Impairment: Transition from Aging to Alzheimer's Disease," *Int. Conf. Alzheimer's Dis. Relat. Disord. (7th 2000 Washington, D.C.)*, 2001.

[43] R. C. Petersen *et al.*, "Alzheimer's Disease Neuroimaging Initiative (ADNI): Clinical characterization," 2010. [Online]. Available: www.neurology.org.

[44] "ADNI | Alzheimer's Disease Neuroimaging Initiative." http://adni.loni.usc.edu/ (accessed Dec. 30, 2021).

[45] "Sistema Nervoso - Toda Matéria." https://www.todamateria.com.br/sistema-nervoso/ (accessed Jan. 04, 2022).

[46] M. E. Raichle and D. A. Gusnard, "Appraising the brain's energy budget," *Proc. Natl. Acad. Sci.*, vol. 99, no. 16, pp. 10237–10239, Aug. 2002, doi: 10.1073/PNAS.172399499.

[47] "Brain Basics: Know Your Brain | National Institute of Neurological Disorders and Stroke." https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Know-Your-Brain (accessed Jan. 04, 2022).

[48] M. Brammer, "The role of neuroimaging in diagnosis and personalized medicine-current position and likely future directions," *Dialogues Clin. Neurosci.*, vol. 11, no. 4, p. 389, 2009, doi: 10.31887/DCNS.2009.11.4/MBRAMMER.

[49] G. Mukhtar and S. Farhan, "Convolutional neural network based prediction of conversion from mild cognitive impairment to alzheimer's disease: A technique using hippocampus extracted from MRI," *Adv. Electr. Comput. Eng.*, vol. 20, no. 2, pp. 113–122, 2020, doi: 10.4316/AECE.2020.02013.

[50] V. M. Aga, "Structural and Functional Imaging," *Psychiatr. Disord. Late Life*, pp. 103–136, 2018, doi: 10.1007/978-3-319-73078-3_15.

[51] "Magnetic Resonance Imaging (MRI)." https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri (accessed Jan. 07, 2022).

[52] "MRI - Mayo Clinic." https://www.mayoclinic.org/tests-procedures/mri/about/pac-20384768 (accessed Jan. 07, 2022).

[53] Dwight G Nishimura, *Principles of magnetic resonance imaging*. 1996.

[54] "quantum spin - Why do some protons align against the magnetic field? - Physics Stack Exchange." https://physics.stackexchange.com/questions/484043/why-do-some-protons-align-against-the-magnetic-field (accessed Jan. 07, 2022).

[55] R. A. Espanha, "Combined MRI with non-image clinical data for brain tumor classification: a CNN/DL approach," 2017.

[56] "Gadolinium | What Is Gadolinium & What Is It Used for in MRIs?" https://www.drugwatch.com/gadolinium/ (accessed Jan. 07, 2022).

[57] "Magnetic resonance imaging - Wikipedia." https://en.wikipedia.org/wiki/Magnetic_resonance_imaging (accessed Jan. 07, 2022).

[58] "Body MRI - magnetic resonance imaging of the chest, abdomen and pelvis." https://www.radiologyinfo.org/en/info/bodymr (accessed Jan. 07, 2022).

[59] "Ressonância Magnética do Crânio (cabeça, cerebral) - Exames de RM." https://www.saudebemestar.pt/pt/exame/imagiologia/ressonancia-magnetica-do-cranio/ (accessed Jan. 07, 2022).

[60] M. E. Mayerhoefer *et al.*, "Introduction to radiomics," *J. Nucl. Med.*, vol. 61, no. 4, pp. 488–495, Apr. 2020, doi: 10.2967/JNUMED.118.222893.

[61] V. Kumar *et al.*, "Radiomics: The process and the challenges," *Magn. Reson. Imaging*, vol. 30, no. 9, pp. 1234–1248, Nov. 2012, doi: 10.1016/j.mri.2012.06.010.

[62] "Radiomic Features — pyradiomics v3.0.1.post11+g03d23f7 documentation." https://pyradiomics.readthedocs.io/en/latest/features.html (accessed Jan. 08, 2022).

[63] "Customizing the Extraction — pyradiomics v3.0.1.post11+g03d23f7 documentation." https://pyradiomics.readthedocs.io/en/latest/customization.html (accessed Jan. 08, 2022).

[64] "Welcome to pyradiomics documentation! — pyradiomics v3.0.1.post11+g03d23f7 documentation." https://pyradiomics.readthedocs.io/en/latest/ (accessed Jan. 08, 2022).

[65] J. Mccarthy, "WHAT IS ARTIFICIAL INTELLIGENCE?," 2004, Accessed: Jan. 11, 2022. [Online]. Available: http://www-formal.stanford.edu/jmc/.

[66] A. Smiti, "When machine learning meets medical world: Current status and future challenges," *Comput. Sci. Rev.*, vol. 37, Aug. 2020, doi: 10.1016/j.cosrev.2020.100280.

[67] M. Yousef Shaheen, "Adoption of machine learning for medical diagnosis Adoption of machine learning for medical diagnosis," 2021, doi: 10.14293/S2199-1006.1.SOR-.PPHMKA6.v1.

[68] I. El Naqa and M. J. Murphy, "What Is Machine Learning?," in *Machine Learning in Radiation Oncology*, 2015, pp. 3–11.

[69] H. Washizaki, H. Uchida, F. Khomh, and Y.-G. Guéhéneuc, "Studying Software Engineering Patterns for Designing Machine Learning Systems," Accessed: Jan. 11, 2022. [Online]. Available: http://www.washi.cs.waseda.ac.jp/iwesep19/.

[70] M. N. Wernick, Y. Yang, J. G. Brankov, G. Yourganov, and S. C. Strother, "© BRAND X PICTURES," doi: 10.1109/MSP.2010.936730.

[71] O. Campesato, *Artificial Intelligence Machine Learning and Deep Learning*. 2020.

[72] S. Suganyadevi, V. Seethalakshmi, and K. Balasamy, "A review on deep learning in medical image analysis," *Int. J. Multimed. Inf. Retr.*, 2021, doi: 10.1007/s13735-021-00218-1.

[73] M. Wernick, Y. Yang, J. Brankov, G. Yourganov, and S. Strother, "Machine learning in medical imaging," in *IEEE Signal Processing Magazine*, 2010, vol. 27, no. 4, pp. 25–38, doi: 10.1109/MSP.2010.936730.

[74] C. V. M. de Brito, "Cloud-based analytics for monitoring and classification of arrhythmias," 2018.

[75] "Types of neurons - Queensland Brain Institute - University of Queensland." https://qbi.uq.edu.au/brain/brain-anatomy/types-neurons (accessed Oct. 28, 2022).

[76] C. Stangor and J. Walinga, "4.1 The Neuron Is the Building Block of the Nervous System." BCcampus, Oct. 17, 2014.

[77] "A Basic Introduction To Neural Networks." http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html (accessed Jan. 12, 2022).

[78] "GitHub - hereismari/mnist-tensorflow: Exploring mnist dataset with TensorFlow and getting 99% accuracy in tests." https://github.com/hereismari/mnist-tensorflow (accessed Jan. 12, 2022).

[79] D. M. Carvalho De Abreu Fontes, "Sistema Web para Optimização do Workflow em Serviço Radiologia Web system for Workflow Optimization in Radiology Service," 2016.

[80] N. Ketkar, *Deep Learning with Python*. Apress, 2017.

[81] "Deep Learning vs Machine Learning or How AI Benefits Business – NIX United." https://nix-united.com/blog/artificial-intelligence-vs-machine-learning-vs-deep-learning-explaining-the-difference/#id-what-is-deep-learning (accessed Jan. 15, 2022).

[82] "Deep Learning made easy with Deep Cognition | by Favio Vázquez | Becoming Human: Artificial Intelligence Magazine." https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351 (accessed Jan. 15, 2022).

