Fernanda Namie Takemoto Furukita

Automatized model for mapping existing bibliography on Amyotrophic Lateral Sclerosis datasets

São Paulo, SP

2024

Fernanda Namie Takemoto Furukita

Automatized model for mapping existing bibliography on Amyotrophic Lateral Sclerosis datasets

Final paper presented to the Department of Computer Engineering and Digital Systems of the Polytechnic School of the University of São Paulo to obtain the degree of Engineer.

University of São Paulo – USP

Polytechnic School

Department of Computer Engineering and Digital Systems (PCS)

Supervisor: Prof. Dr. Edson Satoshi Gomi

São Paulo, SP

2024

This work is dedicated to my family, whose support made me believe I could become an Engineer.

Acknowledgements

Throughout this whole work, I was lucky enough to be assisted by numerous professionals. Their support certainly enriched the knowledge that I was able to obtain from this project. Working in such an interdisciplinary environment has allowed me to develop not only my skills in computer science, but also in biology and medicine.

Firstly, I would like to thank Professor Edson Gomi, Anne-Sophie Rolland and Amaël Broustet, who guided me during the writing of this document as my supervisors and ensured that I had an enriching experience. I extend my gratitude to David Devos, who provided clinical advice, and to Cecile Bordier, our functional MRI specialist whose expertise was crucial at the beginning of my work. Additionally, I am thankful to Dr. Véronique Danel-Brunaud for allowing me to attend consultations with Amyotrophic Lateral Sclerosis patients, giving me direct insight into the challenges posed by the disease.

Finally, I am equally grateful to my family and friends, whose unwavering support has been instrumental since I decided to embark on this journey. I cannot thank you all enough for being such an important part of my life.

Abstract

Amyotrophic Lateral Sclerosis (ALS) is a neurodegenerative disease that affects both upper and lower motor neurons, causing motor and extra-motor symptoms. Currently, there is only one recommended treatment for ALS, which indicates the need for new prognostic biomarkers that could pave the way for novel therapies. Several studies aim to prove the value of technology to develop models capable of predicting the evolution of the disease in each patient. However, in order to apply Machine Learning techniques in these cases, data is necessary. Given the difficulty in finding available and accessible datasets containing different types of data, such as clinical, genetic and imaging, the present work proposes to create a platform centralizing information in academic papers that have used or presented ALS data so information on existing databases and on previous work can be more easily found. To do so, an automatized pipeline is used for analyzing articles found in Google Scholar in the domain by applying Natural Language Processing techniques and, then, the data gathered is made available by an API and displayed in a user interface. As a result, out of 980 articles, 675 were processed, obtaining 57.11%, 53.87% and 53.77% of accuracy for classifying, respectively, databases, data types and amounts of data by the end of the study.

Keywords: Amyotrophic Lateral Sclerosis, Datasets, Natural Language Processing, API, Web Development.

List of Figures

List of Tables

List of abbreviations and acronyms

AI	Artificial Intelligence
ALS	Amyotrophic Lateral Sclerosis
ALSFRS	Amyotrophic Lateral Sclerosis Functional Rating Score
ALSFRS-R	Amyotrophic Lateral Sclerosis Functional Rating Score Revised
ALSOD	Amyotrophic Lateral Sclerosis Database
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BIDS	Brain Imaging Data Structure
BOLD	Blood-Oxygen Level Dependent
CATI	Centre d'Acquisition et de Traitement automatisée de l'Image (Image Acquisition and automated Treatment Center)
CORS	Cross-Origin Resource Sharing
CSF	Cerebrospinal Fluid
DTI	Diffusion Tensor Imaging
ECAS	Edinburgh Cognitive and Behavioural ALS Screen
fMRI	functional Magnetic Resonance Imaging
FVC	Forced Vital Capacity
GM	Gray Matter
HMC	Head-Motion Correction
LLM	Large Language Model
ML	Machine Learning
MRI	Magnetic Resonance Imaging
NLG	Natural Language Generation
NLP	Natural Language Processing

NLU	Natural Language Understanding
PCA	Principal Component Analysis
PD	Persistence Diagram
PI	Persistence Image
PL	Persistence Landscape
PRO-ACT	Pooled Resource Open-Access ALS Clinical Trials
PTM	Pre-Trained Model
ROI	Region of Interest
rsfMRI	resting state functional Magnetic Resonance Imaging
SDC	Susceptibility Distortion Correction
SVM	Support Vector Machine
STC	Slice-Timing Correction
TDA	Topological Data Analysis
TDM	Text and Data Mining
WM	White Matter

Contents

1 Introduction

This project is composed of two main parts. The first was previously completed as my final project at École Centrale de Nantes, and the second is my graduation project at Escola Politécnica da USP. Analyzing the results that were obtained in the first instance of the proposed work, the next steps can be established for the project presented by this document. More details about what has been accomplished prior to this project can be found in Appendix A.

To introduce the subject, Amyotrophic Lateral Sclerosis (ALS), a rare and fatal neurodegenerative disease with no curative treatment, is characterized by motor deficits, but also cognitive disorders that are often overlooked and appear before or after the onset of motor symptoms. The etiology of ALS is unknown, depending on genetic and environmental factors. While the progression of the disease varies considerably from one individual to another, it generally deteriorates rapidly, resulting in a life expectancy generally of 3 to 5 years after diagnosis. Treatments aim to improve quality of life and prolong survival rather than provide a cure, enhancing the importance of identifying biomarkers for ALS and developing predictive models for disease progression.

The research proposed previously involves the analysis of medical images, in particular different magnetic resonance imaging (MRI) modalities, such as T1-weighted MRI, diffusion MRI (DTI) and functional MRI. By using pipelines to process MRI images and apply topological data analysis (TDA) and machine learning (ML) techniques, it was sought to evaluate the efficacy of the topological analysis in refining imaging characterization. TDA involves calculating the persistent homology of connectivity matrices, converting them into persistent images, and then feeding them into a machine learning model. In essence, the aim of such venture was to understand the effects of the TDA by developing a model capable of predicting the progression of ALS over time using information from medical imaging.

To do so, the PULSE cohort was used as source of data. It is a closed French dataset created by the Lille University Hospital including ALS cases and healthy controls from different centers. The study accompanied patients through several sessions, so some of them presented at least two Magnetic Resonance Images separated by a period of time (3, 6 or 12 months apart), which allowed us to have a longitudinal analysis.

Two main pipelines have been explored: the fMRI pipeline and the T1 pipeline. An overall good accuracy (93% in average) was obtained for the Random Forest model in the fMRI pipeline for classifying patients as fast or slow progressors of the disease. However, this result is considered preliminary, given that only 13 patients were available for this particular experiment, which is a very small sample to test the model with. In this case, solely patients that presented images for both M000 (inclusion) and M006 (6 months after inclusion) were included.

The main difficulty during this analysis was the lack of sufficient data to validate results. Even though there were results that were considerably positive and others that were remarkably negative, it was not possible to affirm by the end of the study if the models did truly work or not, since there was such few data available that fit into the requirements established. In addition, no public datasets containing the target MRI modalities in ALS for at least two sessions were found. Therefore, external validation could not be accomplished either.

According to (STEINBACH et al., 2018), in regards to the lack of datasets with a larger number of subjects, better replication and patient characterization, studies in the domain have been limited. This is due, also, to the highlighted heterogeneity associated with ALS, which enhances the need for more comprehensive datasets to characterize all of the disease phenotypes. It arguments in favor of data sharing so a global cohort may be finally created.

In this context, considering the difficulties in finding existing datasets for Amyotrophic Lateral Sclerosis, a new need was detected. A platform containing centralized information on studies that have been conducted using different ALS data would certainly be helpful to understand which datasets are available in the academic field, and which analyses have been made in the past. In such manner, we establish the objective for the next steps of the work hereby presented. From an automatized and scalable model that is capable of analyzing extensive amounts of academic papers related to ALS datasets, information concerning type and amount of data may be collected and displayed in a simple user interface.

To do so, a new pipeline is proposed to achieve such goal, composed of five main steps, shown in Figure 1. It includes collecting articles related to ALS datasets and basic information, such as title, authors, DOI, among others using webscraping techniques; processing its texts; characterizing the datasets from each article with Natural Language Processing (NLP) models; creating an Application Programming Interface (API) and a user interface for making the information available. Chapter 3 describes the methodology adopted to develop the solution.

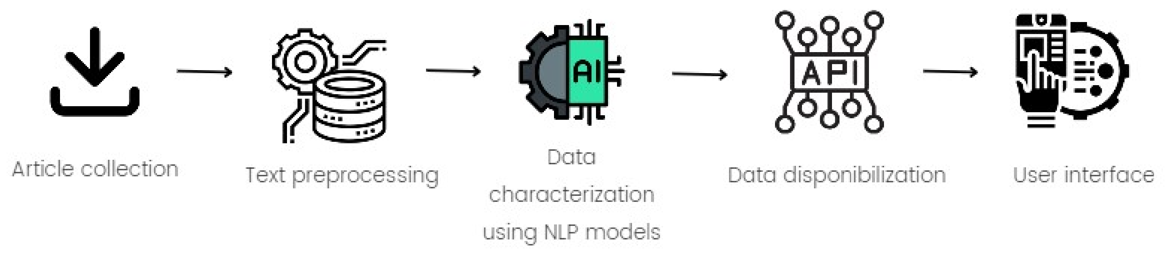


Figure 1 – **Pipeline specifying steps aimed by the graduation project.**

Having implemented the processing above, in order to evaluate the obtained results, the only encountered manner of ensuring the accuracy of data characterization using NLP techniques is by manually checking the given answers. Therefore, the final outcomes presented in Chapter 4 are analyzed in Chapter 5 using as metric the amount of information considered correct after human intervention, i.e. reading each abstract. Finally, Chapter 6 concludes the project and suggests how it can be improved in the future.

2 Conceptual Aspects

2.1 Amyotrophic Lateral Sclerosis

Amyotrophic Lateral Sclerosis, a neurodegenerative disease characterized by its heterogeneity, affects progressively and irreversibly both upper and lower motor neurons, leading to motor and extra-motor symptoms, as shown in Figure 2. The manifestation of ALS varies significantly depending on the site of onset (Figure 3). Patients with spinal-onset ALS typically experience muscle weakness, whereas those with bulbar-onset ALS often present with dysarthria (difficulty with speech) and dysphagia (difficulty swallowing). Additionally, up to 50% of ALS patients develop cognitive and/or behavioral impairments as the disease progresses, which underscores its classification as a neurodegenerative rather than a purely neuromuscular disorder.

The cause of ALS is largely unknown (∼85%), with only a minority of cases explained by genetic mutations (10-15%). The following genes are responsible for up to 70% of familial ALS cases: C9orf72, TAR DNA-binding protein (TDP-43), Cu/Zn superoxide dismutase (SOD1), and Fused in sarcoma (FUS). Environmental and lifestyle factors, such as exposure to cyanobacterial blooms, physical activity, and smoking, have also been investigated as potential contributors to ALS susceptibility (Hardiman et al., 2017 (HARDIMAN et al., 2017)).

ALS remains a rare disorder; however, its incidence increases with age, and its prevalence is expected to grow as global life expectancy rises (Feldman et al., 2022 (FELDMAN et al., 2022)). It is more common among populations of European descent and individuals aged 60 to 79 years.

Diagnosis follows the El Escorial (Brooks et al., 2000 (BROOKS et al., 2000)) and Airlie House criteria, but there is no definitive test for ALS. Clinical investigation is essential to exclude other possible causes and confirm disease progression. For patients with a family history of ALS and presenting symptoms, genetic testing may be included. Conditions that mimic ALS, such as multifocal motor neuropathy, axonal motor predominant chronic inflammatory demyelinating polyneuropathy, spinobulbar muscular atrophy, and inclusion body myositis, often complicate and delay diagnosis.

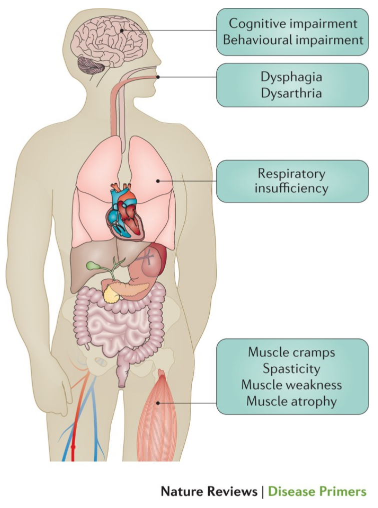


Figure 2 – **Main symptoms of ALS.**

Image extracted from Hardiman et al. (2017) (HARDIMAN et al., 2017).

Regarding treatment, over 50 different drugs have been tested, but only Riluzole is currently recommended and extends survival by approximately three months after 18 months of treatment compared to placebo. It has no significant effect on muscle strength and its mechanism is poorly understood. Common adverse effects of Riluzole include elevated liver enzymes and fatigue, but it is generally considered safe. In addition, symptomatic treatments may be indicated to improve patients' quality of life due to the aggressive progression of ALS.

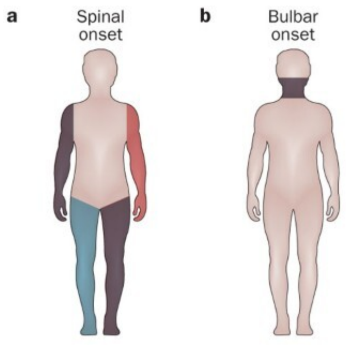


Figure 3 – **ALS onset sites.**

There are two main onset sites: spinal (A) and bulbar (B). *Image extracted from Swinnen and Robberecht (2014)* (SWINNEN; ROBBERECHT, 2014).

Current research is actively seeking diagnostic and prognostic biomarkers for Amyotrophic Lateral Sclerosis (ALS) to enhance the understanding of disease progression and improve treatment strategies. Identifying reliable biomarkers could be instrumental in stratifying patients for clinical trials, aiding the development of new therapies, and assessing their effectiveness. Among the biomarkers discovered, levels of neurofilament light chain (NfL) and phosphorylated neurofilament heavy polypeptide in cerebrospinal fluid (CSF) are noteworthy, as they are currently utilized for differential diagnosis and prognosis of ALS. Additionally, corticospinal tract degeneration observed in MRI scans has shown promise as a diagnostic biomarker. Advancements in structural and functional MRI techniques are expected to uncover new diagnostic biomarkers, with changes in brain connectivity emerging as a potentially valuable tool for ALS biomarker development (Feldman et al., 2022 (FELDMAN et al., 2022)).

As to what concerns prognosis, it is highly variable for Amyotrophic Lateral Sclerosis. The ALS functional rating score (ALSFRS), the most commonly clinical scale measuring the disease progression, is a score starting at 48 and decreasing with the appearance of physical disabilities. Among biochemical markers of prognosis for the disorder are serum urate, serum creatinine, serum chloride and increased serum. In addition, worsening respiratory function indicates short survival. A considerable amount of statistical and ML models have been proposed for early ALS diagnosis and prognosis, predicting the disease's evolution, however even the best ones still retain uncertainty. Some of the developed models are described as follows.

2.1.1 Prognostic Models for Predicting ALS Evolution using Clinical Data

Recent research has explored a wide range of methods, including Machine Learning, Deep Learning, and statistical data analysis, to predict the progression of ALS. These approaches encompass both supervised and unsupervised models, each aiming to improve the accuracy and reliability of ALS progression predictions. For example, in the study by Hothorn and Jung (2014) (HOTHORN; JUNG, 2014), a random forest algorithm was trained using clinical data from the DREAM-Phil Bowen ALS Prediction Prize4Life Challenge, which covered a three-month period. The results, however, were not particularly encouraging.

The most effective predictor for the future ALSFRS slope turned out to be the past ALSFRS slope, which merely confirmed findings from previous studies. The only variable identified as a strong candidate for predicting ALS progression was the disease onset site, underscoring its potential significance in prognosis. Due to these limitations, subsequent studies have proposed new models in the hope of achieving better accuracy and more reliable predictions in ALS prognosis.

Tang et al. (2019) (TANG et al., 2019) propose comparing model-based (linear models) and model-free (machine learning based) techniques for predicting the evolution of the ALSFRS score over time. Clustering was also attempted for grouping patients according to phenotype using unsupervised machine learning methods. The longitudinal patient data was also extracted from the Prize4Life Challenge Data accessed in 2016, which included 8000 patients and 200 clinical features tracked over 12 months at the time. For both cases, model-based and model-free, prediction was only moderately successful.

Correlation between observed changes in ALSFRS scores and the predictions were around 0.427 and 0.545. In relation to the clustering, reliable and consistent partitions were observed, however the study concludes that new biomarkers are still necessary aiming to improve such progression predictions using machine learning.

On the other hand, the algorithm developed by Elamin et al. (2015) (ELAMIN et al., 2015) was able to reliably predict prognosis in ALS patients by determining a prognostic index to separate the patients in three risk groups (low, medium or high risk). To identify which clinical data should be considered in the computation of the index, Kaplan-Meier methods and Cox proportional hazards regression were implemented. Data corresponded to an Irish cohort, which was used in training applying cross validation, and an Italian cohort for external confirmation.

Additionally, a recent systematic review (Tavazzi et al., 2023 (TAVAZZI et al., 2023)) analyzed 15 studies on patient stratification, 28 on prediction of ALS progression and 6 on both stratification and prediction based on artificial intelligence and statistical methods. The results show that the ALSFRS and ALSFRS-R scores were the main prediction targets, so it was certain that we would test our models for predicting disease progression using the ALSFRS score. Also, among the most often found methods were K-means, hierarchical and expectation-maximisation clustering for stratification and random forests, logistic regression, the Cox proportional hazard model and deep learning techniques for prediction. Nonetheless, deep learning has not yet been established as superior to traditional methods, even though it could have potential for prediction applications. Therefore, it led us to believe that it is worthy testing supervised and unsupervised methods, but not deep learning in this case.

2.1.2 Prognostic and Diagnostic Models using MRI Data

Other studies have been proposed to explore the usage of MRI Data as the input of model training methods either along with clinical or the imaging alone. Two examples that used clinical and MRI data combined were presented by Schuster et al. (2017) (SCHUSTER; HARDIMAN; BEDE, 2017) and Van Der Burgh et al. (2017) (van der Burgh et al., 2017). They all found better performance of their models when using both clinical and imaging datasets than separately (Table 1).

For Schuster et al. (2017) (SCHUSTER; HARDIMAN; BEDE, 2017), MRI data corresponded to features calculated from surface-based morphometry and diffusion tensor white matter parameters. It included precentral/paracentral gyri for cortical thickness and superior/inferior corona radiata, anterior/posterior limbs of the internal capsule, cerebral peduncles and genu/body/splenium of the corpus callosum for white matter. As for Van Der Burgh et al. (2017) (van der Burgh et al., 2017), T1-weighted and diffusion-weighted images were used to determine structural connectivity matrices and brain morphology data.

Study	Accuracy clinical data	Accuracy MRI data	Accuracy both data
Survival prediction (mortality within 18-months) (SCHUSTER; HARDIMAN; BEDE, 2017)	66.67%	77.08%	79.17%
Survival category (short, medium or long) (van der Burgh et al., 2017)	68.8%	62.5%	84.4%

Table 1 – **Survival prediction accuracy using clinical and imaging dataset.**

Schuster et al. (2017) (SCHUSTER; HARDIMAN; BEDE, 2017) and Van Der Burgh et al. (2017) (van der Burgh et al., 2017) obtained similar accuracy percentage, and best when clinical and MRI data were combined.

In a diagnostic context, rather than prognostic, the study by Kushol et al. (2023) (KUSHOL et al., 2023) introduces a Vision Transformer (ViT) architecture to differentiate ALS patients from healthy controls. The study utilizes a combination of T1-weighted, R2*, and FLAIR images sourced from multi-center datasets. The results of this approach are overall positive, as the framework presents better accuracy (88%) than other deep learning-based techniques.

This high level of accuracy, when combined with the prognostic outcomes derived from MRI and clinical data, underscores the potential of magnetic resonance imaging as a valuable biomarker for ALS. These findings support the aim of the present project, reinforcing the idea that MRI could play a critical role in both diagnosing ALS and understanding its progression.

2.2 ALS Databases

Research on ALS has resulted in the conception of different databases related to the disease. However, very few datasets are publicly available and only under approval of user request. In addition, most accessible published data only include genetic and/or clinical features, but no imaging. This is due to the fact that Amyotrophic Lateral Sclerosis is a disease classified as rare, representing around 3 cases per 100,000 individuals in European populations, where it is the most incident (HARDIMAN et al., 2017). Other than that, given motor symptoms' rapid aggravation in ALS patients, it is difficult to demand dislocation so imaging exams may be accomplished and also complicated to obtain exploitable images when quality depends on the minimization of movement during the data collection.