[83] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/J.NEUNET.2014.09.003.

[84] I. Arel, D. Rose, and T. Karnowski, "Deep machine learning-A new frontier in artificial intelligence research," *IEEE Comput. Intell. Mag.*, vol. 5, no. 4, pp. 13–18, Nov. 2010, doi: 10.1109/MCI.2010.938364.

[85] J. Patterson and A. Gibson, *Deep Learning A Practitioner's Approach*. 2017.

[86] "Deep Learning in a Nutshell: Core Concepts | NVIDIA Technical Blog." https://developer.nvidia.com/blog/deep-learning-nutshell-core-concepts/ (accessed Oct. 28, 2022).

[87] "Types of Neural Networks Activation Functions." https://www.v7labs.com/blog/neural-networks-activation-functions (accessed Jan. 15, 2022).

[88] J. Wang, H. Zhu, S. H. Wang, and Y. D. Zhang, "A Review of Deep Learning on Medical Image Analysis," *Mob. Networks Appl.*, vol. 26, no. 1, pp. 351–380, Feb. 2021, doi: 10.1007/s11036-020-01672-7.

[89] S. Rafael and M. Pereira, "Automatic Segmentation and Classification of Brain Tumors based on Multisequence MRI Images with Deep Learning Methods," 2018.

[90] "Introduction to Loss Functions." https://algorithmia.com/blog/introduction-to-loss-functions (accessed Jan. 15, 2022).

[91] H. Robbins and S. Monro, *A Stochastic Approximation Method*, vol. 22. 1951.

[92] D. P. Kingma and J. L. Ba, *Adam: A Method for Stochastic Optimization*. International Conference on

Learning Representations, ICLR, 2014.

[93] Y. N. Dauphin, H. De Vries, and Y. Bengio, "RMSProp and equilibrated adaptive learning rates for non-convex optimization," in *Advances in Neural Information Processing Systems*, vol. 2015-January, Neural information processing systems foundation, 2015, pp. 1504–1512.

[94] M. F. Rodrigues, "Brain Semantic Segmentation: A Deep Learning approach in Human and Rat MRI studies," 2018.

[95] "A Practical Guide To Hyperparameter Optimization." https://nanonets.com/blog/hyperparameter-optimization/ (accessed Jan. 15, 2022).

[96] Pranoy Radhakrishnan, "What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network?" https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a (accessed Jan. 15, 2022).

[97] A. Budhiraja, "Dropout in (Deep) Machine learning." https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 (accessed Jan. 15, 2022).

[98] "When to Use MLP, CNN, and RNN Neural Networks." https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/ (accessed Jan. 16, 2022).

[99] "XGBoost Documentation — xgboost 1.6.1 documentation." https://xgboost.readthedocs.io/en/stable/ (accessed Jun. 02, 2022).

[100] "Introduction to XGBoost Algorithm | by Nadeem | Analytics Vidhya | Medium." https://medium.com/analytics-vidhya/introduction-to-xgboost-algorithm-d2e7fad76b04 (accessed Jun. 02, 2022).

[101] "A Gentle Introduction to XGBoost for Applied Machine Learning." https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ (accessed Jun. 02, 2022).

[102] "Using SHAP Values to Explain How Your Machine Learning Model Works | by Vinícius Trevisan | Towards Data Science." https://towardsdatascience.com/using-shap-values-to-explain-how-your-machine-learning-model-works-732b3f40e137 (accessed Oct. 04, 2022).

[103] "Cross-Validation K-fold em Python | by Vilson Rodrigues | Medium." https://vilsonrodrigues.medium.com/cross-validation-k-fold-em-python-53890869b880 (accessed Oct. 28, 2022).

[104] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-Validation," *Encycl. Database Syst.*, pp. 532–538, 2009, doi: 10.1007/978-0-387-39940-9_565.

[105] "What is a Confusion Matrix?" https://www.unite.ai/what-is-a-confusion-matrix/ (accessed Jan. 17, 2022).

[106] G. S. Handelman *et al.*, "Peering into the black box of artificial intelligence: Evaluation metrics of machine learning methods," *American Journal of Roentgenology*, vol. 212, no. 1. American Roentgen Ray Society,

pp. 38–43, Jan. 01, 2019, doi: 10.2214/AJR.18.20224.

[107] "scikit-learn: machine learning in Python — scikit-learn 1.0.2 documentation." https://scikit-learn.org/stable/ (accessed Jan. 18, 2022).

[108] "FreeSurfer." https://surfer.nmr.mgh.harvard.edu/ (accessed Jun. 02, 2022).

[109] "TkMeditGuide/TkMeditWorkingWithData/FreeviewSegmentations - Free Surfer Wiki." https://surfer.nmr.mgh.harvard.edu/fswiki/TkMeditGuide/TkMeditWorkingWithData/FreeviewSegmentations (accessed Jun. 02, 2022).

[110] B. Fischl, "FreeSurfer," *Neuroimage*, vol. 62, no. 2, p. 774, Aug. 2012, doi: 10.1016/J.NEUROIMAGE.2012.01.021.

[111] F. Chollet, *Deep Learning with Python*, Manning. 2017.

[112] "Jupyter Lab: Evolution of the Jupyter Notebook | by Parul Pandey | Towards Data Science." https://towardsdatascience.com/jupyter-lab-evolution-of-the-jupyter-notebook-5297cacde6b (accessed Jun. 18, 2022).

[113] P.Alves, "Mining MRI Predictors for Mild Cognitive Impairment." https://github.com/paulojorge99/Mining-MRI-Predictors-for-Mild-Cognitive-Impairment.

[114] "IDA." https://ida.loni.usc.edu/ (accessed Jul. 03, 2022).

[115] J. Wang, L. He, H. Zheng, and Z. L. Lu, "Optimizing the Magnetization-Prepared Rapid Gradient-Echo (MP-RAGE) Sequence," *PLoS One*, vol. 9, no. 5, p. e96899, May 2014, doi: 10.1371/JOURNAL.PONE.0096899.

[116] "3D Slicer image computing platform | 3D Slicer." https://www.slicer.org/ (accessed Jul. 29, 2022).

[117] "Radiomic Features — pyradiomics documentation." https://pyradiomics.readthedocs.io/en/latest/features.html (accessed Oct. 13, 2022).

[118] M. Pereira and M. Freire, "Biomedical Diagnostics and Clinical Technologies: Applying High-Performance Cluster and Grid Computing Medical inforMation science reference," 2011, doi: 10.4018/978-1-60566-280-0.ch007.

[119] "What Happens to the Brain in Alzheimer's Disease? | National Institute on Aging." https://www.nia.nih.gov/health/what-happens-brain-alzheimers-disease (accessed Oct. 16, 2022).

[120] "Understand Network Predictions Using Occlusion - MATLAB & Simulink." https://www.mathworks.com/help/deeplearning/ug/understand-network-predictions-using-occlusion.html (accessed Jun. 05, 2022).

APPENDICES

APPENDIX I: DATA SELECTION

```python
# Python: function that returns the paths of all existing files

def percorrePasta(pasta):
    files = []
    for f in glob.glob(pasta):
        if(os.path.isdir(f + "/")):
            f = f + "/*"
            files += percorrePasta(f)
        else:
            if(os.path.splitext(f)[1] == ".nii." ):
                files += [f]
    return files
arr = percorrePasta('DS1_scans/*')

print(arr)
```

```python
# Python: Gather in a single file all the information from all the exams

dictImageID = {}

with open('total_total.csv', 'r') as read_obj:
    csv_reader = reader(read_obj)
    for row in csv_reader:
        dictImageID[row[13]] = {"id": row[0], "sex": row[2], "deltames":row[4], "age": row[6], "mmse":row[7], "cdr":row[8], "protocol":row[12]}

idPaciente = 0
lastFound = []

df   =   pd.DataFrame([],columns=['nr_paciente','paciente',   'sexo',   'tecnica','data', 'visita', 'visita_id', 'idade','MMSE', 'Global CDR','protocol' ,'nome'])

for found in arr:
    splitted = found.split('/')
    splitUnderline = splitted[5].split('_')
    imageID = splitUnderline[len(splitUnderline)-1].replace(".nii", "")
    imageID = imageID.replace("I","")
    user = dictImageID[imageID]
    dMes = user["deltames"].split(' ')
    visita_id = 0
    if(dMes[1] == "Month"):
        visita_id = dMes[2]
    if lastFound != [] and lastFound[1] != splitted[1]:
        idPaciente+=1

    df.loc[df.shape[0]] = [idPaciente, splitted[1], user["sex"], splitted[2],splitted[3], user["deltames"], visita_id, user["age"], user["mmse"], user["cdr"], user["protocol"], splitted[5] ]
    lastFound = splitted
```

```python
# Python: based on the file "total_total.csv", which contains the 3602 scans already mentioned, patients with only one scan performed are eliminated and repeated scans are also eliminated.

dictImageID = {}
```

```
with open('total_total.csv', 'r') as read_obj:
    csv_reader = reader(read_obj)
    for row in csv_reader:
        dictImageID[row[13]] = {"id": row[0], "sex": row[2], "deltames":row[4], "age": row[6], "mmse":row[7], "cdr":row[8], "protocol":row[12], "image-id":row[13], "tecnica":row[3],"exames":[]}

def getImageID(splitted):
    splitUnderline = splitted[5].split('_')
    imageID = splitUnderline[len(splitUnderline)-1].replace(".nii", "")
    imageID = imageID.replace("I","")
    return imageID

pacientes = {}
idPaciente = 0

for found in arr:
    splitted = found.split('/')
    imageID = getImageID(splitted)
    user = dictImageID[imageID]
    dMes = user["deltames"].split(' ')
    visita_id = 0
    if(dMes[1] == "Month"):
        visita_id = dMes[2]
    if(user["id"] not in pacientes):
        pacientes[ser["id"]] = {"sex":user["sex"],"tecnica":splitted[2],"protocol":user["protocol"], "exames":[]}
        pacientes[user["id"]]["exames"].append({"data":splitted[3], "dmes": visita_id, "image-id": imageID, "age":user["age"],"mmse":user["mmse"],"cdr":user["cdr"]})
    else:
        pacientes[user["id"]]["exames"].append({"data":splitted[3], "dmes": visita_id, "image-id": imageID, "age":user["age"],"mmse":user["mmse"],"cdr":user["cdr"]})

dgg = pd.DataFrame([],columns=['nr_paciente','paciente', 'sexo', 'tecnica','data', 'visita', 'visita_id', 'idade','MMSE', 'Global CDR','protocol' ,'nome'])

removidos = 0
for key, value in list(pacientes.items()):
    arrDmes = []
    toRemove = []
    for exame in pacientes[key]["exames"]:
        if(exame["dmes"] in arrDmes):
            toRemove.append(exame)
        else:
            arrDmes.append(exame["dmes"])
    for exame in toRemove:
        pacientes[key]["exames"].remove(exame)

    if(len(value["exames"])<=1):
        del pacientes[key]
dgg = pd.DataFrame([],columns=['nr_paciente','paciente', 'sexo', 'tecnica','data', 'visita_id', 'idade','MMSE', 'Global CDR','protocol' ,'nome'])

idPaciente = 0
for key, value in list(pacientes.items()):
    for exame in value["exames"]:
        dgg.loc[dgg.shape[0]] = [idPaciente, key, value["sex"], value["tecnica"], exame["data"], exame["dmes"], exame["age"], exame["mmse"], exame["cdr"], value["protocol"],exame["image-id"]]
    idPaciente+=1

dgg.to_csv('results.csv', index=False)
```

```
# Python: Creation of a file where each line is a patient, and everything else (all information) is in columns.

dictImageID = {}

with open('total_total.csv', 'r') as read_obj:
```

```
6.      csv_reader = reader(read_obj)
7.      for row in csv_reader:
8.          dictImageID[row[13]] = {"id": row[0], "sex": row[2], "deltames":row[4], "age":
    row[6], "mmse":row[7], "cdr":row[8], "protocol":row[12], "image-id":row[13],
    "tecnica":row[3],"exames":[]}
9.
10. def getImageID(splitted):
11.     splitUnderline = splitted[5].split('_')
12.     imageID = splitUnderline[len(splitUnderline)-1].replace(".nii", "")
13.     imageID = imageID.replace("I","")
14.     return imageID
15.
16. pacientes = {}
17.
18. for found in arr:
19.     splitted = found.split('/')
20.     imageID = getImageID(splitted)
21.     user = dictImageID[imageID]
22.     dMes = user["deltames"].split(' ')
23.     visita_id = 0
24.     if(dMes[1] == "Month"):
25.         visita_id = dMes[2]
26.     if(user["id"] not in pacientes):
27.         pacientes[user["id"]] =
    {"sex":user["sex"],"tecnica":splitted[2],"protocol":user["protocol"], "exames":[]}
28.         pacientes[user["id"]]["exames"].append({"data":splitted[3], "dmes": visita_id,
    "image-id": imageID, "age":user["age"],"mmse":user["mmse"],"cdr":user["cdr"]})
29.     else:
30.         pacientes[user["id"]]["exames"].append({"data":splitted[3], "dmes": visita_id,
    "image-id": imageID, "age":user["age"],"mmse":user["mmse"],"cdr":user["cdr"]})
31.
32. dgg = pd.DataFrame([],columns=['nr_paciente','paciente', 'sexo', 'tecnica','data',
    'visita', 'visita_id', 'idade','MMSE', 'Global CDR','protocol' ,'nome'])
33.
34. for key, value in list(pacientes.items()):
35.     arrDmes = []
36.     toRemove = []
37.     for exame in pacientes[key]["exames"]:
38.         if(exame["dmes"] in arrDmes):
39.             toRemove.append(exame)
40.         else:
41.             arrDmes.append(exame["dmes"])
42.     for exame in toRemove:
43.         pacientes[key]["exames"].remove(exame)
44.     if(len(value["exames"])<=1):
45.         del pacientes[key]
46.
47. dff = pd.DataFrame([],columns=['nr_paciente','paciente', 'sexo', 'tecnica', 'protocol',
    'visita_0', 'visita_6','visita_12','visita_18','visita_24','visita_36', 'visita_48',
    'idade_0', 'idade_6','idade_12','idade_18','idade_24','idade_36','idade_48', 'MMSE_0',
    'MMSE_6','MMSE_12','MMSE_18','MMSE_24','MMSE_36','MMSE_48','CDR_0','CDR_6','CDR_12',
    'CDR_18','CDR_24','CDR_36','CDR_48','imagem_0','imagem_6','imagem_12','imagem_18',
    'imagem_24','imagem_36','imagem_48'])
48.
49. idPaciente = 0
50.
51. for key, value in list(pacientes.items()):
52.     visitas = [[0, ' ', ' ', ' ', ' '],[0, ' ', ' ', ' ', ' '],[0, ' ', ' ', ' ', ' '],
    [0, ' ', ' ', ' ', ' '],[0, ' ', ' ', ' ', ' '],[0, ' ', ' ', ' ', ' '],[0, ' ', ' ', '
    ', ' ']]
53.
54.     for exame in value["exames"]:
55.         visita = {"id":1,"idade":exame["age"],"mmse":exame["mmse"], "cdr":exame["cdr"],
    "imagem-id":exame["image-id"]}
56.         if(exame["dmes"] == "6"):
57.             visitas[1] = list(visita.values())
58.         elif(exame["dmes"] == "12"):
59.             visitas[2] = list(visita.values())
60.         elif(exame["dmes"] == "18"):
61.             visitas[3] = list(visita.values())
```

```
62.         elif(exame["dmes"] == "24"):
63.             visitas[4] = list(visita.values())
64.         elif(exame["dmes"] == "36"):
65.             visitas[5] = list(visita.values())
66.         elif(exame["dmes"] == "48"):
67.             visitas[6] = list(visita.values())
68.         else:
69.             visitas[0] = list(visita.values())
70.
71.     transposta = list(map(lambda *i: [j for j in i], *visitas))
72.     transpostaEmLinha = []
73.     for x in transposta:
74.         transpostaEmLinha+=x
75.
76.     dff.loc[dff.shape[0]] = [idPaciente, key, value["sex"], value["tecnica"],
    value["protocol"]] + transpostaEmLinha
77.     idPaciente+=1
78.
79. dff.to_csv('results_final.csv', index=False)
```