Among known ALS datasets, there is the Amyotrophic Lateral Scleroris Database (ALSOD), the Pooled Resource Open-Access ALS Clinical Trials (PRO-ACT) database and Answer ALS Data Portal. ALSOD was first published in 1999 and later updated in 2008. It stores the mutation points in the SOD1 amino acid sequence that characterize individuals affected by Amyotrophic Lateral Sclerosis. New features that were added more recently, such as new identified mutations on SOD1 and supplying more detail to patient information, have helped explaining ALS phenotypes. Until 2008, there were 50 reported users and data from 97 patients from Asia, Europe and North America (WROE et al., 2008).

PRO-ACT is composed by clinical trials data regarding over than 8,600 subjects. It includes demographics, family histories and longitudinal clinical and laboratory data, which encompasses the ALSFRS-R score and Vital Capacity. By 2014, it was the largest public dataset concerning clinical trials data on ALS, aiming to provide enough statistical power for new researches on the disease. (ATASSI et al., 2014)

Answer ALS Data Portal is still in the process of data collecting and its goal consists in becoming the widest clinical and omics dataset concerning ALS to help in the global development of new treatments and the search for a cure. Currently there has been more than 1,200 participants and 110T of data points for several sessions (NEUROMINE,).

It is very helpful for studying prognosis that both PRO-ACT and Answers ALS Data Portal present longitudinal data. All three are public, but can only be downloaded by submitting a form for requesting the data and being approved after analysis. Additionally, none of them have declared to have collected any imaging modalities from patients.

2.3 Natural Language Processing

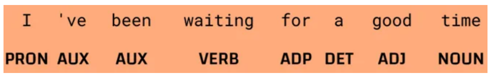
2.3.1 Introduction to NLP

Natural Language Processing (NLP) is a technique developed in the field of Artificial Intelligence (AI) that combines both natural human languages and computers. Its origins trace back to the early 1950s, defined initially as the joining of computer sciences and linguistics. The history behind NLP is marked by clear phases, including machine translation between languages, an AI-driven and semantics-oriented era and a grammatical-logical stage (FANNI et al., 2023). In the 90s, statistical models for NLP became specially popular, comprising simple and robust approximations, more rigorous evaluation, Machine Learning methods and employing large texts in algorithm training.

Some ML techniques are particularly common in NLP development. Support Vector Machines are applied for classifying inputs into categories. Hidden Markov Models might be useful to solve problems such as inference, pattern matching and training, and are extensively used for speech recognition. Conditional Random Fields can be of interest in predicting the classification for different entities in a passage. N-grams, or sequences of N items, may have their frequency/distribution calculated. In this case, items refer to either characters, words or phonemes.

According to (NADKARNI; OHNO-MACHADO; CHAPMAN, 2011), NLP tasks involve the following steps.

1. Sentence boundary detection: Determine how sentences are delimited.
2. Tokenization: Identify tokens, that is, words or punctuations, within a sentence.
3. Part-of-speech assignment to individual words: Classify tokens grammatically as pronouns, nouns, verbs, adjectives, etc. An example is presented by figure 4.



I	've	been	waiting	for	a	good	time
PRON	AUX	AUX	VERB	ADP	DET	ADJ	NOUN

Figure 4 – **Example of POS tagging.**

Extracted from (GOPENAI, 2023)

4. Morphological decomposition of compound words: An important sub-task consists in converting a word to a root using suffix removal. Crucial in synthetic languages, such as German and Hungarian.
5. Shallow parsing: Recognition of phrases. An example may be consulted in figure 5, where a sentence (S) is parsed in noun phrases (NP) and verb phrases (VP).
6. Problem-specific segmentation: Separating texts per segments considered to be meaningful.

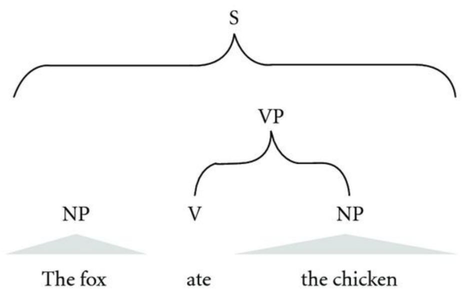


Figure 5 – **Example of shallow parsing.**

Extracted from (THESSEN; CUI; MOZZHERIN, 2012)

2.3.2 NLP's Current State

Recently, the language representation model proposed as BERT (Bidirectional Encoder Representations from Transformers) (DEVLIN et al., 2018b) has gained visibility. Differently from previous solutions, it reads the sequence of words in both directions, left-to-right and right-to-left, instead of in only either one of them. In such manner, it is capable of learning more about language context and flow and it uses embeddings to compare results to queries, which is more representative than simple word matching. Nonetheless, BERT also presents limitations, as large model sizes and presenting a maximum input length of usually 512 tokens.

Open-source pre-trained models (PTMs) that have deployed BERT can be found in Hugging Face (JONES et al., 2024), a prominent PTM registry in the NLP community. Its goal is to promote collaboration, so models may be continuously used and improved. The platform does not only give access to pre-trained algorithms, but it also provides good documentation and transparency, helping users to learn and better understand the code development.

2.3.3 NLP Applications

By converting written and spoken human language into computationally exploitable data, NLP applications may be categorized in two main types: Natural Language Understanding (NLU) for extracting meaning or identifying context from a text and Natural Language Generation (NLG) for producing human-like texts. It already permeates several current applications in computer software and in mobile phones, but also rising in healthcare. Even though NLP is only starting in hospital environments, it has presented significantly good results, facilitating and speeding doctors and researchers' tasks. Information extraction and retrieval are two of the most used applications in the medical domaine.

Among Natural Language Processing usages in healthcare, there is MedLEE, used for detecting patients suspected to present tuberculosis and breast cancer; CTakes, which has been used to evaluate pneumonia diagnosis based on radiology reports; Cogstack, for information retrieval and extraction implemented in King's College Hospital in the United Kingdom (FANNI et al., 2023). IBM has too conducted a medical diagnosis demonstration, using Watson, its supercomputer famous for its highly parallel hardware (NADKARNI; OHNO-MACHADO; CHAPMAN, 2011).

Thus, it has proven to be beneficial in biomedical contexts as well, even when considering relatively large bodies of text. Despite still being limited for diagnosis given its legal and medical implications, for instance, it can be largely used for collecting important data and in question answering applications.

2.4 Legal implications

Considering the current context of Generative AI growing usage, laws all around the world have been readapted to approach cases of Text and Data Mining, also comprising the application of Webscraping and NLP methods. The act of automatically collecting online data, referred to as Webscraping, is probably still one of the least defined on how it stands in copyright legislation. Nonetheless, there are specific cases that have occurred in the United States and can be analyzed for reference.

According to (ILIN; KELLI, 2024), the case of Doe et al. v. GitHub, Inc. et al., 22 is an example where the defendants (Microsoft, GitHub and OpenAI) were accused of what was called as "software piracy". By using scraped data from public code repositories to train GitHub Copilot, an AI coding assistant, they were alleged to have violated the open-source licenses, that require crediting the codes' authors, and GitHub's terms of service and privacy policies. This is still an open case in November 2024.

On the other hand, there were cases that were favorable to allowing online data collection as long as it is publicly available and doesn't require the acceptance of the website's terms of conditions in order to be accessed. In the hiQ Labs, Inc v. Linkedin Corporation, hiQ was prosecuted for supposedly infringing Linkedin's User Agreement by collecting public data from its website. However, The US Federal Court's decision in the hiQ v. Linkedin case has stated the following (text extracted from (COURTS, 2022)):

Although there are significant public interests on both sides, the district court properly determined that, on balance, the public interest favors hiQ's position. We agree with the district court that giving companies like LinkedIn free rein to decide, on any basis, who can collect and use data — data that the companies do not own, that they otherwise make publicly available to viewers, and that the companies themselves collect and use —risks the possible creation of information monopolies that would disserve the public interest.

Regarding Text and Data Mining (TDM), laws are typically more specific. TDM is composed of identifying text to be analyzed, copying significative amounts of materials, extracting data and recombination. Since scientific publications in any format are copyrightable, applying mining techniques, such as machine learning (COLONNA, 2017), to academic papers, may be classified as copyright infringement, but it depends.

The argumentation described in (LIPTON, 2020) shows that some academic journal publishers, e.g. Elsevier and Oxford University Press, provide permission for performing text and data mining in their content for non-commercial use. In these cases, it is not necessary to seek permission from publishers individually to reuse their work. In American legislation, although the fair use doctrine from Paragraph 107 of the United States Copyright Act varies for each situation, previous decisions have indicated that, when applied for research purposes, most TDM is classified as fair use.

In Australia, by 2020, the system allows use of copyrighted works in certain exceptions, as in research and study, for instance. In 2014, the United Kingdom adopted exemption to text and data mining when it comes to non-commercial research. As it it is stated in the amended law ([436], Par. 29A):

. . . the making of a copy of a work by a person who has lawful access to that work does not infringe copyright if it is made so that that person can carry out a computational analysis of anything included in that work for non-commercial research purposes.

In other European countries, which encompasses France, Estonia and Germany, it is also stated by law that, as long as there isn't commercial intent in the TDM application, it is legal. For France, specifically, it is important that the data is reproduced only from lawful sources. Even though Europe has been recently updating copyright law statements, it is still argued by scholars and policymakers that Europe stands behind United States on copyright flexibility, delaying advances in data usage.

To conclude, Brazilian law has also been recently rediscussed in light of recent global changes to legislation. The Federal senate's PL number 2.338/2023, more specifically in article 42, only authorizes data access and usage by entities whose mission is connected to the public's interest, as for research institutions. It is also clearly stated that, concerning protected work, TDM must not implicate in the creation of concurring products or that may impact normal exploration of the used work (SOUZA; SCHIRRU; ALVARENGA, 2024).

3 Methods

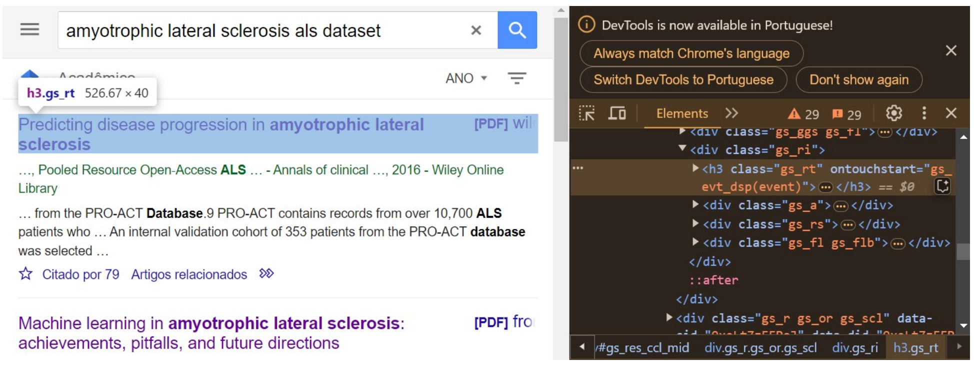
As described previously in chapter 2, there were 5 main steps that were necessary for reaching the objectives in this project. Therefore, the following sections are divided according to the work developed for each step.

3.1 Article collection

At first, it was thought that article collection could be achieved by developing a Python code as the following. The logic consisted in getting the results in the Google Scholar website for the query “amyotrophic lateral sclerosis als dataset” per page. Each page contains 10 different papers and the program iterates throughout the maximum number of pages possible. After, having as reference the class names for each item displayed in Google Scholar from the HTML code, which may be obtained by inspecting its page as represented in Figure 6, basic information on the results could be collected, such as title, authors, year of publication, access link, among others. In this case, the BeautifulSoup library was very useful for storing the HTML source codes.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import re
import cloudscraper

def get_html_from_page(page):
    url = f'https://scholar.google.com/scholar?hl=pt-BR&
       as_sdt=0%2C5&q=amyotrophic+lateral+sclerosis+als+
       dataset&btnG=&start={page}'
    scraper = cloudscraper.create_scraper(
        browser={
            'browser': 'chrome',
            'platform': 'android',
            'desktop': False
        }
    )
    response = scraper.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    return soup, response.status_code
```

```
def get_scholar_articles(soups):
    scholar_articles = []
    for soup in soups:
        for el in soup.select('.gs_r'):
            if len(el.select('.gs_rt')) > 0:
                scholar_articles.append({
                    'title': el.select('.gs_rt')[0].text,
                    'title_link': el.select('.gs_rt␣a')[0]['
                       href'],
                    'authors': el.find("div", {"class": "
                       gs_a"}).get_text(),
                    'year': re.findall(r'\d+', el.find("div"
                       , {"class": "gs_a"}).get_text())[0],
                    'cited_by_count': el.select('.gs_nph+␣a'
                       )[0].text,
                    'cited_link': 'https://scholar.google.
                       com' + el.select('.gs_nph+␣a')[0]['
                       href']})
    return scholar_articles

soup, code = get_html_from_page(0)
if code == 200:
    nb_of_results = 0
    for word in soup.find('div', {'id': 'gs_ab_md'}).
       get_text().split():
        if word.replace('.', '').isnumeric():
            nb_of_results = int(word.replace('.', ''))

all_pages_soups = []
for page in range(nb_of_results):
    html, code = get_html_from_page(page)
    if code == 200:
        all_pages_soups.append(html)
    else:
        break

scholar_articles = get_scholar_articles(all_pages_soups)
```


amyotrophic lateral sclerosis als dataset
ANO
h3.gs_rt 526.67 × 40
Citado por 79
Artigos relacionados
DevTools is now available in Portuguese!
Always match Chrome's language
Switch DevTools to Portuguese
Don't show again
Elements
29
29
#gs_res_ccl_mid
div.gs_r.gs_or.gs_scl
div.gs_ri
h3.gs_rt

Predicting disease progression in **amyotrophic lateral sclerosis** [PDF] wil

…, Pooled Resource Open-Access **ALS** … - Annals of clinical …, 2016 - Wiley Online Library

… from the PRO-ACT **Database**.9 PRO-ACT contains records from over 10,700 **ALS** patients who … An internal validation cohort of 353 patients from the PRO-ACT **database** was selected …

Machine learning in **amyotrophic lateral sclerosis**: achievements, pitfalls, and future directions [PDF] fro

```
<div class="gs_ggs gs_fl">…</div>
<div class="gs_ri">
  <h3 class="gs_rt" ontouchstart="gs_
  evt_dsp(event)">…</h3> == $0
  <div class="gs_a">…</div>
  <div class="gs_rs">…</div>
  <div class="gs_fl gs_flb">…</div>
 </div>
 ::after
</div>
<div class="gs_r gs_or gs_scl" data-
```

Figure 6 – **Google Scholar inspection for obtaining class names of each item displayed in its page.**

Then, the full texts for each article should be accessed in order to identify further information on the ALS datasets. To do so, the source links for each one of them would have to be explored as well. However, most papers are protected by a CAPTCHA system, intended to differentiate human users from bots trying to access them. In this context, it prevents the Webscraping methods hereby used from accessing the needed texts.

Several libraries were tested for making the GET request on the articles' websites, including the Python standard library **requests**, **selenium** (MUTHUKADAN,) along with a Chrome Driver, **cloudflare** (CLOUDFLARE-SCRAPE,), **httpx** (HTTPX. . . ,) and **cloudscraper** (PYPI, 2023). Nonetheless, no option worked, receiving the *forbidden* code as 403. Other solutions involved using proxies and Anti-CAPTCHA algorithms, however this could implicate legal issues.

By better studying laws concerning Webscraping, as it was described in Chapter 2, it was decided that it was best not to automatize the gathering of papers. So, even though there are tools such as PyPaperBot (FERRULLI,), manual download was the only viable solution, given blurry legal implications of such automatic collection.

3.2 Text preprocessing

The **pypdf** library (PYPI, b) was used to read the papers PDF files and provide the text in Python. Yet, the full texts could not be simply provided to the Natural Language Processing (NLP) models to be analyzed. An extensive amount of large texts could result in reaching memory limits and high execution time. Furthermore, NLP models may have a maximum number of accepted tokens. For such reasons, it was decided to extract the abstracts from the scientific articles, when possible. To detect the abstracts, two heuristics were adopted:

- It is assumed that the abstract will be found in the first page of the PDF file. If not found in the first page, it is searched for in the second page, which may be the case when there is a cover page for the article.

- The abstract is supposed to be between the word “Abstract” and “Keywords” or “Key words". Several articles are constructed as presenting both items.

The implementation in Python, shown below, consists in using regex pattern identification for finding the beginning of the abstract, indicated by the word “abstract”, and the end of it, indicated by the word “keywords” or “key words”. If both searches result in non-empty objects and abstract comes before keywords, the abstract is considered to be found. First this process is done for the first page, but if it doesn’t succeed, it is attempted for the second page as well. When the abstract can’t be encountered neither in the first page, nor in the second page, the full text from the first page is used.

```
pdf_files = [file for file in os.listdir() if 'pdf' in file]
for pdf_file in pdf_files:
    reader = PdfReader(pdf_file)
    page = reader.pages[0]
    abstract = page.extract_text().lower()
    answers = []

    abstract_start = re.search(r'abstract', abstract)
    abstract_end = re.search(r'key(\s?)words', abstract)
    print(pdf_file)
    if abstract_start != None:
        if abstract_end != None:
            if (abstract_start.span()[1] < abstract_end.span
                ()[0]):
                 abstract = abstract[abstract_start.span()
                    [1]:abstract_end.span()[0]]
            else:
                 abstract = abstract[abstract_start.span()
                    [1]:]
        else:
            abstract = abstract[abstract_start.span()[1]:]
    else:
        page = reader.pages[1]
        abstract = page.extract_text().lower()

        abstract_start = re.search(r'abstract', abstract)
        abstract_end = re.search(r'key(\s?)words', abstract)
```

```
if abstract_start != None:
    if abstract_end != None:
        if (abstract_start.span()[1] < abstract_end.
            span()[0]):
            abstract = abstract[abstract_start.span
                ()[1]:abstract_end.span()[0]]
        else:
            abstract = abstract[abstract_start.span
                ()[1]:]
    else:
        abstract = abstract[abstract_start.span()
            [1]:]
```

3.3 Data characterization

In order to characterize the dataset, Natural Language Processing techniques were chosen for detecting information automatically in the collected texts. However, as the idea is to completely automatize the process and human intervention should not be necessary in order to attest the correctness of the gathered data, a verification system has to be proposed to avoid misinformation. Thus, it was decided to use the concept of Triple Modular Redundancy, represented by Figure 7, to reduce errors from this step.

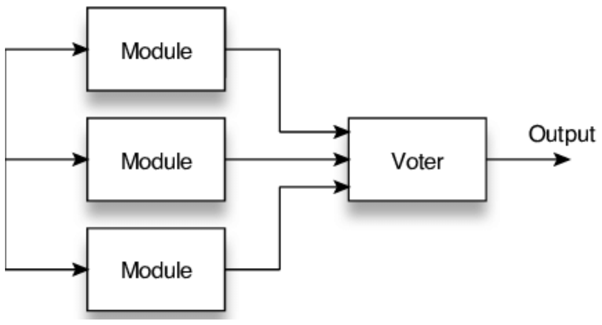


Figure 7 – **Triple Modular Redundancy.**

Extracted from (TARASYUK; TROUBITSYNA; LAIBINIS, 2010).

In this approach, three different NLP modules are used for analyzing the text. Considering that there are queries that should be answered by the models given a context, the modules selected are placed in the Question Answering category. Then, a voter, provided with the answers given by each model, it decides which answer should be outputted.

The context in each article is the abstract extracted in the previous step. The queries to be answered are the following: “Which dataset is being used?”, “What kind of data is being used?" and "How many subjects in the dataset?". The three modules correspond to three different Question Answering NLP models obtained from the Hugging Face platform, including **distilbert/distilbert-base-cased-distilled-squad** (SANH et al., 2019), **google-bert/bert-large-cased-whole-word-masking-finetuned-squad** (DEVLIN et al., 2018a) and **deepset/roberta-base-squad2** (HAYSTACK,). All algorithms have been selected by presenting a large amount of downloads made by other Hugging Face users and a high F1-score.

Model	**Downloads last month (in November 25th 2024)**	**F1-score**	**Evaluation dataset**	**Model size**
roberta-base-squad2	1,116,795	82.91	SQuAD 2.0 dev set	124M params
bert-large-cased	174,511	83.1	SQuAD v2.0	334M params
distilbert-base-cased	358,553	87.1	SQuAD v1.1 dev set	65.2M params

Table 2 – **Comparison between the chosen NLP models.**

Although all three use the language representation from BERT, each has its own training procedure. The roberta model has been trained on question-answer pairs, which includes unanswerable questions, from SQuAD 2.0. The bert-large-cased model is pretrained using a masked language modeling objective and trained using Whole Word Masking, where each masked WordPiece token is predicted independently. DistilBERT uses the BERT base model as a teacher and is also pretrained on the same database in a self-supervised manner, but it is smaller and faster.

So, for each NLP module, the following code in Python is run for obtaining the answers to three questions defined previously.

```
queries = ['Which dataset is being used?', 'What kind of data is
    being used?', 'How many subjects in the dataset?']
all_answers = {}
for pdf_file in pdf_files:
    answers = []
    for query in queries:
        answers.append(model(question=query, context=abstract)['
            answer'])
    all_answers[pdf_file] = answers
```

The definition of each model is done using the **pipeline** function provided by the transformers library from **Hugging Face**, as shown in the code below.

```
google_bert = pipeline('question-answering', model='google-bert/
    bert-large-uncased-whole-word-masking-finetuned-squad')
```

```
answers_google_bert = extract_data_from_abstract(pdf_files ,
    google_bert)
roberta = pipeline('question-answering', model='deepset/roberta-
    base-squad2')
answers_roberta = extract_data_from_abstract(pdf_files , roberta)
distilbert = pipeline('question-answering', model='distilbert/
    distilbert-base-cased-distilled-squad')
answers_distilbert = extract_data_from_abstract(pdf_files ,
    distilbert)
```

At last, it was necessary to define a voter using heuristics so an answer could be selected among the three outputs obtained. I decided to compute the compatibility among the answers provided by each NLP model and, if at least two of the collected information presented a similitarity of 50% or higher, the longest output, considered to be the most complete, was chosen. So, if the three answers were more than 50% compatible, the longest among all three would be selected. If only two answers were more than 50% compatible, the longest among both would be selected. If there wasn't enough compatibility among no pair of answers, none would be selected.

```
all_answers = {}
for file in answers_google_bert.keys():
    answers = []
    for i in range(3):
        answer_google_bert = answers_google_bert[file][i]
        answer_distilbert = answers_distilbert[file][i]
        answer_roberta = answers_roberta[file][i]
        ratio_google_roberta = fuzz.token_set_ratio(
            answer_google_bert , answer_roberta) >= 50
        ratio_google_distilbert = fuzz.token_set_ratio(
            answer_google_bert , answer_distilbert) >= 50
        ratio_distilbert_roberta = fuzz.token_set_ratio(
            answer_distilbert , answer_roberta) >= 50
        if (ratio_google_roberta and ratio_google_distilbert and
            ratio_distilbert_roberta):
            answers.append(max([answer_google_bert ,
                answer_distilbert , answer_roberta], key=len))
        elif (ratio_google_roberta):
            answers.append(max([answer_google_bert ,
                answer_roberta], key=len))
        elif (ratio_google_distilbert):
                answers.append(max([answer_google_bert,
                    answer_distilbert], key=len))
            elif (ratio_distilbert_roberta):
                answers.append(max([answer_distilbert,
                    answer_roberta], key=len))
            else:
                answers.append(None)
        all_answers[file] = answers
```

The function fuzz.token_set_ratio() from the **fuzzywuzzy** library in Python (PYPI, a) was useful for calculating the compatibility among strings. It uses the Levenshtein Distance as metric, that is, the minimum number of single-characters edits needed for transforming one word into another, including insertions, deletions and/or substitutions.

Having the modular system ready for functioning, preprocessing along with information gathering were executed for every article collected. Next, the results that were returned were saved on an numpy file format and on an excel file. In order to evaluate the performance of the voter on filtering bad answers, each paper and the corresponding given output were manually compared.

3.4 Data provision

In order to make data available and to facilitate further analysis on the obtained results, it was opted to compile all the information in a DataFrame object from the **pandas** library. In a first instance, a CSV file with basic data on the articles was imported. Then, using the “PDF Name" column, both data from the CSV and data collected using the NLP models after preprocessing the texts were merged into a DataFrame. The code below shows how this is written in Python.

```
df = pd.read_csv('result.csv')
df['Database'] = [None] * len(df)
df['Data␣Type'] = [None] * len(df)
df['Amount␣of␣Data'] = [None] * len(df)
for pdf_file in all_answers.keys():
    df.loc[df['PDF␣Name'] == pdf_file, ['Database', 'Data␣Type',
        'Amount␣of␣Data']] = all_answers[pdf_file][0],
      all_answers[pdf_file][1], all_answers[pdf_file][2]
df.head()
```

Then, three different plots were created to help summarizing the distribution of the obtained data. One for the amount of papers published per year in the given theme, another for the proportion of types of data that were detected (percentage of clinical, genetic, imaging and others) and classification of datasets by different amounts of data made available (between 0-100, 101-1000, 1001-10000 and 10001+). The following piece of code was developed for drawing the plots. The images for the graphs must be saved locally so they can be properly sent by the API afterwards.

```
import seaborn as sns
import matplotlib.pyplot as plt
import io
from flask import send_file
from flask import request

df_filtered = df[['Name', 'Scholar␣Link', 'Year', 'Authors', '
    Database', 'Data␣Type', 'Amount␣of␣Data']]
plt.ylabel('Count')
plt.xlabel('Amount␣of␣data')
plt.bar(intervals_amount_of_data.keys(),
    intervals_amount_of_data.values(), edgecolor='black')
plt.savefig('AmountData.png')

plt.clf()
total = len(df['Data␣Type'].dropna())
clinical = sum([('clinic' in x) for x in df['Data␣Type'].dropna
    ()])
genetic = sum([('gene␣' in x or 'genetic' in x) for x in df['
    Data␣Type'].dropna()])
imaging = sum([('imaging' in x or 'image' in x) for x in df['
    Data␣Type'].dropna()])
others = total - sum([clinical, genetic, imaging])
data = [clinical, genetic, imaging, others]
keys = ['Clinical', 'Genetic', 'Imaging', 'Others']
explode = [0, 0, 0.3, 0]
palette_color = sns.color_palette('viridis')
plt.pie(data, labels=keys, colors=palette_color,
        explode=explode, autopct='%.0f%%')
plt.savefig('DataType.png')

plt.clf()
sns.histplot(df['Year'], palette='viridis')
plt.savefig('Year.png')
```

Having all information compiled and the plots generated, it is possible to make it available externally through an API. In this case, the **flask** library (FLASK,) in Python was used. It is important to note that, in order to create the interface, it is necessary to configure CORS (Cross-Origin Resource Sharing) so it may be accessed by other domains. There are five paths that are made available, returning different information.

Three functions are relative to the three plots created and described previously. One method is to make available the general data collected for each article. In this case, there are variables that may also be included in the request URL to filter the provided data, such as starting year and ending year, so only papers published between both years are returned. It is also possible to determine which papers are described by the resulting json object using the data_type variable. Finally, there's one path that supplies information on the amount of papers that were analyzed, including how many there were initially, how many were downloaded by this study, among others.

```
app = Flask(__name__)
CORS(app)

@app.route('/data')
def get_data():
    df_final = df_filtered
    data_type = request.args.getlist('data_type')
    starting_year = request.args.get('starting_year')
    ending_year = request.args.get('ending_year')

    if (len(data_type) > 0):
        if ('clinical' in data_type):
            df_final = df_filtered[[('clinic' in x) if (x !=
                None) else False for x in df_filtered['Data␣Type'
                ]]]
        if ('genetic' in data_type):
            if ('clinical' in data_type):
                df_final = pd.concat([df_final, df_filtered[[('
                    gene␣' in x or 'genetic' in x) if (x != None)
                     else False for x in df_filtered['Data␣Type'
                    ]]]]).drop_duplicates(subset='Name')
            else:
                df_final = df_filtered[[('gene␣' in x or '
                    genetic' in x) if (x != None) else False for
                    x in df_filtered['Data␣Type']]]
        if ('imaging' in data_type):
            if ('clinical' in data_type or 'genetic' in
               data_type):
                df_final = pd.concat([df_final, df_filtered[[(
                    'imaging' in x or 'image' in x) if (x != None)
                     else False for x in df_filtered['Data␣Type'
                    ]]]]).drop_duplicates(subset='Name')
            else:
                df_final = df_filtered[[('imaging' in x or '
                    image' in x) if (x != None) else False for x
                    in df_filtered['Data␣Type']]]

        if (starting_year != None):
            df_final = df_final[df_final['Year'] > float(
                starting_year)]

        if (ending_year != None):
            df_final = df_final[df_final['Year'] < float(ending_year
                )]

        df_final.fillna('', inplace=True)
        data_json = df_final.to_dict('records')
        return jsonify(data_json)

@app.route('/plot_year')
def get_plot_year():
    with open('Year.png', 'rb') as fh:
        bytes_year = io.BytesIO(fh.read())
        return send_file(bytes_year,
                        download_name='plot_year.png',
                        mimetype='image/png')

@app.route('/plot_data_type')
def get_plot_data_type():
    with open('DataType.png', 'rb') as fh:
        bytes_data_type = io.BytesIO(fh.read())
        return send_file(bytes_data_type,
                        download_name='plot_data_type.png',
                        mimetype='image/png')
```

```
@app.route('/plot_amount_data')
def get_plot_amount_data():
    with open('AmountData.png', 'rb') as fh:
        bytes_amount_data = io.BytesIO(fh.read())
        return send_file(bytes_amount_data,
                         download_name='plot_amount_data.png',
                         mimetype='image/png')

@app.route('/info_results')
def get_info_results():
    info_results = {}

    columns = ['Name', 'DOI', 'Year', 'Journal', 'Authors', '
       Database', 'Data␣Type', 'Amount␣of␣Data']
    for column in columns:
        info_results[column] = str(df[column].isna().
           value_counts()[0])
    info_results['Downloaded'] = str(df['Downloaded'].
       value_counts()[1])
    print(info_results)
    return jsonify(info_results)

app.run()
```

3.5 User interface

To develop the user interface, the REACT framework was selected given that it is currently a technology extensively used in companies and, more importantly, well documented. By fetching the information provided by the API written in Python and described previously, a simple page was created so databases on ALS could be consulted and filtered and data collected could be summarized by the drawn plots.

In order to fetch the data from the API, the following functions were used. The UseEffect function was used so data would be updated every time the page was refreshed.

```
    const fetchPlotYear = async () => {
        const res = await fetch(baseURL + "plot_year");
        const imageBlob = await res.blob();
        const imageObjectURL = URL.createObjectURL(imageBlob);
        setPlotYear(imageObjectURL);
```

```
      console.log(imageObjectURL)
};

const fetchPlotDataType = async () => {
  const res = await fetch(baseURL + "plot_data_type");
  const imageBlob = await res.blob();
  const imageObjectURL = URL.createObjectURL(imageBlob);
  setPlotDataType(imageObjectURL);
  console.log(imageObjectURL)
};

const fetchPlotAmountData = async () => {
  const res = await fetch(baseURL + "plot_amount_data");
  const imageBlob = await res.blob();
  const imageObjectURL = URL.createObjectURL(imageBlob);
  setPlotAmountData(imageObjectURL);
  console.log(imageObjectURL)
};

useEffect(() => {
  fetchPlotYear();
  fetchPlotDataType();
  fetchPlotAmountData();
}, []);

useEffect(() => {
  fetch(baseURL + "data")
  .then((res) => {
    return res.json();
  })
  .then((data) => {
    setDatabasesData(data);
    setDataToDisplay(data.slice(0, TOTAL_VALUES_PER_PAGE));
  });
}, []);

useEffect(() => {
  fetch(baseURL + "info_results")
  .then((res) => {
      return res.json();
    })
    .then((data) => {
      setInfoResults(data);
    });
  }, []);
```

Specifically for when data filtering is applied, it was necessary to get values from input fields placed in the user interface, and only then build the request URL specifying the selected variables, which included starting year, ending year and type of data. The following code refers to this specific case.

```
    function filterData() {
    var link = baseURL + "data?";
    if (clinical) {
      link = link + "data_type=clinical&";
    }
    if (genetic) {
      link = link + "data_type=genetic&";
    }
    if (imaging) {
      link = link + "data_type=imaging&";
    }
    if (startingYear !== null && startingYear !== "") {
      link = link + "starting_year=" + startingYear + "&";
    }
    if (endingYear !== null && endingYear !== "") {
      link = link + "ending_year=" + endingYear + "&";
    }

    setCurrentPageNumber(1);

    fetch(link)
    .then((res) => {
      return res.json();
    })
    .then((data) => {
      setDatabasesData(data);
      console.log(data.length);
      setDataToDisplay(data.slice(0, TOTAL_VALUES_PER_PAGE));
    });
  }

    <input className="Databases-toggle" type="checkbox" onChange
        ={(e) => {setClinical(e.target.checked)}}/><span style
        ={{"padding": "10px"}} className="Home-text">Clinicos</
        span>
    <input className="Databases-toggle" type="checkbox" onChange
        ={(e) => {setGenetic(e.target.checked)}}/><span className
        ="Home-text" style={{"padding": "10px"}} >Geneticos</span
        >
    <input className="Databases-toggle" type="checkbox" onChange
        ={(e) => {setImaging(e.target.checked)}}/><span className
        ="Home-text" style={{"padding": "10px"}} >Imagens</span>
```

To display the data in a table format, the code was the following. It iterates over the result provided by the API and has as columns Name, Authors, Year, Scholar Link, Database, Data Type and Amount of Data for each paper.

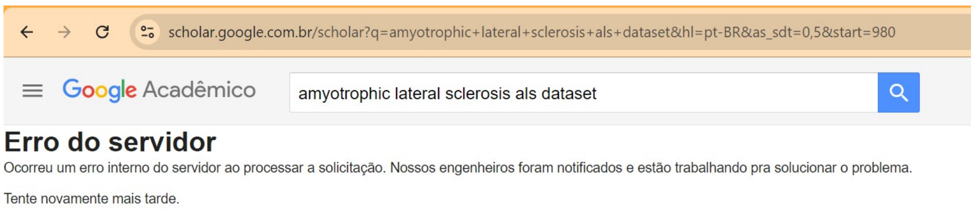
```
      const DisplayData=dataToDisplay.map(
        (info)=>{
            return(
                <tr>
                    <td>{info['Name']}</td>
                    <td>{info['Authors']}</td>
                    <td>{info['Year']}</td>
                    <td style={{maxWidth: '300px', wordWrap: '
                        break-word'}}><a href={info['Scholar Link
                        ']}>{info['Scholar Link']}</a></td>
                    <td>{info['Database']}</td>
                    <td>{info['Data Type']}</td>
                    <td>{info['Amount of Data']}</td>
                </tr>
            )
        }
      )

    <table className="table table-striped">
            <thead>
                <tr>
                <th>Name</th>
                <th>Authors</th>
                <th>Year</th>
                <th>Scholar Link</th>
                <th>Database</th>
                <th>Data Type</th>
                <th>Amount of Data</th>
                </tr>
            </thead>
            <tbody>
                {DisplayData}
            </tbody>
    </table>
```

Then, to present general information on the obtained results and the plots summarizing the data usage from academic work on ALS, the structure below is used.