APPENDIX II: SCANS SEGMENTATION

```
1.  # Bash: CREATE TABLES WITH ALL THE INFORMATION ABOUT VOLUMES AND AREAS OF EACH REGION
2.
3.  #!/bin/bash
4.  path=`dirname $0`
5.  cd $path
6.  echo "Paulo Alves - STATS2TABLE"
7.
8.  export FREESURFER_HOME=/usr/local/freesurfer
9.  sleep 1
10. source $FREESURFER_HOME/SetUpFreeSurfer.sh
11. sleep 1
12.
13. export SUBJECTS_DIR=$PWD
14. list="`ls -d */`"
15.
16. asegstats2table --subjects $list --meas volume --skip --statsfile wmparc.stats --all-segs
    --tablefile wmparc_stats.csv --delimiter comma
17. asegstats2table --subjects $list --meas volume --skip --tablefile aseg_volume_stats.csv -
    -delimiter comma
18. asegstats2table --subjects $list --meas mean --skip --tablefile aseg_mean_stats.csv --
    delimiter comma
19. aparcstats2table --subjects $list --hemi lh --meas volume --skip --tablefile
    aparc_volume_lh.csv --delimiter comma
20. aparcstats2table --subjects $list --hemi lh --meas thickness --skip --tablefile
    aparc_thickness_lh.csv --delimiter comma
21. aparcstats2table --subjects $list --hemi lh --meas area --skip --tablefile
    aparc_area_lh.csv --delimiter comma
22. aparcstats2table --subjects $list --hemi lh --meas meancurv --skip --tablefile
    aparc_meancurv_lh.csv --delimiter comma
23. aparcstats2table --subjects $list --hemi rh --meas volume --skip --tablefile
    aparc_volume_rh.csv --delimiter comma
24. aparcstats2table --subjects $list --hemi rh --meas thickness --skip --tablefile
    aparc_thickness_rh.csv --delimiter comma
25. aparcstats2table --subjects $list --hemi rh --meas area --skip --tablefile
    aparc_area_rh.csv --delimiter comma
26. aparcstats2table --subjects $list --hemi rh --meas meancurv --skip --tablefile
    aparc_meancurv_rh.csv --delimiter comma
```

```
# Python: Processing the different CSV files created above

colunas = ["Paciente"]
pacientes = {}
soma = 0
listaDeFicheiros  =  ["aseg_volume_stats.csv",  "aseg_mean_stats.csv",  "aparc_volume_rh.csv",
"aparc_volume_lh.csv","aparc_thickness_rh.csv","aparc_thickness_lh.csv","aparc_meancurv_rh.csv
","aparc_meancurv_lh.csv","aparc_area_rh.csv","aparc_area_lh.csv","wmparc_stats.csv"]

id_Paciente = 0
for ficheiro in listaDeFicheiros:
    with open(ficheiro, 'r') as read_obj:
        csv_reader = reader(read_obj)
        for collumn in csv_reader:
            i = 0
            for coluna in collumn:
                if ficheiro == "aseg_volume_stats.csv":
                    coluna = coluna + "_volume"
                if ficheiro == "aseg_mean_stats.csv":
                    coluna = coluna + "_mean"
                if i != 0:
                    colunas.append(coluna)
                i+=1
                soma +=1
            break
        for row in csv_reader:
            nomePaciente = row[0][5:15]
            if nomePaciente in pacientes.keys():
                pacientes[nomePaciente].extend(row[1:])
            else:
                pacientes[nomePaciente] = list(row[1:])
            id_Paciente+=1

df = pd.DataFrame([],columns=colunas)
for paciente in pacientes:
    lista = [paciente]
    lista.extend(pacientes[paciente])

    df.loc[df.shape[0]] = lista

df.to_csv('Final_Freesurfer.csv', index=False)
```

APPENDIX III: RADIOMICS EXTRACTION (PARTICULAR CASE EXAMPLE: HIPPOCAMPUS)

```
PATH_SCANS = "/notebooks/disk2/DS2_FreeSurfer/"

PATH_PARAM = "params.yaml"
print("Parameter file, absolute path:", os.path.abspath(PATH_PARAM))

PATH_CSV_FREESURFER_FILES = 'paths_freesurfer.csv'

PATH_CSV_RADIOMICS_ALL = 'D.all_radiomics_segments.csv'
```

```
# Get the ".nii.gz" paths and get a dictionary with them.

paths_nii = {}
def percorrePasta_2(pasta, barrasIniciais):
    if barrasIniciais == 0:
        barrasIniciais = len(pasta.split("/"))-1
    files = []
```

```
    for f in glob.glob(pasta):
        if(os.path.isdir(f + "/")):
            f = f + "/*"
            files += percorrePasta_2(f, barrasIniciais)
        else:
            if(os.path.splitext(f)[1] == ".gz" ):
                nomeFicheiro = f.split("/")[2+barrasIniciais]
                numeroPaciente = f.split("/")[barrasIniciais][5:15]
                if nomeFicheiro == "aseg.nii.gz" or nomeFicheiro == "brain.nii.gz" :
                    if numeroPaciente not in paths_nii:
                        paths_nii[numeroPaciente] ={"aseg":"","brain":"","diretorio": ""}
                    paths_nii[numeroPaciente][nomeFicheiro.replace(".nii.gz", "")] = f
                    paths_nii[numeroPaciente]["diretorio"] = f.replace(nomeFicheiro, "")
    return paths_nii

percorrePasta_2(PATH_SCANS+'*', 0)
```

Example results:

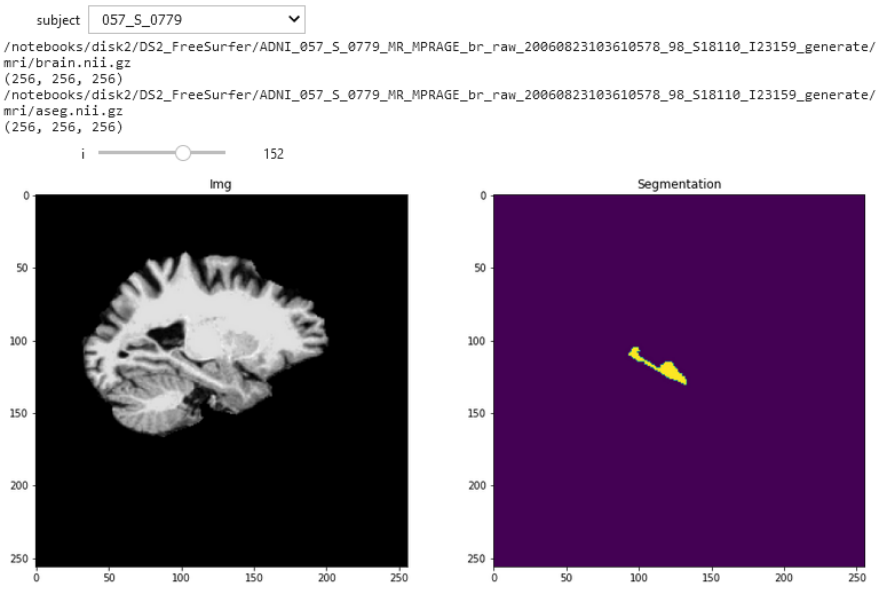
```
print(paths_nii["006_S_0681"])
```

```
{'aseg': '/notebooks/disk2/DS2_FreeSurfer/ADNI_006_S_0681_MR_MP-RAGE__br_raw_20060831143335593_1_S18451_I23677
_generate/mri/aseg.nii.gz', 'brain': '/notebooks/disk2/DS2_FreeSurfer/ADNI_006_S_0681_MR_MP-RAGE__br_raw_20060
831143335593_1_S18451_I23677_generate/mri/brain.nii.gz', 'diretorio': '/notebooks/disk2/DS2_FreeSurfer/ADNI_00
6_S_0681_MR_MP-RAGE__br_raw_20060831143335593_1_S18451_I23677_generate/mri/'}
```

```
# function for selecting the label and display masks

def get_labelnp_from_mgz(labelPath, regionlist):
    nplabel = nib.load(labelPath).get_fdata()
    for reg in regionlist:
        nplabel[nplabel == reg] =-1
    nplabel[nplabel != -1] =0
    nplabel[nplabel == -1] =1
    return nplabel
```

```
# Display the images and masks

list_subject = list(paths_nii.keys())

@interact
def display_img_mask(subject=list_subject):
    print(paths_nii[subject]["brain"])
    img = nib.load(paths_nii[subject]["brain"]).get_fdata()
    print(img.shape)

    print(paths_nii[subject]["aseg"])
    label = get_labelnp_from_mgz(paths_nii[subject]["aseg"], [17,53])
##17 and 53 are the labels corresponding to the hippocampus
    print(label.shape)

    @interact
    def viewer_scan(i=(60,200)):
        plt.figure(figsize=(15,15))
        plt.subplot(2,2,1)
        plt.imshow(img[i,:,:], cmap="gray")
        plt.title("Img")
        plt.subplot(2,2,2)
        plt.imshow(label[i,:,:])
        plt.title("Segmentation")
        plt.show()
```

Results:




```
1.  #save paths in CSV
2.  def create_path_freesurfer_csv(paths_dic, path_csv):
3.      colunas = ["ID", "Image", "Mask"]
4.      df = pd.DataFrame([],columns=colunas)
5.      for paciente in paths_dic:
6.          coluna1 = paths_dic[paciente]["brain"]
7.          coluna2 = paths_dic[paciente]["aseg"]
8.          lista = [paciente, coluna1, coluna2]
9.          df.loc[df.shape[0]] = lista
10.
11.     df.to_csv(path_csv, index=False)
12.
13. create_path_freesurfer_csv(paths_nii,PATH_CSV_FREESURFER_FILES)
```

```
1.  extractor = featureextractor.RadiomicsFeatureExtractor(PATH_PARAM)
2.  print('Extraction parameters:\n\t', extractor.settings)
3.  print('Enabled filters:\n\t', extractor.enabledImagetypes)
4.  print('Enabled features:\n\t', extractor.enabledFeatures)
```

Results:

```
Extraction parameters:
        {'minimumROIDimensions': 2, 'minimumROISize': None, 'normalize': False, 'normalizeScale': 1, 'removeO
utliers': None, 'resampledPixelSpacing': None, 'interpolator': 'sitkBSpline', 'preCrop': False, 'padDistance':
5, 'distances': [1], 'force2D': False, 'force2Ddimension': 0, 'resegmentRange': None, 'label': 1, 'additionalI
nfo': True}
Enabled filters:
        {'Original': {}, 'Wavelet': {}, 'LoG': {'sigma': [1.0, 2.0, 3.0, 4.0, 5.0]}, 'Square': {}, 'SquareRoo
t': {}, 'Logarithm': {}, 'Exponential': {}, 'Gradient': {}, 'LBP2D': {}, 'LBP3D': {}}
Enabled features:
        {'firstorder': [], 'glcm': [], 'gldm': [], 'glrlm': [], 'glszm': [], 'ngtdm': [], 'shape': [], 'shape
2D': []}
```

```
1.  # function that extracts the radiomics features from the hippocampus
2.
3.  def calc_radiomics_files(in_image_file, in_mask_file):
4.      img = nib.load(in_image_file).get_fdata()
5.      mask = get_labelnp_from_mgz(in_mask_file, [17,53])
6.      return extractor.execute(GetImageFromArray(img),
    GetImageFromArray((mask>0).astype(np.uint8)))
```

```
# extract the radiomic features of the hippocampus
%%time

all_df['radiomics'] = all_df.apply(lambda row: calc_radiomics_files(row[ "Image"], row["Mask"]) ,axis = 1)

all_df.head()
```

Results:

```
CPU times: user 5h 14min 22s, sys: 49min 45s, total: 6h 4min 7s
Wall time: 3h 17min 49s
```

	ID	Image	Mask	radiomics
0	006_S_0681	/notebooks/disk2/DS2_FreeSurfer /ADNI_006_S_068...	/notebooks/disk2/DS2_FreeSurfer /ADNI_006_S_068...	{'diagnostics_Versions_PyRadiomics': '2.2.0', ...
1	941_S_1203	/notebooks/disk2/DS2_FreeSurfer /ADNI_941_S_120...	/notebooks/disk2/DS2_FreeSurfer /ADNI_941_S_120...	{'diagnostics_Versions_PyRadiomics': '2.2.0', ...
2	011_S_0003	/notebooks/disk2/DS2_FreeSurfer /ADNI_011_S_000...	/notebooks/disk2/DS2_FreeSurfer /ADNI_011_S_000...	{'diagnostics_Versions_PyRadiomics': '2.2.0', ...
3	057_S_0779	/notebooks/disk2/DS2_FreeSurfer /ADNI_057_S_077...	/notebooks/disk2/DS2_FreeSurfer /ADNI_057_S_077...	{'diagnostics_Versions_PyRadiomics': '2.2.0', ...
4	033_S_0920	/notebooks/disk2/DS2_FreeSurfer /ADNI_033_S_092...	/notebooks/disk2/DS2_FreeSurfer /ADNI_033_S_092...	{'diagnostics_Versions_PyRadiomics': '2.2.0', ...

```
# Create CSV with the results obtained above

def cria_CSV_radiomics_all(PATH_CSV_RADIOMICS_ALL,all_df):
    full_df = pd.DataFrame([dict(**c_row.pop('radiomics'), **c_row) for _, c_row in all_df.iterrows()])
    print(full_df.shape, 'data prepared')
    first_cols = all_df.columns[:-1].tolist()
    full_df = full_df[first_cols + [c_col for c_col in full_df.columns if c_col not in first_cols]]
    # export the whole table
    full_df.to_csv('D.all_radiomics_table.csv', index=False)
    return full_df
    full_df.sample(3)

full_df=cria_CSV_radiomics_all(PATH_CSV_RADIOMICS_ALL, all_df)
full_df.sample(3)
```

Results:

```
(426, 2178) data prepared
```

	ID	Image	Mask	diagnostics_Versions_PyRadiomic
11	116_S_0382	/notebooks/disk2 /DS2_FreeSurfer /ADNI_116_S_038...	/notebooks/disk2 /DS2_FreeSurfer /ADNI_116_S_038...	2.2.
401	133_S_0727	/notebooks/disk2 /DS2_FreeSurfer /ADNI_133_S_072...	/notebooks/disk2 /DS2_FreeSurfer /ADNI_133_S_072...	2.2.
334	116_S_1315	/notebooks/disk2 /DS2_FreeSurfer /ADNI_116_S_131...	/notebooks/disk2 /DS2_FreeSurfer /ADNI_116_S_131...	2.2.