```
    <div className='Home-text'>Dos {infoResults['Name']} artigos
        resultantes da pesquisa no Google Scholar, {infoResults
       ['Downloaded']} puderam ser baixados. Dentre estes, foram
        coletados datasets para {infoResults['Database']}
       artigos, tipos de dados para {infoResults['Data Type']}
       artigos e quantidade de dados para {infoResults['Amount
       of Data']} artigos. Informacoes quantificaveis sao
       resumidas pelos graficos a seguir.</div>
    <div className="Summary" style={{ display: "grid",
       gridTemplateColumns: "repeat(3, 1fr)", gridGap: 20 }}>
        <div><p className='Home-text'>Publicacoes coletadas por
           ano</p><img src={plotYear} alt="/plotYear" /></div>
        <div><p className='Home-text'>Distribuicao dos tipos de
           dados coletados</p><img src={plotDataType} alt="/
           plotDataType" /></div>
        <div><p className='Home-text'>Distribuicao da quantidade
            de dados nos datasets</p><img src={plotAmountData}
           alt="/plotAmountData" /></div>
  </div>
```

4 Results

In the article collection phase, 980 academic paper titles concerning Amyotrophic Lateral Sclerosis datasets were consulted in Google Scholar. Even though it indicated during search that approximately more than 30.000 results had been found, from page 98 and above, the error shown in Figure 8 was outputted.


scholar.google.com.br/scholar?q=amyotrophic+lateral+sclerosis+als+dataset&hl=pt-BR&as_sdt=0,5&start=980

Google Acadêmico

amyotrophic lateral sclerosis als dataset

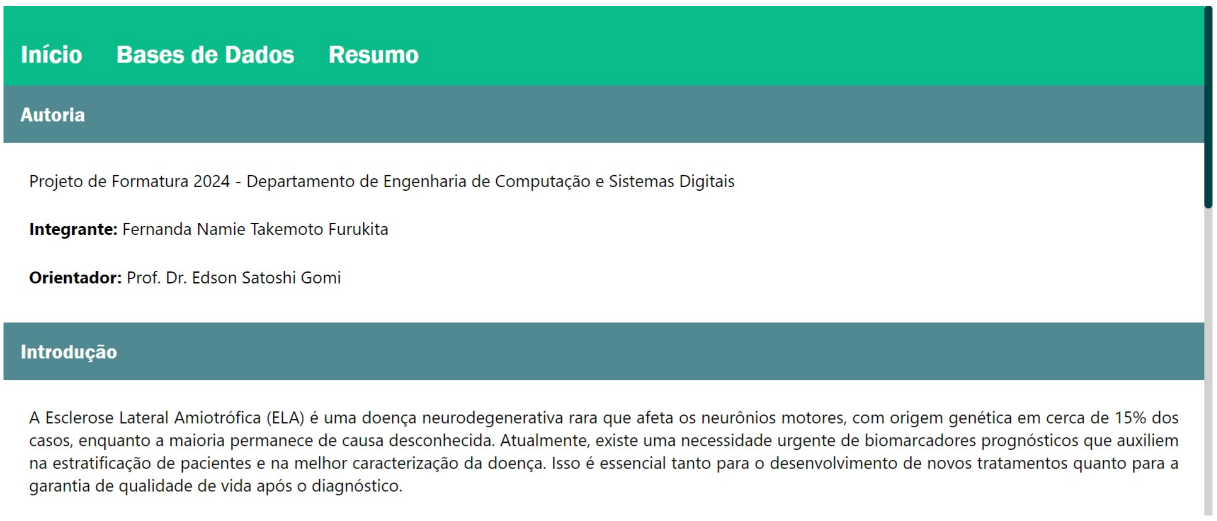
Erro do servidor

Ocorreu um erro interno do servidor ao processar a solicitação. Nossos engenheiros foram notificados e estão trabalhando pra solucionar o problema.

Tente novamente mais tarde.

Figure 8 – Error in Google Scholar.

From the exploitable articles, 666 of them could be downloaded and processed in the following stages. Among these, it was automatically found information regarding the used database for 412, the data type for 405 and amount of data for 479. Figures 9, 10 and 11 show the User Interface along with the generated data.



Autoria

Projeto de Formatura 2024 - Departamento de Engenharia de Computação e Sistemas Digitais

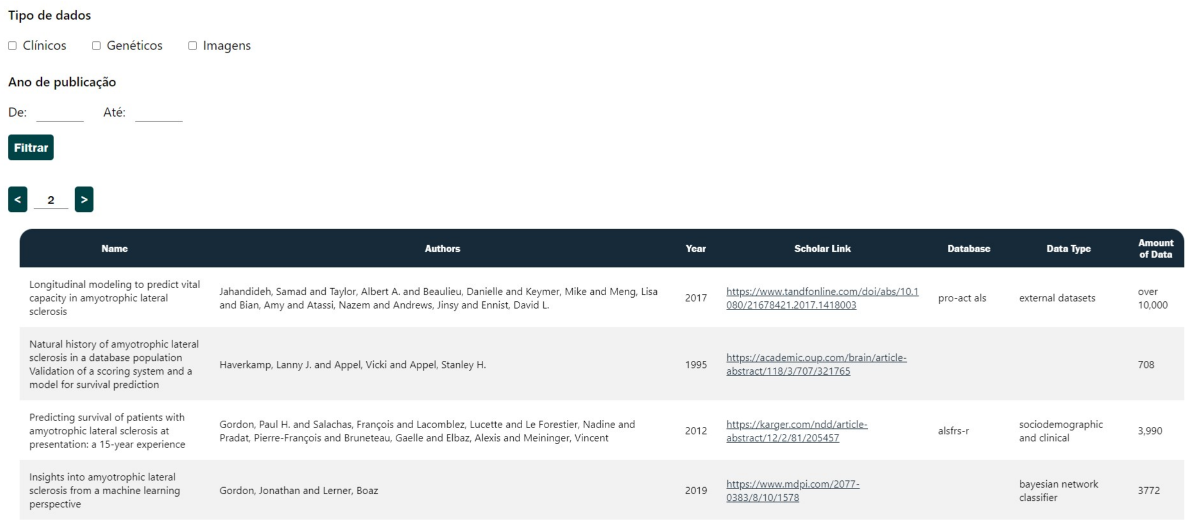
Integrante: Fernanda Namie Takemoto Furukita

Orientador: Prof. Dr. Edson Satoshi Gomi

Introdução

A Esclerose Lateral Amiotrófica (ELA) é uma doença neurodegenerativa rara que afeta os neurônios motores, com origem genética em cerca de 15% dos casos, enquanto a maioria permanece de causa desconhecida. Atualmente, existe uma necessidade urgente de biomarcadores prognósticos que auxiliem na estratificação de pacientes e na melhor caracterização da doença. Isso é essencial tanto para o desenvolvimento de novos tratamentos quanto para a garantia de qualidade de vida após o diagnóstico.

Figure 9 – Home Page in the developed User Interface.


Tipo de dados

☐ Clínicos ☐ Genéticos ☐ Imagens

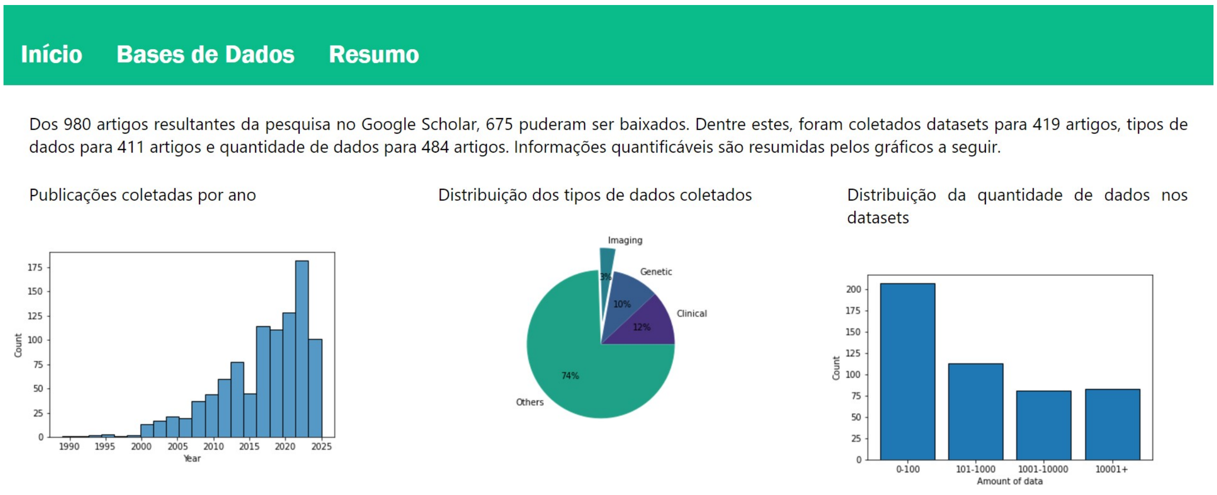
Ano de publicação

De: ______ Até: ______

Name	Authors	Year	Scholar Link	Database	Data Type	Amount of Data
Longitudinal modeling to predict vital capacity in amyotrophic lateral sclerosis	Jahandideh, Samad and Taylor, Albert A. and Beaulieu, Danielle and Keymer, Mike and Meng, Lisa and Bian, Amy and Atassi, Nazem and Andrews, Jinsy and Ennist, David L.	2017	https://www.tandfonline.com/doi/abs/10.1080/21678421.2017.1418003	pro-act als	external datasets	over 10,000
Natural history of amyotrophic lateral sclerosis in a database population Validation of a scoring system and a model for survival prediction	Haverkamp, Lanny J. and Appel, Vicki and Appel, Stanley H.	1995	https://academic.oup.com/brain/article-abstract/118/3/707/321765			708
Predicting survival of patients with amyotrophic lateral sclerosis at presentation: a 15-year experience	Gordon, Paul H. and Salachas, François and Lacomblez, Lucette and Le Forestier, Nadine and Pradat, Pierre-François and Bruneteau, Gaelle and Elbaz, Alexis and Meininger, Vincent	2012	https://karger.com/ndd/article-abstract/12/2/81/205457	alsfrs-r	sociodemographic and clinical	3,990
Insights into amyotrophic lateral sclerosis from a machine learning perspective	Gordon, Jonathan and Lerner, Boaz	2019	https://www.mdpi.com/2077-0383/8/10/1578		bayesian network classifier	3772

Figure 10 – Databases Page in the developed User Interface.

The plots in Figure 11 represent, respectively from left to right, the amount of the collected papers published per year in a histogramme, the proportion of data type detected using the NLP models (imaging, clinical, genetic or others) and the distribution of amount of data obtained in automatic information gathering, separated by 1-100, 101-1000, 1001-10000 and 10000+.



Dos 980 artigos resultantes da pesquisa no Google Scholar, 675 puderam ser baixados. Dentre estes, foram coletados datasets para 419 artigos, tipos de dados para 411 artigos e quantidade de dados para 484 artigos. Informações quantificáveis são resumidas pelos gráficos a seguir.

Publicações coletadas por ano

Distribuição dos tipos de dados coletados

Distribuição da quantidade de dados nos datasets

Figure 11 – Summary Page in the developed User Interface.

5 Analysis and Evaluation

The Triple Modular Redundancy was the stage of the study that most demanded analysis so its performance could be evaluated, consequently it is the focus of this section. As a first metric, the amount of time that the method took was considerable, resulting in 7h 24m 22s in total. The machine where the code was executed used an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz as a processor and presented a 16 GB RAM.

Thus, to guarantee larger scalability of this model, perhaps it would be necessary to test performance in a GPU, for example, in order to evaluate if it performs betters by profiting of a parallelizable architecture, and seek other solutions to minimize wrong answers instead of the Triple Modular Rendundancy. By running three different models, execution time has been triplicated. Another option would be explore other NLP models that also present high F1-scores but run faster.

Secondly, it was important to determine how accurate the proposed approach is. In order to assure which answers were valid according to each paper, a manual analysis was made. So, each paper's abstract was read in order to assess if the collected information was correct or incorrect. Table 3 shows percentages of databases, data types and amounts of data considered to be correctly classified. However, when separating predictions obtained from articles that didn't follow the established heuristics for extracting the abstracts and the ones that followed it, different accuracies were obtained.

Information	**Total accuracy**	**Accuracy for detected abstracts**	**Accuracy for undetected abstracts**
Database	57.11%	66.67%	30.12%
Data Type	53.87%	57.79%	31.17%
Amount of Data	53.77%	61.14%	55.82%

Table 3 – **TMR Accuracies for each type of information and for different subgroups of abstracts.**

The Fisher's Exact Test was used to determine if there is statistical difference between the accuracies for the two sub-populations of predictions - where abstract was correctly found and where it hasn't been found. Such test has been selected for this case, given that it is applicable to binary sequences. The p-values were calculated and corresponded to 8.54e-06, 1.79e-05 and 8.85e-03 for the tests considering database, data type and amount of data respectively. As can be observed, using $\alpha = 0.01$, all are below the chosen parameter, indicating significant difference between the accuracies.

Therefore, by observing that only 34.09% of the papers had detectable abstracts in the proposed technique, it is possible to imply that improvements in the heuristics could aid to achieve better results. Other methods, more robust than stating heuristics, although more computationally costly, might be necessary. Another alternative would be to manually preprocess the texts and directly extract the abstracts for feeding the NLP models.

Finally, based on the gathered results, graphs represented in figure 11 presented in the previous section, summarize some of the characteristics from the papers and the data that they were classified to have used or to have collected. The plot on the left, indicating publications per year, shows that most studies have been more recently conducted. A considerable amount of papers have been published between 2015 and 2024. In addition, the graphs representing the proportions of collected data type and amount of data allow us to confirm earlier hypothesis that there are few imaging data and containing large numbers of participants.

6 Final Considerations

From the analysis presented in Section 5, it is possible to affirm that the proposed method was moderately successful in extracting relevant information on academic papers regarding existing datasets for Amyotrophic Lateral Sclerosis. By the end of the project, a platform was developed and could be easily navigated to consult the obtained results. So it may be concluded that the initial objective, to create a User Interface as source of information on ALS databases, has been met.

Nonetheless, it could still be improved and further developed by future work. More than searching for alternatives that could improve abstract detection and models that perform better in extracting information from scientific texts, new purposes for Natural Language Processing can be found in this line of work.

As an example, (LUO et al., 2024) has proven the value of Large Language Model (LLM) training in neurosciences. By using 332,807 abstracts and 123,085 full-text articles to train what has been called as BrainGPT, Luo et al. have achieved promising results for applying AI to provide information in the biomedical field.

Given that this project has shown potential for NLP models to gather data and how number of studies regarding ALS has been growing notably in the last few years, future work could include gathering papers to build a solid academic database with authors' permission and train an LLM to work as a chat bot for answering questions concerning the disease. This could be helpful not only in the scientific community researching the subject, but also to recently diagnosed patients that are not necessarily from the medical domain.

There is power in collecting and transforming information that is currently accessible to few to help propagating knowledge. In this context, Natural Language Processing might be used to minimize disinformation in the initial steps of ALS and turn the process of diagnosis acceptance less complicated. Thus, there is great potential in future works regarding this topic.

Bibliography

ATASSI, N. et al. The pro-act database: design, initial analyses, and predictive features. *Neurology*, v. 83, n. 19, p. 1719–1725, 2014. Citado na página 24.

BLEILE, B. et al. *The Persistent Homology of Dual Digital Image Constructions*. 2021. Disponível em: <https://arxiv.org/abs/2102.11397>. Citado na página 71.

BROOKS, B. R. et al. El escorial revisited: Revised criteria for the diagnosis of amyotrophic lateral sclerosis. *Amyotrophic Lateral Sclerosis and Other Motor Neuron Disorders*, Taylor & Francis, v. 1, n. 5, p. 293–299, 2000. PMID: 11464847. Disponível em: <https://doi.org/10.1080/146608200300079536>. Citado na página 19.

BUBENIK, P. *Statistical topological data analysis using persistence landscapes*. 2015. Disponível em: <https://arxiv.org/abs/1207.6437>. Citado na página 62.

BUBENIK, P. The persistence landscape and some of its properties. In: _____. *Abel Symposia*. Springer International Publishing, 2020. p. 97–117. ISBN 9783030434083. Disponível em: <http://dx.doi.org/10.1007/978-3-030-43408-3_4>. Citado na página 62.

CARRIèRE, M. et al. *PersLay: A Neural Network Layer for Persistence Diagrams and New Graph Topological Signatures*. 2020. Disponível em: <https://arxiv.org/abs/1904.09378>. Citado 2 vezes nas páginas 61 and 70.

CARRIèRE, M. et al. *Representations manual*. Disponível em: <https://gudhi.inria.fr/python/latest/representations.html>. Citado na página 70.

CHEPUSHTANOVA, S. et al. Persistence images: An alternative persistent homology representation. *CoRR*, abs/1507.06217, 2015. Disponível em: <http://arxiv.org/abs/1507.06217>. Citado na página 62.

CHOE, S.; RAMANNA, S. Cubical homology-based machine learning: An application in image classification. *Axioms*, v. 11, n. 3, 2022. ISSN 2075-1680. Disponível em: <https://www.mdpi.com/2075-1680/11/3/112>. Citado 2 vezes nas páginas 60 and 62.

CLEANING Confounders in your Data with Nilearn. 2024. Disponível em: <https://carpentries-incubator.github.io/SDC-BIDS-fMRI/05-data-cleaning-with-nilearn.html>. Citado na página 68.

CLOUDFLARE-SCRAPE. Disponível em: <https://github.com/Anorov/cloudflare-scrape>. Citado na página 31.

COLONNA, L. *Legal Implications of Data Mining: Assessing the European Union's Data Protection Principles in Light of the United States Government's National Intelligence Data Mining Practices*. [S.l.], 2017. (Research Paper No. 5). Legal Implications of Data Mining: Assessing the European Union's Data Protection Principles in Light of the United States Government's National Intelligence Data Mining Practices, Liane Colonna, eds., Beställningar, 2016. Disponível em: <https://ssrn.com/abstract=2924541>. Citado na página 27.

COMMUNITY, T. S. *mannwhitneyu - SciPy API*. 2008–2024. Disponível em: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mannwhitneyu.html>. Citado na página 86.

CONTI, D. C. Magnetic resonance imaging. 03 2016. Disponível em: <https://www.researchgate.net/publication/299512554_Magnetic_Resonance_Imaging>. Citado na página 58.

CONTRIBUTORS, B. *Brain Imaging Data Structure. Magnetic Resonance Imaging. Timing Parameters.* Disponível em: <https://bids-specification.readthedocs.io/en/stable/04-modality-specific-files/01-magnetic-resonance-imaging-data.html#timing-parameters>. Citado na página 68.

COOK, P. *N4BiasFieldCorrection*. 2021. Disponível em: <https://github.com/ANTsX/ANTs/wiki/N4BiasFieldCorrection>. Citado na página 69.

COURTS, U. F. hiQ Labs, Inc. v. Linkedin Corporation, No. 17-16783. [s.n.], 2022. Disponível em: <https://law.justia.com/cases/federal/appellate-courts/ca9/17-16783/17-16783-2022-04-18.html>. Citado na página 27.

DEVELOPERS, T. fMRIPrep. *fMRIPrep - Usage Notes*. 2016. Disponível em: <https://fmriprep.org/en/stable/usage.html>. Citado na página 66.

DEVELOPERS, T. nilearn. *Schaefer 2018 Atlas*. Disponível em: <https://nilearn.github.io/dev/modules/description/schaefer_2018.html>. Citado na página 68.

DEVLIN, J. et al. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. Disponível em: <http://arxiv.org/abs/1810.04805>. Citado na página 34.

DEVLIN, J. et al. Bert: Bidirectional encoder representations from transformers. *arXiv preprint arXiv:1810.04805*, p. 15, 2018. Citado na página 25.

DLOTKO, P. *Cubical complex user manual*. Disponível em: <https://gudhi.inria.fr/python/latest/cubical_complex_user.html>. Citado na página 70.

DOUAUD, G. et al. Integration of structural and functional magnetic resonance imaging in amyotrophic lateral sclerosis. *Brain*, v. 134, n. Pt 12, p. 3470–3479, December 2011. Epub 2011 Nov 10. Citado na página 59.

ELAMIN, M. et al. Predicting prognosis in amyotrophic lateral sclerosis: A simple algorithm. *Journal of Neurology*, v. 262, n. 6, p. 1447–1454, June 2015. Epub 2015 Apr 11. Citado 2 vezes nas páginas 22 and 85.

ESTEBAN, O. et al. fmriprep: a robust preprocessing pipeline for functional mri. *Nature Methods*, v. 16, n. 1, p. 111–116, Jan 2019. Disponível em: <https://doi.org/10.1038/s41592-018-0235-4>. Citado 2 vezes nas páginas 66 and 67.

FANNI, S. C. et al. Natural language processing. In: *Introduction to Artificial Intelligence*. [S.l.]: Springer, 2023. p. 87–99. Citado 2 vezes nas páginas 24 and 26.

FELDMAN, E. L. et al. Amyotrophic lateral sclerosis. *Lancet*, v. 400, n. 10360, p. 1363–1380, October 2022. Epub 2022 Sep 15. Citado 2 vezes nas páginas 19 and 21.

FERRULLI, V. *PyPaperBot*. Disponível em: <https://github.com/ferru97/PyPaperBot>. Citado na página 31.

FLASK. *User's Guide*. Disponível em: <https://flask.palletsprojects.com/en/latest/>. Citado na página 38.

GIOTTO-TDA - CubicalPersistence. 2021. Disponível em: <https://giotto-ai.github.io/gtda-docs/latest/modules/generated/homology/gtda.homology.CubicalPersistence.html>. Citado na página 70.

GIOTTO-TDA - gtda.diagrams: Persistence Diagrams. 2021. Disponível em: <https://giotto-ai.github.io/gtda-docs/0.5.1/modules/diagrams.html>. Citado na página 70.

GOPENAI. *Understanding PoS Tagging: An In-Depth Exploration*. 2023. Disponível em: <https://blog.gopenai.com/understanding-pos-tagging-an-in-depth-exploration-747f981d3514>. Citado na página 25.

GROLEZ, G. et al. The value of magnetic resonance imaging as a biomarker for amyotrophic lateral sclerosis: A systematic review. *BMC Neurology*, v. 16, n. 1, p. 155, August 2016. Citado na página 59.

GUPTA, A. et al. Accuracy of conventional mri in als. *Canadian Journal of Neurological Sciences*, v. 41, n. 1, p. 53–57, January 2014. Citado na página 59.

HARDIMAN, O. et al. Amyotrophic lateral sclerosis. *Nature Reviews Disease Primers*, v. 3, p. 17071, October 2017. Erratum in: Nat Rev Dis Primers. 2017 Oct 20;3:17085. doi: 10.1038/nrdp.2017.85. Citado 3 vezes nas páginas 19, 20, and 23.

HAYSTACK. *deepset/roberta-base-squad2*. Disponível em: <https://huggingface.co/deepset/roberta-base-squad2>. Citado na página 34.

HD-BET. Disponível em: <https://github.com/MIC-DKFZ/HD-BET>. Citado na página 69.

HOTHORN, T.; JUNG, H.-H. Randomforest4life: A random forest for predicting als disease progression. *Amyotrophic Lateral Sclerosis and Frontotemporal Degeneration*, v. 15, n. 5-6, p. 444–452, September 2014. Citado 2 vezes nas páginas 21 and 85.

HTTPX. A next-generation HTTP client for Python. Disponível em: <https://www.python-httpx.org/>. Citado na página 31.

ILIN, I.; KELLI, A. Natural language, legal hurdles: Navigating the complexities in natural language processing development and application. *Journal of the University of Latvia. Law*, v. 17, p. 44–67, Oct. 2024. Disponível em: <https://journal.lu.lv/jull/article/view/2064>. Citado na página 27.

JONES, J. et al. What do we know about hugging face? a systematic literature review and quantitative validation of qualitative claims. In: *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: Association for Computing Machinery, 2024. (ESEM '24), p. 13–24. ISBN 9798400710476. Disponível em: <https://doi.org/10.1145/3674805.3686665>. Citado na página 26.

KOCAR, T. D. et al. Feature selection from magnetic resonance imaging data in als: A systematic review. *Therapeutic Advances in Chronic Disease*, v. 12, p. 20406223211051002, October 13 2021. Citado na página 59.

KUSHOL, R. et al. Sf2former: Amyotrophic lateral sclerosis identification from multi-center mri data using spatial and frequency fusion transformer. *Computerized Medical Imaging and Graphics*, v. 108, p. 102279, 2023. ISSN 0895-6111. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0895611123000976>. Citado 2 vezes nas páginas 23 and 85.

LILLE Neuroscience & Cognition. Disponível em: <http://lilncog.eu/en/lille-neuroscience-and-cognition/>. Citado na página 58.

LIPTON, V. Legal issues arising in open scientific data. In: LIPTON, V. J. (Ed.). *Open Scientific Data*. Rijeka: IntechOpen, 2020. cap. 10. Disponível em: <https://doi.org/10.5772/intechopen.91713>. Citado na página 27.

LUO, X. et al. Large language models surpass human experts in predicting neuroscience results. *Nature Human Behaviour*, nov. 2024. ISSN 2397-3374. Disponível em: <https://www.nature.com/articles/s41562-024-02046-9>. Citado na página 49.

MARIA, C. et al. *Rips complex user manual*. Disponível em: <https://gudhi.inria.fr/python/latest/rips_complex_user.html>. Citado na página 70.

MUNCH. *Teaspoon - Featurization*. 2020. Disponível em: <https://teaspoontda.github.io/teaspoon/F_PD.html>. Citado na página 70.

MURPHY, K.; BIRN, R. M.; BANDETTINI, P. A. Resting-state fmri confounds and cleanup. *NeuroImage*, v. 80, p. 349–359, October 15 2013. Epub 2013 Apr 6. Citado 2 vezes nas páginas 59 and 68.

MUTHUKADAN, B. *Selenium with Python*. Disponível em: <https://selenium-python.readthedocs.io/>. Citado na página 31.

NADKARNI, P. M.; OHNO-MACHADO, L.; CHAPMAN, W. W. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, BMJ Group BMA House, Tavistock Square, London, WC1H 9JR, v. 18, n. 5, p. 544–551, 2011. Citado 2 vezes nas páginas 25 and 26.

NEUROMINE. Disponível em: <https://dataportal.answerals.org/home>. Citado na página 24.

NICHOLS, T. *FAST*. 2020. Disponível em: <https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FAST>. Citado na página 69.

PYPI. *fuzzywuzzy*. Disponível em: <https://pypi.org/project/fuzzywuzzy/>. Citado na página 36.

PYPI. *pypdf*. Disponível em: <https://pypi.org/project/pypdf/>. Citado na página 31.

PYPI. *Cloudscraper*. 2023. Disponível em: <https://pypi.org/project/cloudscraper/>. Citado na página 31.

SAGGAR, M. et al. Towards a new approach to reveal dynamical organization of the brain using topological data analysis. *Nature Communications*, v. 9, p. 1399, 2018. Disponível em: <https://doi.org/10.1038/s41467-018-03664-4>. Citado na página 62.

SALNIKOV, V.; CASSESE, D.; LAMBIOTTE, R. Simplicial complexes and complex systems. *European Journal of Physics*, v. 40, n. 1, p. 014001, 2019. Disponível em: <https://iopscience.iop.org/article/10.1088/1361-6404/aae790/meta#ejpaae790f2>. Citado na página 60.

SANH, V. et al. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In: *NeurIPS EMC^2 Workshop*. [S.l.: s.n.], 2019. Citado na página 34.

SAUL, N. *Representations manual*. 2019. Disponível em: <https://persim.scikit-tda.org/en/latest/>. Citado na página 70.

SCHAEFER, A. et al. Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity mri. *Cerebral Cortex*, v. 28, n. 9, p. 3095–3114, Sep 2018. Disponível em: <https://doi.org/10.1093/cercor/bhx179>. Citado na página 68.

SCHUSTER, C.; HARDIMAN, O.; BEDE, P. Survival prediction in amyotrophic lateral sclerosis based on mri measures and clinical characteristics. *BMC Neurology*, v. 17, p. 73, 2017. Citado 2 vezes nas páginas 22 and 23.

SICILIANO, M. et al. Edinburgh cognitive and behavioural als screen (ecas)-italian version: regression based norms and equivalent scores. *Neurological Sciences*, v. 38, n. 6, p. 1059–1068, Jun 2017. Disponível em: <https://doi.org/10.1007/s10072-017-2919-4>. Citado na página 64.

SKAF, Y.; LAUBENBACHER, R. Topological data analysis in biomedicine: A review. *Journal of Biomedical Informatics*, v. 130, p. 104082, 2022. ISSN 1532-0464. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1532046422000983>. Citado 2 vezes nas páginas 60 and 62.

SOUZA, A.; SCHIRRU, L.; ALVARENGA, M. Mineração de textos e dados na pesquisa em saúde: reflexões sobre direitos autorais. *Cadernos de Saúde Pública*, v. 40, 05 2024. Citado na página 28.

ST-ONGE, F. *fMRIPrep pre-processing and post-processing - connectivity matrices extraction using a parcellation*. Disponível em: <https://github.com/brainhack-school2020/stong3_fMRI_processing/blob/master/fMRIPrep_tutorial/fMRIPrep%20pre-processing%20and%20post-processing%20-%20connectivity%20matrices%20extraction%20using%20a%20parcellation.ipynb>. Citado na página 68.

STEINBACH, R. et al. Developing a neuroimaging biomarker for amyotrophic lateral sclerosis: Multi-center data sharing and the road to a "global cohort". *Frontiers in Neurology*, v. 9, p. 1055, December 4 2018. Citado na página 16.

SUDO, T.; AHARA, K. *CubicalRipser: Persistent homology for 2D image and 3D voxel data (and 1D scalar timeseries)*. 2018. Disponível em: <https://github.com/shizuo-kaji/CubicalRipser_3dim>. Citado na página 70.

SWINNEN, B.; ROBBERECHT, W. The phenotypic variability of amyotrophic lateral sclerosis. *Nature Reviews Neurology*, v. 10, p. 661–670, 2014. Disponível em: <https://doi.org/10.1038/nrneurol.2014.184>. Citado na página 20.