3 rows × 2178 columns

APPENDIX IV: PROCESSING OF THE EXTRACTED RADIOMICS (PARTICULAR CASE EXAMPLE: WHOLE BRAIN)

```
#see all the radiomics features

def feature_analysis(file):
    radiomics_features=pd.read_csv(file)
    print(radiomics_features.shape)
    return radiomics_features

file="A.all_features_scans.csv"
radiomics_features=feature_analysis(file)
```

```
#drop the columns of all features that aren't numeric

def drop_columns(radiomics_features):
    my_type = 'float64'
    dtypes = radiomics_features.dtypes.to_dict()
    for col_nam, typ in dtypes.items():
         if (typ != my_type) and col_nam != "ID":
                #print(col_nam)
                radiomics_features.drop(col_nam, axis = 1,inplace = True)

cleaned_features=drop_columns(radiomics_features)

#get a csv file with all the numeric radiomics features

dtypes = cleaned_features.dtypes.to_dict()
for col_nam, typ in dtypes.items():
        count=0
        num = cleaned_features[col_nam][0]
        for x in range(1,len(cleaned_features[col_nam])):
            if cleaned_features[col_nam][x]!=num:
                count+=1
        if count==0:
            cleaned_features.drop(col_nam, axis = 1,inplace = True)

print(cleaned_features.shape)
cleaned_features.to_csv("A.all_features_scans.csv", index=False)
```

```
#Clinical information append (Age, MMSE, CDR and Sex)

ficheiro = pd.read_csv("A.all_features_scans.csv")
ficheiroUmaLinha = pd.read_csv("0-DS1_metascans.csv")
ages = []
mmse = []
cdr  = []
sex  = []
for index, row in ficheiroUmaLinha.iterrows():
    for colunaDois in ficheiro.columns:
        ages.append(row["idade_0"])
        ages.append(row["idade_24"])
        mmse.append(row["MMSE_0"])
        mmse.append(row["MMSE_24"])
        cdr.append(row["CDR_0"])
        cdr.append(row["CDR_24"])
        if row["sexo"]=="F":
            sex.append(0)
            sex.append(0)
        else:
            sex.append(1)
            sex.append(1)
        break
```

```
ficheiro["AGE"] = ages
ficheiro["MMSE"] = mmse
ficheiro["CDR"] = cdr
ficheiro["SEX"] = sex

print(ficheiro.shape)
ficheiro.to_csv("A1.cleaned_features_scans.csv", index=False)
```

```
#Clasification function

def classificacao(MMSE,CDR):

    if MMSE > 24 and MMSE <=30 and CDR == 0.0:
        return "CN"
    if MMSE > 24 and MMSE <=30 and CDR == 0.5:
        return "MCI"
    if MMSE > 1 and MMSE <=24 and CDR == 0.5:
         return "AD"
    if MMSE > 1 and MMSE <=30 and CDR > 0.5:
        return "AD"
    else:
        return null
```

```
# Use of the above classification function

ficheiro = pd.read_csv("A1.cleaned_features_scans.csv")
pacientes = {}
paciente = ""
estadoAnterior = ""
for index, row in ficheiro.iterrows():
    mmse = row["MMSE"]
    cdr = row["CDR"]
    paciente_atual = row["ID"].split("/")[1][5:15]
    if paciente == paciente_atual:
        est = classificacao(mmse, cdr)
        estadoAnterior = estadoAnterior + "-" + est
        pacientes[paciente_atual] = estadoAnterior
    else:
        estadoAnterior = classificacao(mmse, cdr)
        paciente = paciente_atual
status =[]
for index, row in ficheiro.iterrows():
    paciente_atual = row["ID"].split("/")[1][5:15]
    status.append(pacientes[paciente_atual])

ficheiro["Transition"] = status

print(ficheiro.shape)
ficheiro.to_csv("A1.cleaned_features_scans.csv", index=False)
```

```
# Labeling

ficheiro = pd.read_csv("A1.cleaned_features_scans.csv")
classes = ["CN-MCI", "CN-AD", "MCI-AD", "CN-CN", "MCI-MCI", "AD-AD"]
classes_ids = {"CN-MCI":0, "CN-AD":1, "MCI-AD":2, "CN-CN":3, "MCI-MCI":4, "AD-AD":5}
Transition_ID = []
for index, row in ficheiro.iterrows():
    if row["Transition"] not in classes:
        ficheiro.drop(index, inplace=True)
    else:
        Transition_ID.append(classes_ids[row["Transition"]])

ficheiro["Transition_ID"] = Transition_ID
print(ficheiro.shape)
ficheiro.to_csv("A1.cleaned_features_scans.csv", index=False)
```

```
# Delete the line for month 24 in the whole CSV file

ficheiro = pd.read_csv("A1.cleaned_features_scans.csv")
df = ficheiro.drop_duplicates(subset='ID', keep='first')
print(df.shape)
df.to_csv("A1.cleaned_features_scans.csv", index=False)
```

```
# Create the Class and Class_ID columns

ficheiro = pd.read_csv("A1.cleaned_features_scans.csv")
classe_lista = []
transit = list(ficheiro["Transition"])
for i in transit:
    a=i.split("-")
    classe_lista.append(a[0])

ficheiro["Class"] = classe_lista
print(ficheiro.shape)

classes = ["CN", "MCI","AD"]
classes_ids = {"CN":0, "MCI":1, "AD":2}
Class_ID = []
for index, row in ficheiro.iterrows():
    if row["Class"] not in classes:
        ficheiro.drop(index, inplace=True)
    else:
        Class_ID.append(classes_ids[row["Class"]])

ficheiro["Class_ID"] = Class_ID
print(ficheiro.shape)
ficheiro.to_csv("A1.cleaned_features_scans.csv", index=False)
```

```
# Functions to obtain the number of exams per transition and per class.

def contagens_transicoes(ficheiro):
    contas = {"CN-MCI":0, "CN-AD":0, "MCI-AD":0, "CN-CN":0, "MCI-MCI":0, "AD-AD":0}
    for index, row in ficheiro.iterrows():
        contas[row["Transition"]] += 1
    return contas

def contagens_classes(ficheiro):
    contas = {"CN":0, "MCI":0, "AD":0}
    for index, row in ficheiro.iterrows():
        contas[row["Class"]] += 1
    return contas

contagens_classes(pd.read_csv("A1.cleaned_features_scans.csv"))
contagens_transicoes(pd.read_csv("A1.cleaned_features_scans.csv"))
```

```
#normalize the data

file = "A1.cleaned_features_scans.csv"
csv = pd.read_csv(file)

columns=["ID","SEX","AGE","MMSE","CDR","Transition","Transition_ID","Class","Class_ID"]

def normalize_radiomic():
    ficheiro = pd.read_csv(file)
    data = ficheiro.drop(columns=columns)
    data_normalized = data.apply(lambda x:( (x - x.min()) / (x.max()-x.min())))
    return data_normalized

df=normalize_radiomic()

listaID=csv["ID"]
listaSex=csv["SEX"]
listaAge=csv["AGE"]
```

```
listaMMSE=csv["MMSE"]
listaCDR=csv["CDR"]
listaTransitions=csv["Transition"]
listaTransitionsID=csv["Transition_ID"]
listaClasses=csv["Class"]
listaClassesID=csv["Class_ID"]

df.insert(0, 'ID', listaID)
df.insert(1, 'Sex', listaSex)
df.insert(2, 'Age', listaAge)
df.insert(3, 'MMSE', listaMMSE)
df.insert(4, 'CDR', listaCDR)
df.insert(5, 'Transition_Label', listaTransitionsID)
df.insert(6, 'Transition', listaTransitions)
df.insert(7, 'Class_Label', listaClassesID)
df.insert(8, 'Class', listaClasses)

print(df.shape)
df.to_csv("A2.features_final_scans_normalized.csv", index = False)
```

```
# Since there is only one "CN-AD" case, it will be removed.

ficheiro = pd.read_csv("A2.features_final_scans_normalized.csv")

classes = ["CN-MCI", "MCI-AD", "CN-CN", "MCI-MCI", "AD-AD"]
classes_ids = {"CN-MCI":0, "MCI-AD":1, "CN-CN":2, "MCI-MCI":3, "AD-AD":4}
Transition_ID = []

for index, row in ficheiro.iterrows():
    if row["Transition"] not in classes:
        ficheiro.drop(index, inplace=True)
    else:
        Transition_ID.append(classes_ids[row["Transition"]])

ficheiro["Transition_Label"] = Transition_ID
print(ficheiro.shape)
ficheiro.to_csv("A3.DS_Brain.csv", index=False)
```

```
# Replacement of labels

df=pd.read_csv("A3.DS_Brain.csv")

df["Transition_Label"].replace(2,1, inplace = True)
df["Transition_Label"].replace(3,2, inplace = True)
df["Transition_Label"].replace(4,3, inplace = True)
df["Transition_Label"].replace(5,4, inplace = True)

df["Transition_Label"].value_counts()

df.to_csv("A3.DS_Brain.csv", index=False)
```