TANG, M. et al. Model-based and model-free techniques for amyotrophic lateral sclerosis diagnostic prediction and patient clustering. *Neuroinformatics*, v. 17, n. 3, p. 407–421, July 2019. Citado 2 vezes nas páginas 21 and 85.

TARASYUK, A.; TROUBITSYNA, E.; LAIBINIS, L. From formal specification in event-b to probabilistic reliability assessment. *Dependability, International Conference on*, v. 0, p. 24–31, 07 2010. Citado na página 33.

TAVAZZI, E. et al. Artificial intelligence and statistical methods for stratification and prediction of progression in amyotrophic lateral sclerosis: A systematic review. *Artificial Intelligence in Medicine*, v. 142, p. 102588, 2023. ISSN 0933-3657. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0933365723001021>. Citado na página 22.

TDA with Python using the Gudhi Library. Representing sublevel sets of functions using cubical complexes. 2021. Disponível em: <https://github.com/GUDHI/TDA-tutorial/commits/master/Tuto-GUDHI-cubical-complexes.ipynb>. Citado na página 71.

THESSEN, A.; CUI, H.; MOZZHERIN, D. Applications of natural language processing in biodiversity science. *Advances in bioinformatics*, v. 2012, p. 391574, 05 2012. Citado na página 25.

van der Burgh, H. K. et al. Deep learning predictions of survival based on mri in amyotrophic lateral sclerosis. *NeuroImage: Clinical*, v. 13, p. 361–369, 2017. ISSN 2213-1582. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2213158216301899>. Citado 3 vezes nas páginas 22, 23, and 85.

WASSERMAN, L. Topological data analysis. *Annual Review of Statistics and Its Application*, Annual Reviews, v. 5, n. Volume 5, 2018, p. 501–532, 2018. ISSN 2326-831X. Disponível em: <https://www.annualreviews.org/content/journals/10.1146/annurev-statistics-031017-100045>. Citado 2 vezes nas páginas 61 and 62.

WROE, R. et al. Alsod: the amyotrophic lateral sclerosis online database. *Amyotrophic Lateral Sclerosis: Official Publication of the World Federation of Neurology Research Group on Motor Neuron Diseases*, v. 9, n. 4, p. 249–250, 2008. Citado na página 24.

YEO, B. et al. The organization of the human cerebral cortex estimated by intrinsic functional connectivity. *Journal of Neurophysiology*, v. 106, n. 3, p. 1125–1165, Sep 2011. Disponível em: <https://doi.org/10.1152/jn.00338.2011>. Citado na página 68.

ZIPSE, L. et al. When right is all that is left: Plasticity of right-hemisphere tracts in a young aphasic patient. *Annals of the New York Academy of Sciences*, v. 1252, p. 237–45, 04 2012. Citado na página 59.

Appendix

A Previous Work

A.1 Introduction

Collaboration Site

The Lille Neuroscience and Cognition Center, led by Doctor L. Buée, comprises five INSERM-accredited teams in partnership with the University of Lille and the CHU of Lille. For the first steps of my work, I was part of the Degenerative and Vascular Cognitive Disorders team, led by Professor D. Devos, which consists of both clinical and pre-clinical researchers. This team combines neurology, pharmacology, and imaging to focus on the translational, transnosographic, and multimodal study of cognitive disorders associated with degenerative processes or neurovascular lesions. For more information about the Lille Neuroscience & Cognition Center, please refer to their website (LILLE. . . ,).

In this context, my thesis involved developing mathematical analysis on Magnetic Resonance Imaging (MRI) as a potential prognostic biomarker for Amyotrophic Lateral Sclerosis (ALS), a neurodegenerative disease with no curative treatment. I thus developed pipelines that included image processing, Topological Data Analysis (TDA), and Machine Learning (ML) techniques. The work required interdisciplinary expertise involving specialists in imaging, clinical research, and mathematics. Throughout this part of the project, I was supervised by Dr. AS. Rolland and Dr. A. Broustet, and I collaborated closely with my partner, Nathan Lesourd.

Magnetic Resonance Imaging

Magnetic Resonance Imaging technology relies on the application of a large magnetic field to induce resonance in certain atoms within the patient and the mechanisms and measurements vary according to the imaging modality. Among the different MRI techniques are structural MRI (T1-weighted), diffusion tensor imaging, magnetic resonance spectroscopy, iron-sensitive sequences (T2*, R2*, and SWI), and functional MRI (fMRI). During my internship, as I focus on T1-weighted (T1w) imaging and resting-state functional MRI (rsfMRI), I will briefly describe these two.

T1-weighted images correspond to structural MRI and reflect the physical organization of the brain when applied in a neurological context. To generate these images, the magnetization vector created by the magnetic moments of hydrogen atoms in the brain is measured as an applied radiofrequency pulse varies. Specifically for T1-weighted imaging, the signal of interest is the magnetization in the longitudinal plane, which increases over time after the induced magnetic field is stopped, a process known as relaxation. This relaxation differs for each tissue based on their water and fat concentration, allowing for anatomical differentiation of brain regions (Conti, 2016 (CONTI, 2016)).

In contrast, rsfMRI relies on blood oxygen level and is called “resting-state” as the patient does not have to perform any specific task during data collection. The measured signal is the Blood-Oxygen Level Dependent (BOLD) signal, which is based on the principle that areas with neural activity consume more oxygen, increasing deoxyhaemoglobin concentration and reducing the BOLD signal (Murphy et al., 2013(MURPHY; BIRN; BANDETTINI, 2013)).The goal of functional MRI is to map brain activity and statistically determine which regions are activated together and thus likely connected. Functional images are composed of several slices taken at different times, making the process highly sensitive to any patient movement. Figure 12 shows examples of T1-weighted and fMRI images side by side.

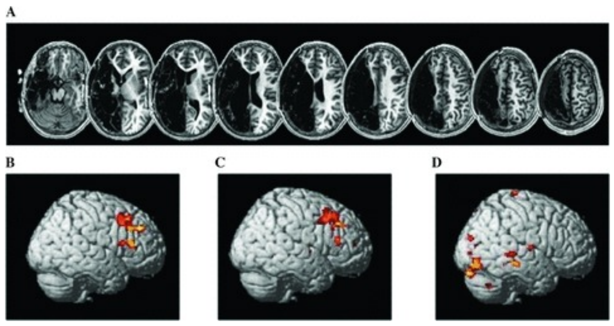


Figure 12 – **Comparison between T1 and functional MRI.**

(A)T1-weighted MRI; (B, C, D) fMRI. The areas marked in yellow indicate greater activation. *Image extracted from Zipse et al. (2012)* (ZIPSE et al., 2012).

Various analyses have been conducted on the use of MRI as a biomarker for ALS. According to Gupta et al. (2014) (GUPTA et al., 2014), conventional MRI is not yet a reliable indicator of the disease, remarking the need for definitive biomarkers in the early stages of ALS, which may be identified through newer neuroimaging strategies, such as diffusion tensor imaging and functional MRI.

Additionally, studies by Douaud et al. (2011) (DOUAUD et al., 2011) and Kocar et al. (2021) (KOCAR et al., 2021) have shown that structural and functional images exhibit alterations in ALS patients, including loss of grey matter volume in the precentral gyrus and either increased or decreased functional connectivity in the precentral and postcentral gyri. Douaud et al. (2011) (DOUAUD et al., 2011) noted that increased functional connectivity was observed in brain regions with decreased structural connectivity. Grolez et al. (2016) (GROLEZ et al., 2016) concluded that MRI patterns correlate with disease severity, progression, and duration, although the use of neuroimaging is still limited by the small number of participants and longitudinal studies.

Topological Data Analysis

Topological Data Analysis (TDA) differs from traditional data analysis by considering the spatial arrangement of data. Topology, in this context, refers to the mathematical study of properties preserved through continuous deformations such as stretching or bending. TDA aims to summarize and visualize complex data by incorporating notions of shape and connectivity, revealing characteristics that may not be apparent in raw data.

Among the methods used in topological analysis, the most common is persistent homology. This technique uses abstract algebra to detect topological features like connected components (dimension 0 or H_0), holes (dimension 1 or H_1), and loops (dimension 2 or H_2), determining whether objects are homologous based on shared features.

To compute persistence, the data is first stored in a simplicial complex, a set of simplices that adhere to the following rules:

- Every face of a simplex in a complex is in the complex.
- The non-empty intersection of two simplices is a face of each.

Following Salnikov et al. (2019) (SALNIKOV; CASSESE; LAMBIOTTE, 2019), a simplex is a convex hull of affinely independent points, with faces that are simplices based on subsets of points. Geometrically, n-simplices are illustrated as shown in figure 13. Simplicial complexes enable higher-dimensional analogs of edges between more than two points, unlike standard networks.

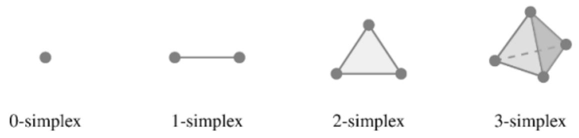


Figure 13 – **Illustration of n-dimensional simplexes.**

Image extracted from Skaf and Laubenbacher (2022) (SKAF; LAUBENBACHER, 2022).

Different simplicial complex implementations have specific applications, that is, the choice of complex depends on the application. The Vietoris-Rips complex V_ϵ, composed of point subsets with a diameter smaller than a threshold ϵ, is frequently used. The cubical complex, which uses cubes to compute homology, fits the digital image structure of grids (Choe and Ramanna, 2022 (CHOE; RAMANNA, 2022)) and is an alternative for processes requiring significant computational resources.

Once data is represented by a simplicial complex, persistence can be calculated for different dimensions. Figure 14 exemplifies the computation in dimensions 0 (H_0) and 1 (H_1). Balls created at each data point grow in radius r. At $r = 0$, there are n disconnected balls. As r increases, connected components merge, which is considered as their death, and eventually all overlap. For the image in figure 14 most to the right, it is possible to observe a hole that is born. By reaching even larger values of r, the hole will also die.

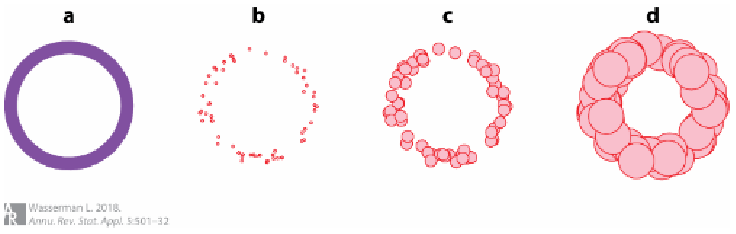


Figure 14 – **Representation of the intuitive idea behind persistent homology.**

Image extracted from Wasserman (2018) (WASSERMAN, 2018).

Persistence is accordingly represented by birth and death coordinates, illustrated in barcode plots and persistence diagrams exemplified by the figure 15. In the barcode plot, the connected components correspond to the gray lines and the hole to the red line. Each barcode represents the moment in time where birth and death occur. In the persistence diagram, the connected components correspond to the blue points and the hole to the red triangle, which are placed according to the x-axis (birth) and to the y-axis (death).

The features that are considered the most relevant to characterize the data are those that survive longer and, therefore, are the ones that are most consistent over different scales. The ones that persist only for a short period, disappearing quickly, may be classified as noise or artifacts. In barcode plots, the survival of each feature is determined by the length of its barcode, while in persistence diagrams what has to be observed is the distance of the point to the diagonal, where death is equal to birth and survival is null.

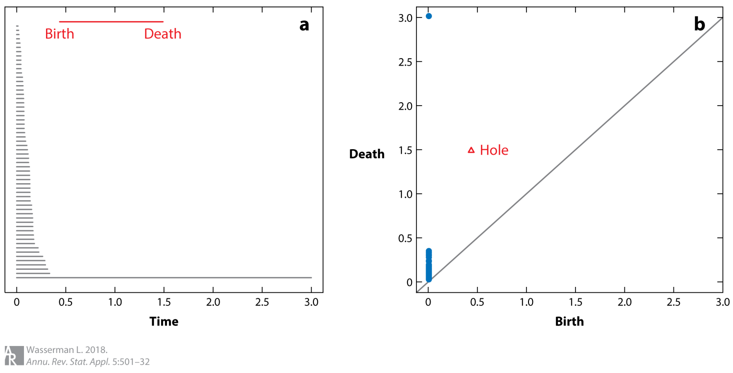


Figure 15 – **Comparison between barcode plot and persistence diagram.**

On the left, the barcode plot calculated for the object represented in the figure 14. On the right, the persistence diagram. *Image extracted from Wasserman (2018)* (WASSERMAN, 2018).

However, the resulting persistences cannot be directly input to Machine Learning models, due to undefined basic operations, including addition and multiplication (Carrière et al., 2020 (CARRIèRE et al., 2020)). Thus, as a final step of the TDA, it is necessary to vectorize the persistence diagrams before performing model training. Two options for vectorization explored throughout the present work were persistence landscapes (PL) and persistence images (PI).

Persistence landscapes map persistence diagrams to a Hilbert space using exact or discrete approximations. PL may be defined by a sequence of functions λ_1, λ_2, ...: $IR \to IR$ where each function λ_k is piecewise linear with slope either 0,1, or -1. Their advantages include invertibility, that is, loss of none or little information, stability, being parameter-free and nonlinear and subsequent fast machine learning computations (Bubenik, 2020 (BUBENIK, 2020)). Nonetheless, it may be a large representation. The resulting landscape has the format shown by the figure 16.

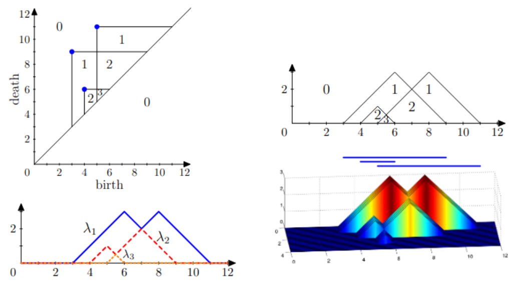


Figure 16 – **Example of the Persistence Landscape representation.**

Image extracted from Bubenik (2015) (BUBENIK, 2015).

Persistence images map persistence diagrams to integrable functions ρ_B, then calculate the integral for each grid box, offering flexibility in feature relevance. Also, when tested for a classification task, PI distance matrices performed well in terms of accuracy and of processing time (Chepushtanova et al., 2015 (CHEPUSHTANOVA et al., 2015)). The final visualization of the persistence image is represented by the figure 17.

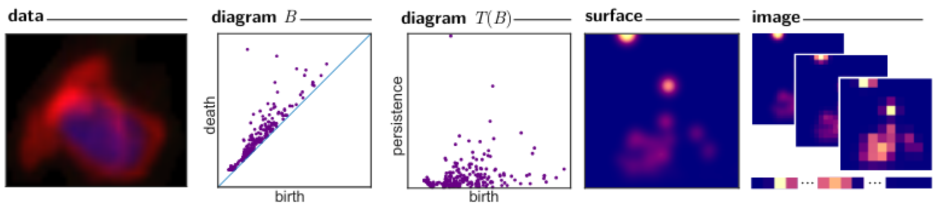


Figure 17 – **Example of the Persistence Image representation.**

Image extracted from Chepushtanova et al. (2015) (CHEPUSHTANOVA et al., 2015).

TDA has been applied across various domains, including the cosmic web, image analysis, and protein structure characterization (Wasserman, 2018 (WASSERMAN, 2018)). In the biomedical field, topological analysis has been used for patient subtyping in diseases like diabetes, aortic stenosis, asthma, oncology, and brain injuries, as well as in medical imaging (CT, MRI, histology) (Skaf and Laubenbacher, 2022 (SKAF; LAUBENBACHER, 2022)). Specific implementations for functional MRI and Cubical Complexes in image classification have also been explored (Choe and Ramanna, 2022 (CHOE; RAMANNA, 2022), Saggar et al., 2018 (SAGGAR et al., 2018)).

In conclusion, TDA has demonstrated significant potential in the diagnosis and prognosis of human diseases and in the analysis of medical imaging, including magnetic resonance imaging. The promising results from the cited studies support the hypothesis that TDA could play a crucial role in identifying biomarkers in MRI data collected from ALS patients.

A.2 Objectives

The primary objective of the first part of the project was to uncover imaging biomarkers to monitor disease progression using longitudinal data from a cohort of newly diagnosed ALS patients. Specifically, my goal was to study the efficiency of applying Topological Data Analysis to MRI data and determine whether it conducted to a finer imaging analysis or not. In order to evaluate the resulting precision, different Machine Learning were to be trained for classifying or predicting disease progression. The images used to feed our model were collected through a collaborative effort led by Lille University Hospital, detailed further in section A.3.

To achieve this objective, the following steps were proposed, as illustrated by the figure 18.

- **Preprocessing:** This step involves preparing the images for further analysis, including corrections, denoising, normalization, and other processes. The specific preprocessing steps depend on the image modality, as different acquisition mechanisms and representations require different treatments.

- **Topological Data Analysis:** As discussed in section A.1, TDA involves calculating persistence diagrams for the preprocessed data for each patient and converting them into representation objects for use in the next step.

- **Model Training:** To classify or predict ALS progression in patients, various models had to be trained using Machine Learning techniques. The final models' performance is evaluated and presented in section A.4.

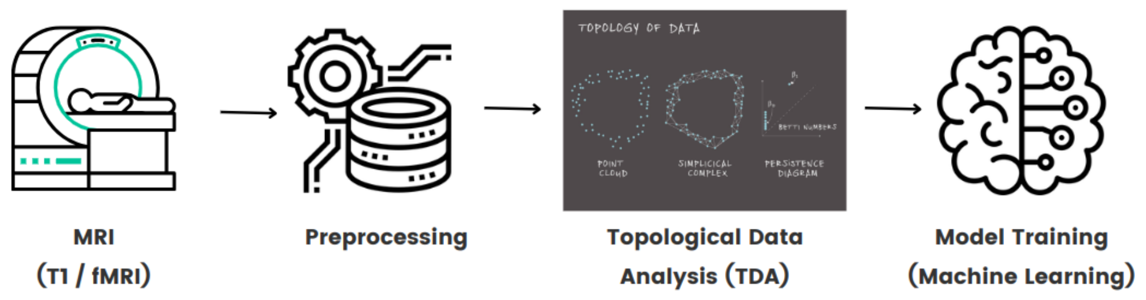


Figure 18 – **General pipeline aimed by the study.**

It is also fundamental that the whole pipeline is reproducible to assure the validation of the results, robust, portable and well-documented for future maintenance.

A.3 Methodologie

Dataset

The PULSE cohort was created to provide a national French dataset for studying predictive factors in ALS evolution. Sponsored by the Lille University Hospital, the project was coordinated by Pr. David Devos. Seventeen ALS expert centers contributed with longitudinal data on newly diagnosed patients.

The collected data included age, sex, weight, treatments, date of first symptom, onset site, ALSFRS-R score, ECAS score (Edinburgh Cognitive and Behavioural ALS Screen) (Siciliano et al., 2017 (SICILIANO et al., 2017)), blood composition, genetic forms in familial cases, MRI images, etc. For prediction analysis, data were gathered not only at inclusion but also every three months up to 36 months. The cohort also included three control groups: a healthy group with blood, a healthy group with MRI, a neurological group affected by other neurodegenerative diseases. The criteria for patient and control inclusion in the PULSE cohort is shown in the table B in the Appendix.

As a multicentric dataset, PULSE includes data from various cities in France, such as Lille, Paris, Tours, Saint-Étienne, Brest, Lyon, Montpellier, Marseille, Nice, Nancy, Clermont-Ferrand, Caen, St-Brieuc, and Angers. MRI equipment varied across centers, potentially causing heterogeneity despite a standardized protocol.

The Image Acquisition and automated Treatment Center (CATI) was responsible for MRI quality control, ensuring imaging quality, absence of artifacts, correct head positioning, and consistent imaging parameters.

From the 463 participants included in the cohort, I had access to data from 357 PULSE subjects for developing the pipeline. All subjects had MRI collected at inclusion

(M000), but subsequent sessions (M003, M006, M012) were optional. Consequently, only 19, 33, and 16 patients had MRI available for 3, 6, and 12 months, respectively. Additionally, some resting-state images were deemed unexploitable during quality control and excluded from analysis. Clinical data could also be lacking and some participants were not ALS patients but served as healthy or neurological controls. The flowchart in Figure 19 illustrates each step of the data selection process until model training.

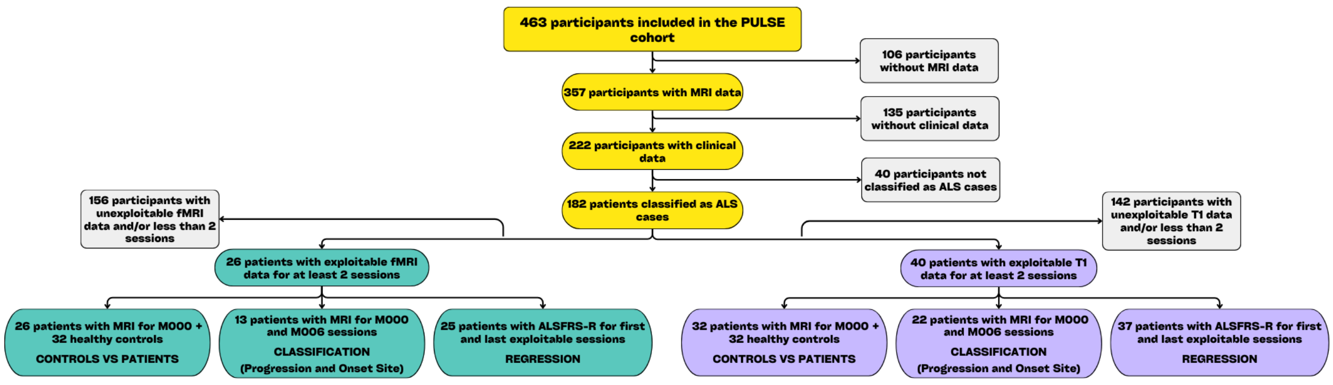

463 participants included in the PULSE cohort
106 participants without MRI data
357 participants with MRI data
135 participants without clinical data
222 participants with clinical data
40 participants not classified as ALS cases
182 patients classified as ALS cases
156 participants with unexploitable fMRI data and/or less than 2 sessions
142 participants with unexploitable T1 data and/or less than 2 sessions
26 patients with exploitable fMRI data for at least 2 sessions
40 patients with exploitable T1 data for at least 2 sessions
26 patients with MRI for M000 + 32 healthy controls
CONTROLS VS PATIENTS
13 patients with MRI for M000 and M006 sessions
CLASSIFICATION
(Progression and Onset Site)
25 patients with ALSFRS-R for first and last exploitable sessions
REGRESSION
32 patients with MRI for M000 + 32 healthy controls
CONTROLS VS PATIENTS
22 patients with MRI for M000 and M006 sessions
CLASSIFICATION
(Progression and Onset Site)
37 patients with ALSFRS-R for first and last exploitable sessions
REGRESSION

Figure 19 – **Flowchart showing data selection from PULSE for the developed pipelines.**

Using a Python script and an Excel file provided by CATI, which classified each MRI as unexploitable, borderline, or perfect, I evaluated which patients could be analyzed. For a longitudinal study, only patients with exploitable MRI from at least two sessions were selected. Based on this criterion, 26 patients qualified for the rsfMRI pipeline (having exploitable T1 and rsfMRI for at least two sessions), and 40 patients qualified for the T1 pipeline (having exploitable T1 for at least two sessions).

Execution Environment

The pipelines were tested and executed on a server provided by the University of Lille. The server runs on an Intel(R) Xeon(R) Silver 4116 CPU operating at 2.10 GHz. This processor has an x86_64 architecture and supports 2 threads per core. Each socket contains 12 cores, and there are 2 sockets available on the machine. The operating system installed on the server is Ubuntu 22.04.4 LTS, with the kernel version being Linux 5.15.0-102-generic. The server is equipped with a total of 128 GiB of RAM, providing ample memory for running computationally intensive tasks.

However, it is important to note that the server does not offer GPU support. As a result, certain algorithms that could benefit from GPU acceleration were not optimized during execution. Despite this, the server's significant CPU and RAM resources were adequate for executing the developed pipelines and performing the required TDA and machine learning analyses for ALS patient data.

Pipelines

I developed two distinct pipelines: one for rsfMRI and one for T1 images. The preprocessing steps varied according to the modality being analyzed, which in turn influenced the topological data analysis conducted for each pipeline. However, the model training process at the end was the same for both pipelines. Therefore, the following sections describe the preprocessing and TDA separately for each modality, while the Machine Learning method is explained jointly for both fMRI and T1.

Preprocessing

rsfMRI

Functional MRI is a modality that is sensitive to non-neural sources of variability, demanding specific image treatment prior to further analysis, as mentioned in section A.1. Corrections, such as slice-timing correction (STC), head-motion correction (HMC), and susceptibility distortion correction (SDC), address artifacts, while co-registration and spatial normalization tackle anatomical localization of signals. These preprocessing steps are crucial to ensure that the final outputs of the pipeline reflect neural activity accurately and are therefore valid for further analysis.

Aiming to provide a robust and convenient tool that includes all the fundamental steps of fMRI preprocessing, fMRIPrep was developed, as described in Esteban et al. (2019) (ESTEBAN et al., 2019). Recommended by Dr. Cecile Bordier, it is currently considered the standard and most reliable workflow for this task. Additional usage notes and documentation can be found on their website (DEVELOPERS, 2016).

The steps included in fMRIPrep are illustrated in figure 20. Both T1-weighted and functional images are supplied as input. T1 images undergo intensity non-uniformity correction (N4BiasFieldCorrection) and skull-stripping (antsBrainExtraction). Subsequently, spatial normalization through nonlinear registration (antsRegistration) and brain tissues segmentation (CSF, White matter (WM), and Grey Matter (GM)) are performed. Surface reconstruction and mask refinement are optional. For fMRI, the pipeline includes estimation of head-motion parameters (mcflirt), STC (3dTshift), and SDC. Finally, the image is co-registered to the corresponding T1 reference.

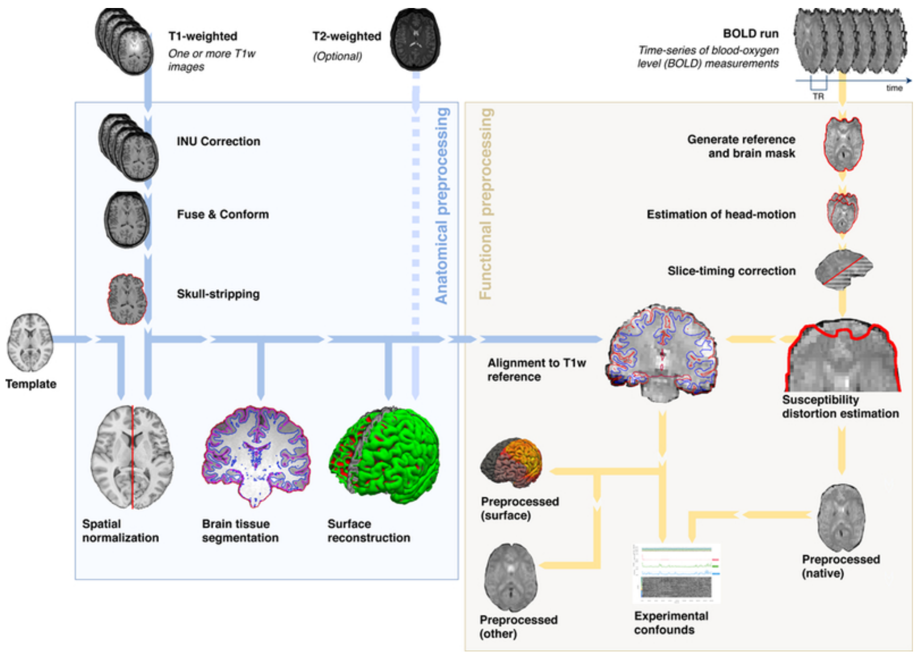


Figure 20 – **Workflow applied by fMRIPrep for preprocessing fMRI.**

Image extracted from Esteban et al. (2019) (ESTEBAN et al., 2019).

To run fMRIPrep, the following flags were used:

- **random-seed 42**: Initializes the random seed for the workflow as 42.
- **omp-nthreads 1**: Declares the maximum number of threads per-process as equal to one.
- **skull-strip-fixed-seed**: When declared along with the previous flag, ensures that skull-stripping does not use a random seed, making the process replicable for future validation.

A FreeSurfer license is required to execute fMRIPrep, and the directory structure must follow the BIDS (Brain Imaging Data Structure) specifications. While PULSE already followed the correct structure, it included files (html, logs, figures, DWI, fmap, tsv files, etc.) that should not be considered during fMRI preprocessing. To ignore them and ensure the pipeline functions correctly, a .bidsignore file was defined as follows:

```
*.html
logs/
figures/
*_xfm.*
*.surf.gii
*_boldref.nii.gz
*_bold.func.gii
```

```
**/*dwi*
**/*fmap*
*.tsv
```

Additionally, each image to be preprocessed had a json file containing the parameters used during data collection. For performing Slice-Timing Correction, it was necessary that this file included the "SliceTiming" information, which corresponds to the time each slice was acquired within each volume (frame) of the acquisition (CONTRIBUTORS,). Some files lacked this specification, so it was added using a bash script, attached in the Appendix F.

The outputs from fMRIPrep include the preprocessed functional images and the estimated confounds. Confounds consist of fluctuations with a potential non-neuronal origin, which may yield spurious results in functional connectivity. Examples include motion, cardiac and respiratory cycles, arterial CO_2 concentration, blood pressure/cerebral autoregulation, and vasomotion (Murphy et al., 2013 (MURPHY; BIRN; BANDETTINI, 2013)). To clean the images, I used the output estimates from fMRIPrep in confound regression, implemented in Python following the tutorial in (CLEANING..., 2024).

Based on the work by Yeo et al. (2011) (YEO et al., 2011), the selected confound regressors were: 6 motion parameters (trans_x, trans_y, trans_z, rot_x, rot_y, rot_z), global signal, cerebral spinal fluid signal and white matter signal. Unlike the tutorial, derivatives of these confound regressors were not included to avoid losing too much information in the regression. The first four time points of the images were discarded due to a strong signal decay artifact at the initial stages of the scan. A low/high pass filter was applied to eliminate high and low-frequency signals, which might be due to noise and intrinsic scanner instabilities.

After confound regression, connectivity matrices were calculated for each image. These matrices are constructed by calculating the connectivity between each pair of different regions of interest (ROIs) in the brain using preprocessed fMRI. The tutorial in (ST-ONGE,) served as the basis for this implementation. We chose the Schaefer atlas ((DEVELOPERS,), Schaefer et al., 2018 (SCHAEFER et al., 2018)), with 300 ROIs, providing a balance between having a significant number of regions and avoiding an overly demanding posterior analysis. Connectivity measurement was implemented in Python using the **ConnectivityMeasure** function from the NiLearn library, with correlation as the chosen connectivity measure.

Negative values in the resulting matrices, which range from -1 to 1, were considered as zero since the meaning of negative values for correlation between brain regions is not yet well understood.

fMRIPrep runs in a Docker container, so environment preparation regarding installations was unnecessary except for pulling its image. A Singularity container was defined with all necessary Python libraries for the steps following image preprocessing, using a .def file (attached in the Appendix). For building the Singularity images, I created a .yml file as shown below. It was then sufficient to run the command **singularity-compose build** in the folder containing this file.