APPENDIX V: MODELLING AND MODEL INTERPRETATION

```
# Import from each of the 4 datasets. Each import corresponds to a different notebook,
and in this appendix they will be presented together, whenever they differ.

CSV_FILE = "A3.DS_Brain.csv"
df=pd.read_csv(CSV_FILE)

CSV_FILE = "B2.DS_Hipo.csv"
df=pd.read_csv(CSV_FILE)

```

```
CSV_FILE = "C2.DS_Entorh.csv"
df=pd.read_csv(CSV_FILE)

CSV_FILE = "D2.DS_Occip.csv"
df=pd.read_csv(CSV_FILE)
```

```
#Declare feature vector and target variable (no variation in the 4 notebooks)

dtypes = df.dtypes.to_dict()

for col_nam, typ in dtypes.items():
    if "Unnamed" in col_nam:
        df.drop(col_nam, axis = 1,inplace = True)

df.drop("Age",axis=1,inplace=True)
df.drop("MMSE",axis=1,inplace=True)
df.drop("CDR",axis=1,inplace=True)
df.drop("Transition",axis=1,inplace=True)
df.drop("Class_Label",axis=1,inplace=True)
df.drop("Class",axis=1,inplace=True)


y=df['Transition_Label']

df.drop("ID", axis = 1,inplace = True)
df.drop("Transition_Label", axis = 1,inplace = True)

X=df

data_dmatrix = xgb.DMatrix(data=X,label=y)
```

```
# Define the model in Dataset 1: Whole Brain

from sklearn.model_selection import cross_val_score
params = {
            'objective':'binary:logistic',
            'max_depth': 4,
            'alpha': 0,
            'learning_rate': 0.05,
            'n_estimators':1000,
            'gamma':0.2,
            'min_child_weight':3,
            'lambda':0.9,
            'subsample':0.8,
            'colsample_bytree':0.7,
            'reg_lambda':1.0,
            'reg_alpha':0.0
}
cross_valid_model = xgb.XGBClassifier(**params)
scores = cross_val_score(cross_valid_model, X, np.ravel(y), cv=10)
scores

print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(),
scores.std()))
```

Results: `0.30 accuracy with a standard deviation of 0.05`

```
# Define the model in Dataset 2: Hippocampus

model = xgb.XGBClassifier(colsample_bytree=1.0, gamma=0.2,
                                    learning_rate=0.001,
                                    max_depth=5,
                                    min_child_weight=3,
```

```
                                        n_estimators=1000, n_jobs=12,
                                        objective='multi:softprob',
                                        reg_alpha=0.1, reg_lambda=1.0,
                                        subsample=1.0)

kfold = KFold(n_splits=10, random_state=42, shuffle=True)

scores = cross_val_score(model, X, np.ravel(y), cv=kfold)
scores

print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(),
    scores.std()))
```

Results: `0.41 accuracy with a standard deviation of 0.07`

```
# Define the model in Dataset 3: Entorhinal

model = xgb.XGBClassifier(colsample_bytree=1.0, gamma=0.2,
                                        learning_rate=0.001,
                                        max_depth=5,
                                        min_child_weight=3,
                                        n_estimators=1000, n_jobs=12,
                                        objective='multi:softprob',
                                        reg_alpha=0.1, reg_lambda=1.0,
                                        subsample=1.0)

kfold = KFold(n_splits=10, random_state=42, shuffle=True)

scores = cross_val_score(model, X, np.ravel(y), cv=kfold)
scores

print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(),
    scores.std()))
```

Results: `0.39 accuracy with a standard deviation of 0.07`

```
# Define the model in Dataset 4: Lateral Occipital

model = xgb.XGBClassifier(colsample_bytree=1.0, gamma=0.2,
                                        learning_rate=0.001,
                                        max_depth=5,
                                        min_child_weight=3,
                                        n_estimators=1000, n_jobs=12,
                                        objective='multi:softprob',
                                        reg_alpha=0.1, reg_lambda=1.0,
                                        subsample=1.0)

kfold = KFold(n_splits=10, random_state=42, shuffle=True)

scores = cross_val_score(model, X, np.ravel(y), cv=kfold)
scores

print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(),
    scores.std()))
```

Results: `0.27 accuracy with a standard deviation of 0.10`

```
# Model improvement (in all 4 cases) through RandomSearch

from datetime import datetime
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV, StratifiedKFold

```

```
def timer(start_time=None):
    if not start_time:
        start_time = datetime.now()
        return start_time
    elif start_time:
        thour, temp_sec = divmod((datetime.now() - start_time).total_seconds(), 3600)
        tmin, tsec = divmod(temp_sec, 60)
        print('\n Time taken: %i hours %i minutes and %s seconds.' % (thour, tmin,
round(tsec, 2)))

params = {
        'learning_rate':[0.001, 0.005, 0.01,0.05],
        'max_depth':[3,4,5,6],
        'min_child_weight':[2,3,4],
        'gamma':[0.1,0.2,0.5],
        'subsample':[0.7,0.8,1.0],
        'colsample_bytree':[0.6,0.7,1.0],
        'reg_lambda':[0.7,0.8,1.0],
        'reg_alpha':[0.0,0.1,0.3]
        }

xgb = XGBClassifier(n_estimators=1000, objective='binary:logistic')

folds = 4
param_comb = 8

skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)

random_search = RandomizedSearchCV(xgb, param_distributions=params, n_iter=param_comb,
n_jobs=10, cv=skf.split(X,y), verbose=3, random_state=1001 )

start_time = timer(None) # timing starts from this point for "start_time" variable
random_search.fit(X, y)
timer(start_time) # timing ends here for "start_time" variable

print('\n Best estimator:')
print(random_search.best_estimator_)

print('\n Best hyperparameters:')
print(random_search.best_params_)
```