```
version: "1.0"
instances:
  fmriprep:
    image: docker://nipreps/fmriprep:latest
  postprocessing:
    build:
      context: ./postprocessing
      recipe: container.def
    depends_on:
     - fmriprep
```

In summary, the preprocessing of functional imaging involved fMRIPrep for corrections and spatial normalization, confound regression, and the computation of connectivity matrices. I applied this procedure to all sessions presented by the 26 patients selected for the rsfMRI pipeline (as specified in section A.3) and for the 32 controls. In total, 94 matrices (62 for patients + 32 for controls) were calculated. All scripts referenced are attached in the Appendix Section.

T1

The T1 pipeline focuses on comparing subjects based on the anatomical structure of the brain. Consequently, the preprocessing steps for T1 images were designed to be minimal yet effective, ensuring that the output is clean and retains only relevant information.

Based on fMRIPrep, there were three tools that were applied to T1 images: N4BiasFieldCorrection (COOK, 2021) (ANTs) for intensity non-uniformity correction, HD-BET (HD-BET,) for skull-stripping and FAST (NICHOLS, 2020) (FSL) for tissue segmentation. Then, in Python, the input for the TDA was the selection of the voxels from the corrected and skull-stripped image that corresponded to WM and GM according to the FAST mask output. All of the required software was containerized in a singularity image defined by as .def file, specified in the Appendix section.

TDA

rsfMRI

To analyze connectivity matrices derived from rsfMRI, distance matrices are needed.

Consequently, we implemented the Vietoris-Rips complex by the Gudhi library (MARIA et al.,). To obtain the distance matrix, each value X in the connectivity matrix was converted to $\frac{1}{X}$. For connectivity values that were zero, they were replaced with the smallest computable positive value to avoid division by zero. This value, according to the numpy library, is approximately $2.2250738585072014 \times 10^{-308}$.

The parameter max_edge_length was established as 10^3, so only the edges of the graph smaller than the threshold are inserted, and then the complex was stored in a simplex tree structure, where max_dimension was equal to 3, referring to maximal graph expansion. Next, using the compute_persistence function, homology_coeff_field was defined as 2. This means that persistence was calculated solely for three dimensions (H_0, H_1 and H_2). Such choice balances computational feasibility and interpretability, avoiding the complexity and computational demands of higher dimensions. The results were stored in .npz files for efficient retrieval and to avoid re-computation.

Regarding vectorization, several alternatives were tested for Persistence Image and Persistence Landscape, offered by different Python libraries. The Table 4 explicits which libraries were tested for each representation.

Library	**PL tested?**	**PI tested?**
PersLay (CARRIèRE et al., 2020)	Yes	Yes
Gudhi (CARRIèRE et al.,)	Yes	Yes
Persim (SAUL, 2019)	No (output vectors with varying lenghts)	Yes for PersImage (PersistenceImager returned empty vectors for H0)
Teaspoon (MUNCH, 2020)	No (output vectors with varying lenghts)	No (implements Persim)
Giotto-TDA (GIOTTO-TDA..., 2021b)	Yes	Yes

Table 4 – **Libraries used for Persistence Landscapes and Persistence Images.**

PI: Persistence Images, PL: Persistence Landscape

T1

For the analysis of T1 images, which focus on directly examining the anatomical structure of the brain, cubical complexes were deemed more suitable than Vietoris-Rips complexes. We tested several libraries for constructing and analyzing cubical complexes, including Gudhi (DLOTKO,), Giotto-TDA (GIOTTO-TDA..., 2021a), and Cripser (SUDO; AHARA, 2018).

Different solutions were explored specifically for cubical complexes due to initial persistence diagrams displaying unexpected behavior, such as thousands of overlapping points. This prompted us to ensure that the issue did not stem from the algorithms provided by the libraries. Ultimately, Gudhi was selected for its consistent performance in generating uniform persistence diagrams, which aligned with the expected format. Therefore, the results presented in section A.4 are based on the use of the Gudhi library.

As explained in Bleile et al. (2021) (BLEILE et al., 2021), there are two different constructions that can be used for cubical complexes, named V-construction, which represents voxels by vertices, and T-construction, which represents voxels by top-dimensional cubes. In this work, T-construction was opted for as it is suggested by the tutorial in the Gudhi library (TDA..., 2021). For calculating the persistence, the parameters used were homology_coeff_field set to 2 and the minimum persistence value was the default, that is, zero. The computed persistence diagrams were stored in .npz files for efficient retrieval, similar to the rsfMRI pipeline.

Concerning the conversion from persistence diagrams to persistence representations, the exact same libraries that were tested for functional imaging were also used in this pipeline. The different results are compared in the section A.4.

Model Training

In developing pipelines for both rsfMRI and T1 imaging data, four main categories of Machine Learning models were tested to classify and predict the ALS disease progression using TDA vectors derived from neuroimaging data. The implementation in Python can be found in Appendix I and J.

1. **Categorizing healthy controls versus ALS patients**: Both clustering and supervised classification were employed to distinguish healthy controls from ALS patients. Clustering included applying Principal Component Analysis (PCA) to the vectors resulting from the TDA and both KMeans and Hierarchical Clustering for grouping the data. The silhouette and adjusted random scores were used to evaluate its performance for diagnosis. Supervised classification was tested for Support Vector Machine (SVM), Random Forest, KNeighbors and Logistic Regression from the sklearn library in Python. For evaluating such models, accuracy was measured.

2. **Classifying ALS patients as fast or slow progressors**: Using supervised classification along with the same models proposed for control vs. patient classification, I tested accuracy for segregating patients as fast or slow progressors. The input for the models were the concatenation of the data relatif to sessions M000 (inclusion) and M006 (6 months after inclusion) for the patients that presented both simultaneously. In this case, the progression of ALSFRS-R in time was the chosen metric for labeling the patients. To classify progression as fast or slow, the progression rate was obtained by using the calculation below.

$$\text{Progression rate (in month)} = \frac{48-\text{ALSFRS-R in M000}}{\text{Date of M000}-\text{Date of first symptom}}$$

$$\text{If progression rate} > 0.6\text{, patient classified as fast.}$$

$$\text{Otherwise, patient classified as slow.}$$

3. **Classifying ALS patients by the onset site of the disease**: The same supervised methods applied as described previously for classifying patients by their disease progression was also used for classifying them as having a bulbar or a spinal onset. It also used as input the concatenation of data relatif to M000 and M006.

4. **Regression for predicting ALSFRS-R delta between two sessions**: Using Linear, Gradient Boosting and Random Forest regression, all implemented by sklearn in Python, I tested applying regression to predict how the ALSFRS-R score would change between two sessions. As input, the data relatif to the first session and the last session found for each patient were concatenated, which allowed to use a broader range of patients. In this case, the coefficient of determination was the evaluation metric.

In all cases, cross validation with 5 iterations was used for evaluating the performance of the trained supervised models. Thus, the results shown in section A.4 present not only an average value for each accuracy/coefficient, but also the standard deviation between the iterations.

Detailed pipelines

To summarize, I implemented two pipelines, one for each MRI acquisition, with similarities but also differences regarding the preprocessing and the simplicial complex, as highlighted by figures 21 and 22. The goal was to uncover a prognostic biomarker to predict the disease progression, but also a diagnostic biomarker to detect the disease at early stages.

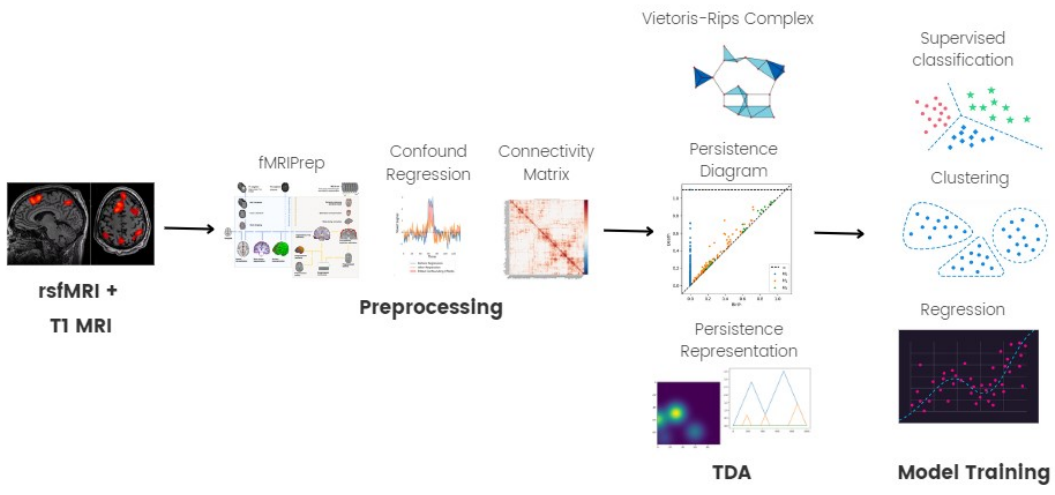


Figure 21 – **The rsfMRI pipeline**

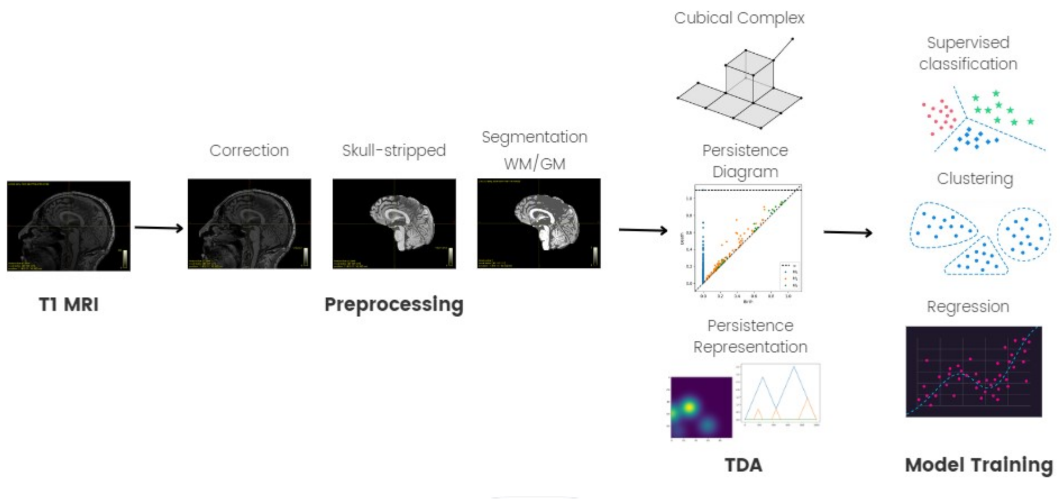


Figure 22 – **The T1 pipeline**

A.4 Results

Characterisation of the Cohort

The clinical characteristics of the cohort are summarized in Table 5, providing a general view of the patient population that participated in the study. This table differentiates the dataset as a whole, the subset included in the rsfMRI pipeline, and the subset included in the T1 pipeline.

	Total		rsfMRI		T1	
	N	Value	N	Value	N	Value
Population characteristics						
Male gender (%)	180	111 (61.7%)	26	14 (53.9%)	39	23 (59.0%)
Age (years)	180	62.0 ± 22.2	26	59.7 ± 8.4	39	60.5 ± 10.4
Time from first symptoms to inclusion (months)	164	14.0 (9.0 to 27.0)	25	20.0 (9.0 to 27.0)	38	20.5 (10.3 to 30.8)
Spinal phenotype (%)	180	138 (76.7%)	26	22 (84.6%)	39	32 (82.1%)
Bulbar phenotype (%)	180	42 (23.3%)	26	4 (15.4%)	39	7 (17.9%)
Clinical outcomes						
ALSFRS-R (/48)	171	38.9 ± 5.5	24	38.8 ± 6.2	36	38.9 ± 5.5
ALSFRS-R rate	158	0.5 (0.2 to 0.9)	23	0.5 (0.3 to 0.7)	35	0.4 (0.3 to 0.7)
Fast progressors (ALSFRS-R rate > 0.6) (%)	158	64 (40.5%)	23	9 (39.1%)	35	11 (31.4%)
ECAS (/136)	137	106.4 ± 18.5	25	110.2 ± 15.7	37	110.0 ± 16.7
Normal ECAS (ECAS > 105)	137	88 (64.2)	25	19 (76.0)	37	28 (75.7)

Table 5 – **Patients' characteristics and clinical outcomes gathered at inclusion.**

Values are expressed as number (percentage) for qualitative variables and mean ± standard deviation or median (1st quartile to 3rd quartile) for quantitative variables. A normal ECAS score, indicating the absence of cognitive impairment as a symptom of ALS, would be more than 105 out of a total of 136 points.

Although men represented the majority in all three groups, the cohort is relatively balanced in terms of gender. The participants are aged between 50 and 70 years, with an average age of around 60 years, which is consistent with the typical age range for ALS patients, as the disease commonly affects individuals in later stages of life. Most patients in the cohort identify as Caucasian, aligning with the demographic characteristics discussed in section 2.1, and are treated with riluzole. The proportion of patients with a spinal onset of ALS is notably higher than those with a bulbar onset, which mirrors the general ALS population, where spinal onset is more common.

In terms of disease progression, the median time from the onset of symptoms to inclusion in the study is longer for the rsfMRI and T1 profiles compared to the total cohort, while the ALSFRS-R progression rate's median remains consistent across all groups.

Regarding ECAS, a majority of the cohort — 64.2%, 76%, and 75.7% in each respective group — scored within the normal range, indicating that most of these patients did not exhibit significant cognitive impairment. This finding is expected, as cognitive impairment is moderate in earlier stages of the disease. It is also important to note that the ECAS scores were not used in the training of the models, and the variability in ECAS scoring, depending on the assessment method, adds another layer of complexity to the interpretation of these results.

Results for rsfMRI

Preprocessing Results

After fMRI preprocessing, 62 connectivity matrices were generated for all sessions corresponding to the 26 patients included in the fMRI pipeline and 32 matrices for control cases. Figure 23 highlights the general structure of the connectivity matrices, where each axis represents 300 brain regions of interest. The color gradient reflects the statistical probability of functional connectivity between each pair of regions, with darker red tones indicating a higher likelihood of connection.

In the left, an example of the functional connectivity obtained for a healthy control from the PULSE cohort. In the right, a comparison that illustrates three connectivity matrices for two patients with differents phenotypes, corresponding to sessions at inclusion, three months after inclusion, and six months after inclusion.

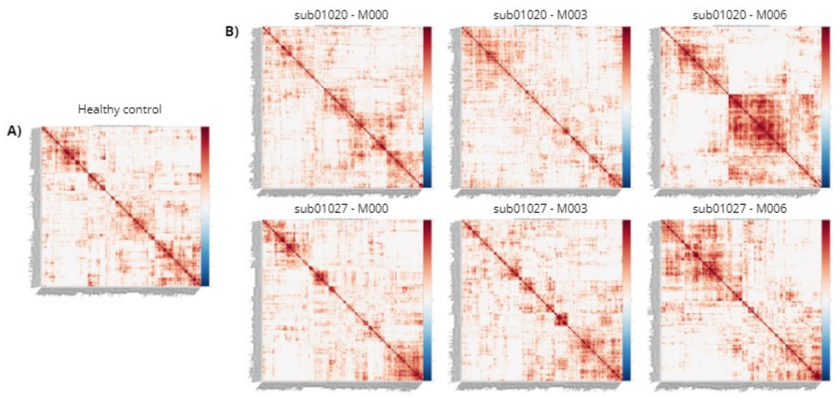


Figure 23 – **Connectivity Matrices for a healthy control and ALS patients.**

Example of the generated connectivity matrices for (A) a healthy control and (B) sessions M000 (inclusion), M003 (three months after inclusion) and M006 (6 months after inclusion) for two patients. Each row in (B) corresponds to a different subject. On top, a patient with a spinal onset site. In the bottom, a patient with a bulbar onset site.

Figure B enables comparison across sessions and between the two patients - on top, with a spinal onset site and, in the bottom, with a bulbar onset site - showing visual differences between the matrices for each individual, as well as changes over time, while maintaining the overall structure. Such divergences, even perceptible to the human eye, led us to believe that the ML models would be able to classify patients by onset site, as well as detect patterns that characterize the progression rate of the disease.

We also highlight that the regions for the control participant are not entirely more or less connected than in ALS patients. It seems to depend on each pair of ROIs from our results. However, dissimilarities between the healthy participant and the matrices in Figure 23 are significant enough to validate further investigation in diagnosis.

TDA Results

From the generated connectivity matrices, persistent homology was calculated for dimensions H_0, H_1, and H_2, and is directly represented by persistence diagrams, as shown in the first row of Figure 24 for one patient at inclusion. Before these results could be utilized in Machine Learning models, I transformed them into persistence representations. Figure 24 also illustrates the resulting persistence landscapes and persistence images in the second and third rows, respectively, using the Gudhi library. It is evident that the resulting persistence features differ significantly depending on the homology dimension analyzed, which in turn affects the final representations.

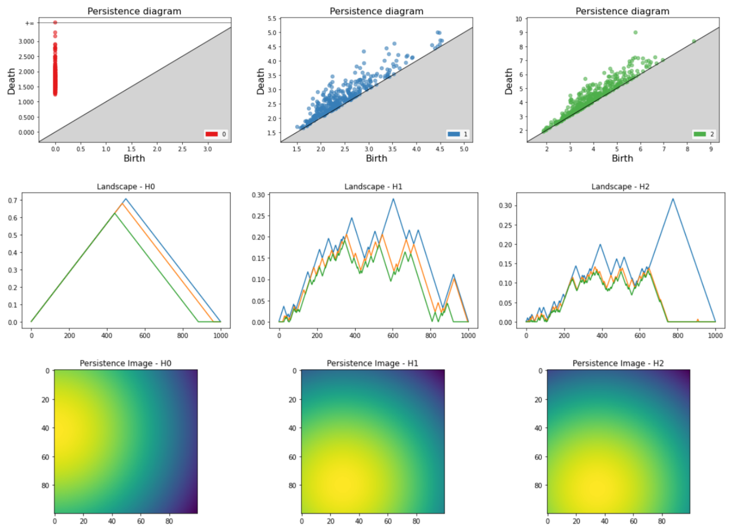


Figure 24 – **Persistence representations for rsfMRI.**

Example of the persistence diagrams and respective generated persistence representations from Gudhi library for session M000 for one patient in dimensions H_0, H_1 and H_2. On the top, the persistence diagrams. In the middle, the persistence landscapes. In the bottom, the persistence images.

Healthy Controls vs. ALS Patients

To distinguish ALS patients from healthy controls, both supervised and unsupervised machine learning methods were analyzed. The unsupervised methods tested included KMeans and Hierarchical Clustering. The results of these clustering methods are presented in Figure 28, which shows the distribution of the participants in a bidimensional space after applying Principal Component Analysis to the TDA vectors. The first graph on the left represents the expected classification, where the ALS patients and healthy controls are distinctly separated. The middle graph displays the results of the KMeans clustering, and the right graph shows the outcomes of Hierarchical Clustering.

Upon comparison, it is evident that neither KMeans nor Hierarchical Clustering successfully separated the controls from the patients across the three persistence dimensions (H_0, H_1, H_2). In fact, the clustering models misclassified nearly all participants, with only 3 out of the 32 healthy controls being correctly clustered apart from the ALS patients. The Adjusted Random Scores, shown below each image, reached very low values, confirming that the groups seem to be randomly designated. However, mostly for H_1 and H_2, the Silhouette Scores are close to 1, showing that the clusters are well-defined.

Table 6 shows the train and test accuracies obtained in dimensions 0, 1 and 2 for four supervised models: Support Vector Machine, Random Forest, KNeighbors and Logistic Regression. From its results, we can conclude that supervised models seem to be more promising, showing adequate results. Particularly, Random Forest performed the best in H_0, showing train accuracy of 100% and test accuracy of 73% in average. The results below were obtained for Perslay Landscape representation, which was the one that seemed to perform the best in this case, presenting the highest test accuracy. A table with the results for all representations can be found in Appendix K.

[t]

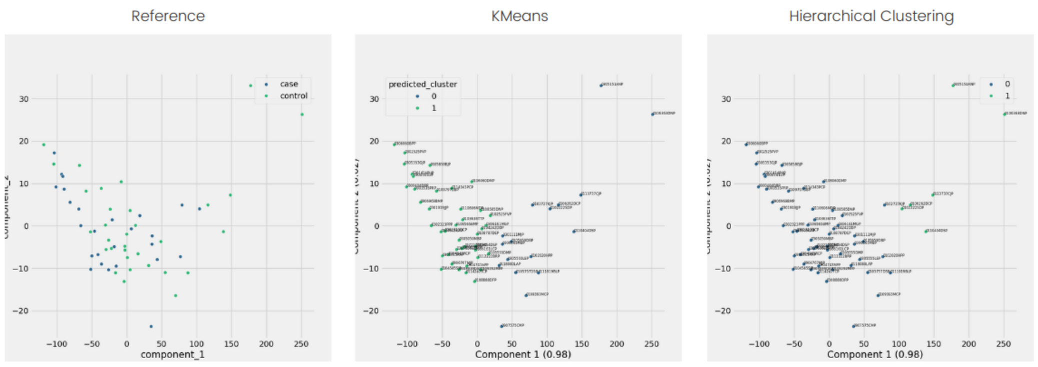


Figure 25 – **Clustering results for dimension H_0.**

Silhouette Score = 0.57 and Adjusted Random Score = -0.02 for KMeans and Silhouette Score = 0.61 and Adjusted Random Score = -0.01 for Hierarchical Clustering.

[t]

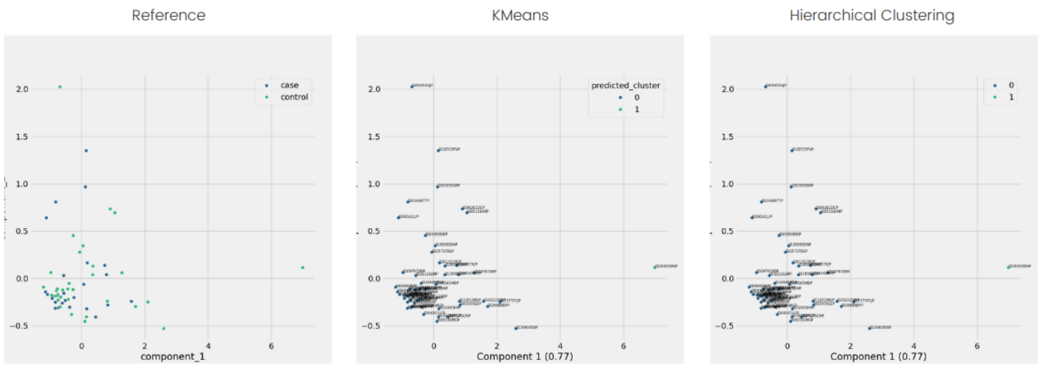


Figure 26 – **Clustering results for dimension H_1.**

Silhouette Score = 0.82 and Adjusted Random Score = -0.01 for both clustering methods.

[t]

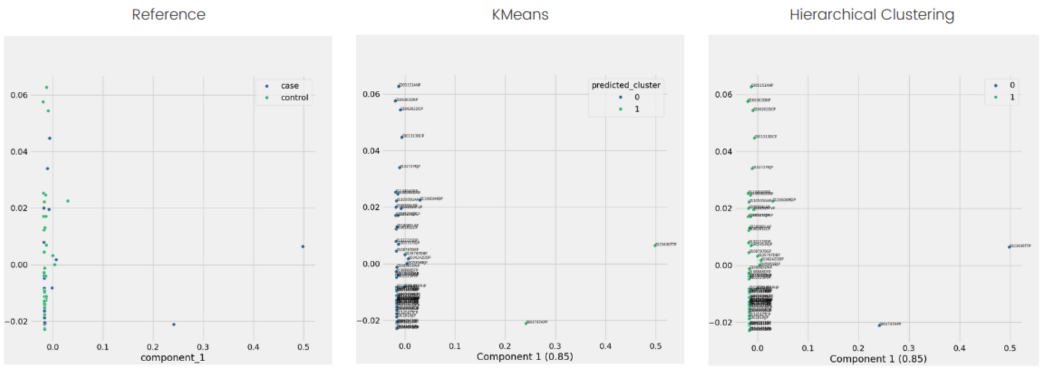


Figure 27 – **Clustering results for dimension H_2.**

Silhouette Score = 0.91 and Adjusted Random Score = 0.02 for both clustering methods.

Figure 28 – **Clustering for classifying healthy controls vs. ALS patients in fMRI.**

In the left, the Principal Component Analysis with the reference, that is, the expected result from clustering. In the middle, the resulting KMeans clustering. In the right, the resulting Hierarchical Clustering.

	SVM		Random Forest		KNeighbors		Logistic Regression	
Dimension	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
H_0	0.68 ± 0.02	0.62 ± 0.05	1.0 ± 0.0	0.73 ± 0.13	0.76 ± 0.04	0.57 ± 0.10	0.72 ± 0.04	0.50 ± 0.11