Example results (Hippocampus):

```
Fitting 4 folds for each of 8 candidates, totalling 32 fits
[Parallel(n_jobs=10)]: Using backend LokyBackend with 10 concurrent workers.
[Parallel(n_jobs=10)]: Done  12 tasks      | elapsed: 87.1min
[Parallel(n_jobs=10)]: Done  24 out of  32 | elapsed: 150.7min remaining: 50.2min

 Time taken: 3 hours 14 minutes and 11.0 seconds.

 Best estimator:
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1.0,
              enable_categorical=False, gamma=0.5, gpu_id=-1,
              importance_type=None, interaction_constraints='',
              learning_rate=0.005, max_delta_step=0, max_depth=5,
              min_child_weight=4, missing=nan, monotone_constraints='()',
              n_estimators=1000, n_jobs=12, num_parallel_tree=1,
              objective='multi:softprob', predictor='auto', random_state=0,
              reg_alpha=0.0, reg_lambda=0.7, scale_pos_weight=None,
              subsample=0.8, tree_method='exact', validate_parameters=1,
              verbosity=None)

 Best hyperparameters:
{'subsample': 0.8, 'reg_lambda': 0.7, 'reg_alpha': 0.0, 'min_child_weight': 4, 'max_depth': 5,
'learning_rate': 0.005, 'gamma': 0.5, 'colsample_bytree': 1.0}
```

```
# Define again with the new parameters the model in Dataset 1: Whole Brain

from sklearn.model_selection import cross_val_score
```

```
params = {
            'objective':'binary:logistic',
            'max_depth': 4,
            'alpha': 0,
            'learning_rate': 0.05,
            'n_estimators':1000,
            'gamma':0.2,
            'min_child_weight':3,
            'lambda':0.9,
            'subsample':0.8,
            'colsample_bytree':0.7,
            'reg_lambda':1.0,
            'reg_alpha':0.0
}
cross_valid_model = xgb.XGBClassifier(**params)
scores = cross_val_score(cross_valid_model, X, np.ravel(y), cv=10)
scores

print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(),
scores.std()))
```

Results: `0.30 accuracy with a standard deviation of 0.05`

```
# Define again with the new parameters the model in Dataset 2: Hippocampus

model = xgb.XGBClassifier(colsample_bytree=1.0, gamma=0.2,
                                    learning_rate=0.001,
                                    max_depth=5,
                                    min_child_weight=3,
                                    n_estimators=1000, n_jobs=12,
                                    objective='multi:softprob',
                                    reg_alpha=0.1, reg_lambda=1.0,
                                    subsample=1.0)

kfold = KFold(n_splits=10, random_state=42, shuffle=True)

scores = cross_val_score(model, X, np.ravel(y), cv=kfold)
scores

print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(),
scores.std()))
```

Results: `0.41 accuracy with a standard deviation of 0.07`

```
# Define again with the new parameters the model in Dataset 3: Entorhinal

model = xgb.XGBClassifier(colsample_bytree=1.0, gamma=0.2,
                                    learning_rate=0.001, max_depth=5,
                                    min_child_weight=3,
                                    n_estimators=1000, n_jobs=12,
                                    objective='multi:softprob',
                                    reg_alpha=0.1, reg_lambda=1.0,
                                    subsample=1.0)

kfold = KFold(n_splits=10, random_state=42, shuffle=True)

scores = cross_val_score(model, X, np.ravel(y), cv=kfold)
scores

print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(),
scores.std()))
```

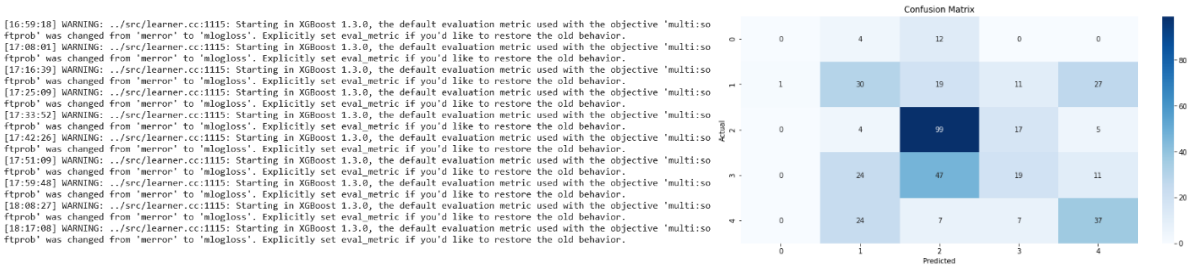
Results: `0.39 accuracy with a standard deviation of 0.05`

```
# Define again with the new parameters the model in Dataset 4: Lateral Occipital

model = xgb.XGBClassifier(colsample_bytree=1.0, gamma=0.2,
                                        learning_rate=0.001,
                                        max_depth=5,
                                        min_child_weight=3,
                                        n_estimators=1000, n_jobs=12,
                                        objective='multi:softprob',
                                        reg_alpha=0.1, reg_lambda=1.0,
                                        subsample=1.0)

kfold = KFold(n_splits=10, random_state=42, shuffle=True)

scores = cross_val_score(model, X, np.ravel(y), cv=kfold)
scores

print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(),
scores.std()))
```

Results: `0.27 accuracy with a standard deviation of 0.10`

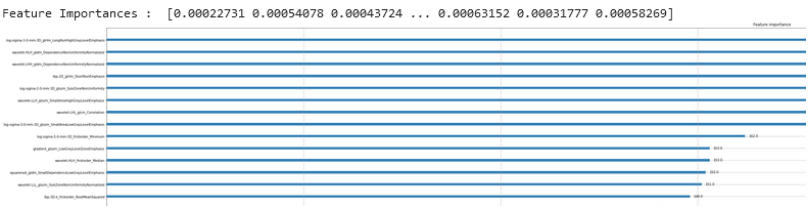
```
# Define "actual_classes, predicted_classes, predicted_proba" and build the confusion
matrix

def cross_val_predict(model, kfold : KFold, X : np.array, y : np.array) ->
Tuple[np.array, np.array, np.array]:
    model_ = cp.deepcopy(model)
    no_classes = len(np.unique(y))

    actual_classes = np.empty([0], dtype=int)
    predicted_classes = np.empty([0], dtype=int)
    predicted_proba = np.empty([0, no_classes])

    for train_ndx, test_ndx in kfold.split(X):
        train_X, train_y, test_X, test_y = X[train_ndx], y[train_ndx], X[test_ndx],
y[test_ndx]
        actual_classes = np.append(actual_classes, test_y)
        model_.fit(train_X, train_y)
        predicted_classes = np.append(predicted_classes, model_.predict(test_X))
        try:
            predicted_proba = np.append(predicted_proba, model_.predict_proba(test_X),
axis=0)
        except:
            predicted_proba = np.append(predicted_proba, np.zeros((len(test_X),
no_classes), dtype=float), axis=0)

    return actual_classes, predicted_classes, predicted_proba

#PLOT MATRIX
def plot_confusion_matrix(actual_classes : np.array, predicted_classes : np.array,
sorted_labels : list):

    matrix = confusion_matrix(actual_classes, predicted_classes, labels=sorted_labels)
    plt.figure(figsize=(12.8,6))
    sns.heatmap(matrix, annot=True, xticklabels=sorted_labels, yticklabels=sorted_labels,
cmap="Blues", fmt="g")
    plt.xlabel('Predicted'); plt.ylabel('Actual'); plt.title('Confusion Matrix')

    plt.show()

actual_classes, predicted_classes, _ = cross_val_predict(model, kfold, X.to_numpy(),
y.to_numpy())
plot_confusion_matrix(actual_classes, predicted_classes, [0, 1, 2, 3, 4])
```

Example results (Hippocampus):




```
1.  #Feature Importance
2.  from matplotlib import pyplot
3.  from xgboost import plot_importance
4.
5.  print("Feature Importances : ", clf_xgb.feature_importances_)
6.
7.  plot_importance(clf_xgb, max_num_features=35)
8.  plt.rcParams["figure.figsize"] = (75,25)
9.  plt.show()
```

Example results (Hippocampus):




```
1.  #Shap Values
2.  clf_xgb = model.fit(X,y)
3.  explainer = shap.Explainer(clf_xgb)
4.  shap_values = explainer.shap_values(X)
5.  shap_obj=explainer(X)
6.  shap.summary_plot(shap_values,X)
7.  //shap.summary_plot(shap_values[1],X) #one class
```

Example results (Hippocampus):