H_1	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.62 ± 0.14	0.77 ± 0.03	0.53 ± 0.09	0.61 ± 0.02	0.52 ± 0.09
H_2	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.43 ± 0.12	0.78 ± 0.03	0.47 ± 0.07	0.56 ± 0.01	0.55 ± 0.07

Table 6 – **Results for controls vs. patients classification in fMRI.**

Comparison between the accuracies obtained by different classification models, including SVM, Random Forest, KNeighbors and Logistic Regression, in dimensions H_0, H_1 and H_2.

Progression Classification

ALS patients were classified as either fast or slow progressors using four different models: Support Vector Machine, Random Forest, KNeighbors, and Logistic Regression. In total, 13 patients were used to train and test this model, where 8 were classified as fast and 5 were classified as slow. The accuracies obtained for dimensions 0, 1, and 2 are shown in Table 7. The Persistence Image representation implemented by the Gudhi library was chosen for this analysis as it achieved the highest overall test accuracy, specifically 0.93 ± 0.13 with Random Forest in H_0, and also performed well in terms of execution time. The complete table with all representations can be found in Appendix L.

The results in H_0 are particularly promising, with accuracy values approaching 100%, which is the ideal outcome. However, the accuracies obtained in H_2 were lower across all models. Assuming that a test accuracy above 70% is considered at least satisfying and above 80% is considered truly effective, the Random Forest and KNeighbors models yielded interesting results in both H_0 and H_1. This level of performance was not achieved by any of the tested models in H_2.

Model	Metric	H_0	H_1	H_2
SVM	Train accuracy	0.96 ± 0.05	0.67 ± 0.07	0.63 ± 0.04
	Test accuracy	0.87 ± 0.16	0.53 ± 0.12	0.60 ± 0.23
	Fast progressor accuracy	0.93 ± 0.13	0.93 ± 0.13	1.0 ± 0.0
	Slow progressor accuracy	0.70 ± 0.40	0.0 ± 0.0	0.0 ± 0.0
Random Forest	Train accuracy	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
	Test accuracy	0.93 ± 0.13	0.77 ± 0.20	0.37 ± 0.22
	Fast progressor accuracy	0.93 ± 0.13	0.93 ± 0.13	0.67 ± 0.42
	Slow progressor accuracy	0.80 ± 0.40	0.50 ± 0.45	0.0 ± 0.0
KNeighbors	Train accuracy	0.91 ± 0.06	0.94 ± 0.05	0.67 ± 0.05
	Test accuracy	0.70 ± 0.27	0.77 ± 0.29	0.47 ± 0.12
	Fast progressor accuracy	0.93 ± 0.13	1.0 ± 0.0	0.87 ± 0.27
	Slow progressor accuracy	0.40 ± 0.49	0.40 ± 0.49	0.0 ± 0.0
Logistic Regression	Train accuracy	0.75 ± 0.15	0.71 ± 0.13	0.81 ± 0.06
	Test accuracy	0.57 ± 0.25	0.50 ± 0.26	0.43 ± 0.25
	Fast progressor accuracy	0.63 ± 0.37	0.73 ± 0.39	0.73 ± 0.39
	Slow progressor accuracy	0.40 ± 0.49	0.10 ± 0.20	0.0 ± 0.0

Table 7 – **Results for progression classification in fMRI.**

Comparison between the accuracies obtained by different classification models, including SVM, Random Forest, KNeighbors and Logistic Regression, in dimensions H_0, H_1 and H_2.

Onset Site Classification

I applied Machine Learning to characterize ALS patients (N = 13, from which 11 were spinal and 2 were bulbar) by the disease's onset site using Support Vector Machine (SVM), Random Forest, KNeighbors, and Logistic Regression. The accuracies achieved by these models, as shown in Table 8, indicate that all models reached overall train and test accuracies of 70% or higher on average in dimensions 0 and 1. Notably, Support Vector Machine and Random Forest achieved accuracies above 80% across all dimensions.

However, when analyzing the accuracy of classifying each onset site individually, a large discrepancy becomes evident between the accuracy for spinal and bulbar classification. This difference may be explained by the imbalance in the available data. To ensure reliable accuracy in classifying onset sites, it would be necessary to obtain more MRI data from patients with bulbar onset.

The results presented were obtained using the Perslay Persistence Image representation, which yielded the highest accuracies despite having one of the longest execution times (ranging from 7 to 10 minutes). For a comparison of different representations, please refer to Appendix M.

Model	**Metric**	H_0	H_1	H_2
SVM	Train accuracy	1.0 ± 0.0	1.0 ± 0.0	0.94 ± 0.05
	Test accuracy	0.83 ± 0.21	0.53 ± 0.12	0.83 ± 0.21
	Spinal accuracy	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
	Bulbar accuracy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Random Forest	Train accuracy	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
	Test accuracy	0.83 ± 0.21	0.83 ± 0.21	0.83 ± 0.21
	Spinal accuracy	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
	Bulbar accuracy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
KNeighbors	Train accuracy	0.93 ± 0.07	0.92 ± 0.04	0.84 ± 0.14
	Test accuracy	0.70 ± 0.27	0.60 ± 0.08	0.57 ± 0.33
	Spinal accuracy	0.87 ± 0.27	0.77 ± 0.20	0.73 ± 0.39
	Bulbar accuracy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Logistic Regression	Train accuracy	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
	Test accuracy	0.70 ± 0.27	0.83 ± 0.21	0.63 ± 0.37
	Spinal accuracy	0.87 ± 0.27	1.0 ± 0.0	0.80 ± 0.40
	Bulbar accuracy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0

Table 8 – **Results for onset site classification in fMRI.**

Comparison between the accuracies obtained by different classification models, including SVM, Random Forest, KNeighbors and Logistic Regression, in dimensions H_0, H_1 and H_2.

Regression for disease's prediction evolution

Linear, Gradient Boosting and Random Forest regressions were tested for predicting the evolution of ALSFRS-R score between two sessions, but performed very poorly, as represented by the Table 9. The train coefficients of determination were high (equal or close to 1), but the test coefficients were, in all cases, negative. This possibly suggests overfitting and reveals that regression is not efficient for predicting the progression of the disease. The results below were obtained for Gudhi Landscape representation, which was the one that seemed to perform the best in this case, presenting less absurd values for the coefficient, and executed very fast. A more detailed table with all results can be consulted in Appendix N.

	Linear		Gradient Boosting		Random Forest	
Dimension	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**
H_0	1.0 $\pm$ 0.0	-4.99 $\pm$ 5.92	0.99 $\pm$ 6.35E-06	-3.28 $\pm$ 3.96	0.83 $\pm$ 0.03	-3.27 $\pm$ 4.81
H_1	1.0 $\pm$ 0.0	-13.09 $\pm$ 14.93	0.99 $\pm$ 3.68E-10	-2.03 $\pm$ 1.18	0.84 $\pm$ 0.02	-2.54 $\pm$ 3.43
H_2	1.0 $\pm$ 0.0	-72.81 $\pm$ 137.14	0.99 $\pm$ 5.67E-10	-2.03 $\pm$ 1.42	0.83 $\pm$ 0.01	-3.66 $\pm$ 4.61

Table 9 – **Results for Regression models in fMRI.**

Comparison between the coefficients of determination obtained by different regression models, including linear, gradient boosting and random forest, in dimensions H_0, H_1 and H_2.

Results for T1

Preprocessing Results

Figure 29 displays the results of the T1-weighted MRI preprocessing steps for one patient at inclusion. The corrected and skull-stripped brain in B) and the segmentation of white and gray matter in C) demonstrate that the preprocessed image is significantly clearer than the raw image in A), making it more suitable for subsequent analysis. This refinement ensures that only relevant information is retained for interpretation by the TDA.

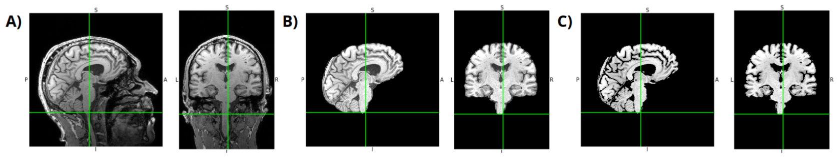


Figure 29 – **Preprocessed T1-w MRI.**

Example of one preprocessed T1-w MRI image for one patient at inclusion. (A) Raw T1; (B) Corrected and skull-stripped T1; (C) Segmented for white and gray matter.

TDA results

As for fMRI, one example for a single patient at inclusion is represented by Figure 30. The persistence diagrams in the first row for dimensions H_0, H_1 and H_2 were converted to persistence landscapes (second row) and persistence images (third row) using the Gudhi library. In this case, we can observe that there are similarities between each column, that is, for each dimension in the general format of the obtained representations.

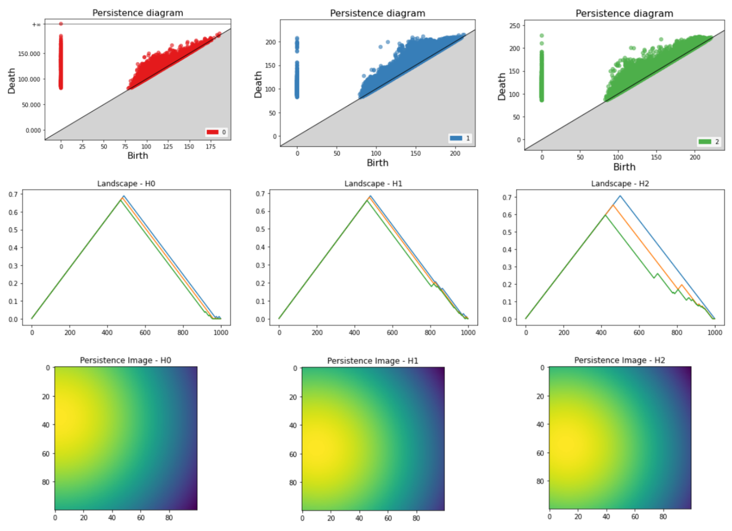
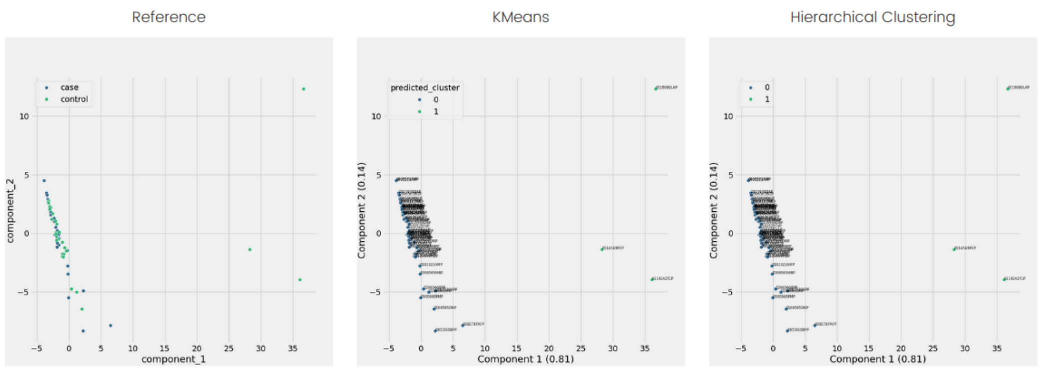


Figure 30 – **Persistence representations for T1-w MRI.**

Example of the persistence diagrams and respective generated persistence representations from Gudhi library for session M000 for one patient in dimensions H_0, H_1 and H_2. On the top, the persistence diagrams. In the middle, the persistence landscapes. In the bottom, the persistence images.

Healthy Controls vs. ALS Patients

As for rsfMRI, we tested Kmeans and Hierarchical Clustering as unsupervised models and Support Vector Machine, Random Forest, KNeighbors and Logistic Regression as supervised models to differentiate healthy controls from ALS patients. In Figure 34, we observe that the classification assigned by both clustering methods (middle and right graphs) do not correspond to the expected classification (left graph). Only 3 of the 32 controls were clustered separately from others. The Silhouette Scores, signalized below the images, have reached significant values, close to 1, nevertheless the Adjusted Random Scores were very close to 0, proving poor performance during clustering.



[t]

Figure 31 – **Clustering results for dimension H_0.**

Silhouette Score = 0.89 and Adjusted Random Score = 0.01 for both clustering methods. [t]

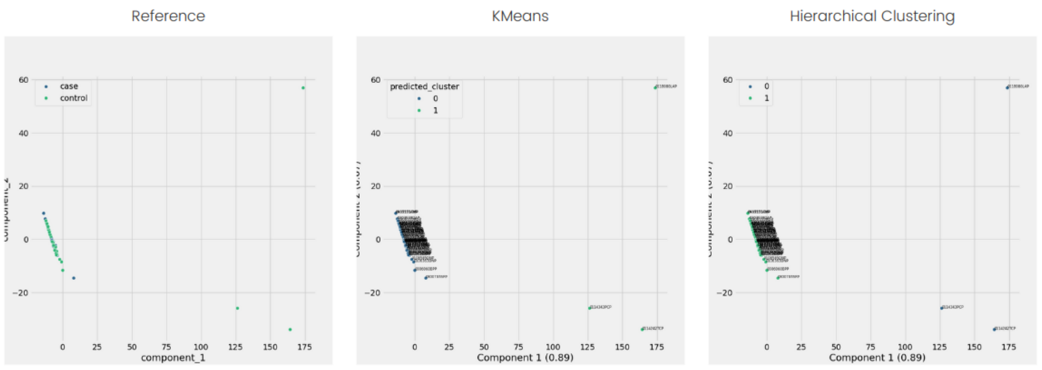


Figure 32 – **Clustering results for dimension H_1.**

Silhouette Score = 0.94 and Adjusted Random Score = 0.01 for both clustering methods. [t]

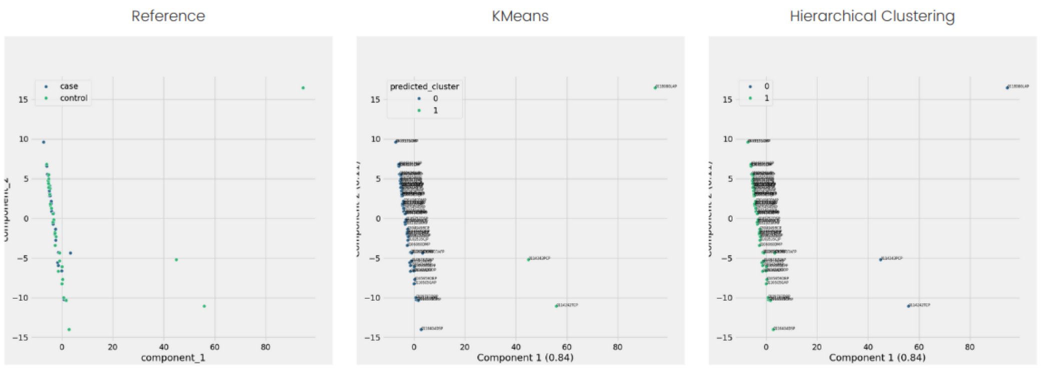


Figure 33 – **Clustering results for dimension H_2.**

Silhouette Score = 0.88 and Adjusted Random Score = 0.01 for both clustering methods.

Figure 34 – **Clustering for classifying healthy controls vs. ALS patients in T1.**

In the left, the Principal Component Analysis with the reference, that is, the expected result from clustering. In the middle, the resulting KMeans clustering. In the right, the resulting Hierarchical Clustering.

Even though the results for supervised classification (Table 10) were not at a satisfactory level, they are still better than for unsupervised models. For instance, the Logistic Regression model achieved train accuracy of 0.61 and test accuracy of 0.63 in average for H_1, which is higher than 50%, indicating that the classification is not entirely random, but still lower than 70% and therefore unsatisfactory. The results below were obtained for Perslay Landscape representation, which was the one that seemed to perform the best in this case. A more detailed table with all results can be consulted in Appendix O.

	SVM		**Random Forest**		**KNeighbors**		**Logistic Regression**	
Dimension	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
H_0	0.63 ± 0.03	0.52 ± 0.11	1.0 ± 0.0	0.53 ± 0.06	0.80 ± 0.02	0.49 ± 0.07	0.60 ± 0.02	0.39 ± 0.13
H_1	0.70 ± 0.04	0.53 ± 0.09	1.0 ± 0.0	0.55 ± 0.11	0.75 ± 0.03	0.50 ± 0.05	0.61 ± 0.04	0.63 ± 0.11
H_2	0.64 ± 0.04	0.48 ± 0.11	1.0 ± 0.0	0.47 ± 0.15	0.72 ± 0.04	0.45 ± 0.13	0.61 ± 0.05	0.55 ± 0.18

Table 10 – **Results for controls vs. patients classification in T1.**

Comparison between the accuracies obtained by different classification models, including SVM, Random Forest, KNeighbors and Logistic Regression, in dimensions H_0, H_1 and H_2.

Progression Classification

Classifying ALS patients as fast (N = 10) or slow (N = 12) progressors using treated T1 images from the PULSE cohort proved to be inefficient. Despite certain models achieving perfect train accuracies (100%), as shown in Table 11, all test accuracies were below 70%, which is unsatisfactory for reliable classification. The notable discrepancy between train and test accuracies suggests potential overfitting.

For dimensions 0 and 1, the analysis utilized Persim's Persistence Image as representation, while Gudhi's Persistence Image was used for dimension 2. These representations were chosen not only because they yielded higher test accuracy, but also because they presented more balanced values between train and test results. However, it's important to mention that Persim had the longest execution time among the tested representations. For a more detailed overview of the results, please refer to the table in Appendix P.

Model	**Metric**	H_0	H_1	H_2
SVM	Train accuracy	0.55 ± 0.03	0.57 ± 0.03	1.0 ± 0.0
	Test accuracy	0.60 ± 0.21	0.55 ± 0.12	0.60 ± 0.14
	Fast progressor accuracy	1.0 ± 0.0	1.0 ± 0.0	0.93 ± 0.13
	Slow progressor accuracy	0.20 ± 0.40	0.0 ± 0.0	0.17 ± 0.21
Random Forest	Train accuracy	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
	Test accuracy	0.60 ± 0.26	0.59 ± 0.09	0.55 ± 0.20
	Fast progressor accuracy	0.67 ± 0.42	0.73 ± 0.23	0.57 ± 0.39
	Slow progressor accuracy	0.57 ± 0.23	0.50 ± 0.11	0.57 ± 0.39
KNeighbors	Train accuracy	0.77 ± 0.06	0.65 ± 0.02	0.74 ± 0.06
	Test accuracy	0.44 ± 0.16	0.35 ± 0.14	0.54 ± 0.30
	Fast progressor accuracy	0.77 ± 0.29	0.73 ± 0.23	0.80 ± 0.24
	Slow progressor accuracy	0.27 ± 0.39	0.10 ± 0.20	0.40 ± 0.37
Logistic Regression	Train accuracy	0.56 ± 0.09	0.59 ± 0.03	0.75 ± 0.14
	Test accuracy	0.40 ± 0.14	0.51 ± 0.18	0.47 ± 0.25
	Fast progressor accuracy	0.70 ± 0.40	0.77 ± 0.29	0.53 ± 0.32
	Slow progressor accuracy	0.20 ± 0.40	0.20 ± 0.40	0.43 ± 0.23

Table 11 – **Results for progression classification in T1.**

Comparison between the accuracies obtained by different classification models, including SVM, Random Forest, KNeighbors and Logistic Regression, in dimensions H_0, H_1 and H_2.

Onset Site Classification

Analyzing the accuracies presented in Table 12, the models have performed very well in classifying patients as bulbar (N = 2) or spinal (N = 20) onset ALS. Notably, Support Vector Machine and Random Forest demonstrated significant results, consistently achieving train accuracies of 100% and average test accuracies of 87% across all tested dimensions. Nonetheless, given the difference in proportion between bulbar and spinal patients, these models would need further validation with more data from patients with a bulbar onset, The results were obtained using the Perslay Persistence Landscape representation, which not only produced good accuracies but also had one of the lowest execution times. For more detailed results, please refer to the table in Appendix Q.

Model	Metric	H_0	H_1	H_2
SVM	Train accuracy	0.99 ± 0.02	1.0 ± 0.0	1.0 ± 0.0
	Test accuracy	0.87 ± 0.17	0.87 ± 0.17	0.87 ± 0.17
	Spinal accuracy	0.93 ± 0.13	0.93 ± 0.13	0.93 ± 0.13
	Bulbar accuracy	0.20 ± 0.24	0.0 ± 0.0	0.0 ± 0.0
Random Forest	Train accuracy	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
	Test accuracy	0.87 ± 0.17	0.87 ± 0.17	0.87 ± 0.17
	Spinal accuracy	0.43 ± 0.39	0.57 ± 0.39	0.47 ± 0.32
	Bulbar accuracy	0.30 ± 0.40	0.50 ± 0.33	0.17 ± 0.21
KNeighbors	Train accuracy	0.90 ± 0.04	0.90 ± 0.06	0.90 ± 0.02
	Test accuracy	0.87 ± 0.17	0.82 ± 0.16	0.79 ± 0.18
	Spinal accuracy	0.63 ± 0.37	0.73 ± 0.23	0.43 ± 0.23
	Bulbar accuracy	0.23 ± 0.29	0.33 ± 0.18	0.0 ± 0.0
Logistic Regression	Train accuracy	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
	Test accuracy	0.78 ± 0.13	0.74 ± 0.15	0.69 ± 0.17
	Spinal accuracy	0.33 ± 0.28	0.60 ± 0.23	0.37 ± 0.19
	Bulbar accuracy	0.17 ± 0.21	0.23 ± 0.29	0.33 ± 0.42

Table 12 – **Results for onset site classification in T1.**

Comparison between the accuracies obtained by different classification models, including SVM, Random Forest, KNeighbors and Logistic Regression, in dimensions H_0, H_1 and H_2.

Regression for disease's prediction evolution

We did not obtain satisfactory results with the regression models. Table 13 shows the coefficients of determination obtained using Gudhi Landscape representations, which was the one that performed the best in this case. Indeed, test coefficients were negative, renforcing that regression is not recommended for predicting the disease progression using MRI images and Topological Data Analysis.

	Linear		**Gradient Boosting**		**Random Forest**	
Dimension	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**
H_0	1.0 ± 0.0	-43.03 ± 75.88	$0.99 \pm$ 5.49E-05	-1.36 ± 1.24	0.81 ± 0.02	-0.77 ± 0.41
H_1	1.0 ± 0.0	-6.19 ± 6.98	$0.99 \pm$ 3.65E-05	-1.37 ± 1.92	0.85 ± 0.03	-0.78 ± 0.61
H_2	1.0 ± 0.0	-8.80 ± 8.92	$0.99 \pm$ 2.55E-05	-2.56 ± 2.18	0.82 ± 0.04	-1.93 ± 2.46

Table 13 – **Results for Regression models in T1-w MRI.**

Comparison between the coefficients of determination obtained by different regression models, including linear, gradient boosting and random forest, in dimensions H_0, H_1 and H_2.

A.5 Conclusion

Unsupervised classification using KMeans and Hierarchical Clustering, along with regression models like Linear, Gradient Boosting, and Random Forest regressions for predicting the delta ALSFRS-R score between two sessions, proved ineffective for both fMRI and T1 data. Such results are disappointing, once Tang et al. (2019) (TANG et al., 2019) was able to achieve reliable clustering partitions and obtained reasonable prediction correlation for the ALSFRS score using only clinical data. Nonetheless, it is important to highlight that the dataset used in this study was considerably larger than PULSE, including 2424 ALS patients after data selection. Consequently, it would be necessary to test our methods with a much more significant amount of patients in order to draw a fair comparison and, thus, it cannot be attested that clinical data is more efficient than imaging data for regression training or clustering.

It is also worth mentioning that Tang et al. (2019) (TANG et al., 2019) obtained more interesting values for the coefficient of determination when predicting the Forced Vital Capacity (FVC), or the maximum amount of air one can exhale from the lungs after fully inhaling, than when predicting the ALSFRS slope with regression models. This means that it may be of our interest to test if better results can be accomplished by targeting the patients' FVC instead of the ALSFRS-R between two sessions.

Regarding diagnosis, despite not achieving state-of-the-art accuracy levels (88%) as shown in Kushol et al. (2023) (KUSHOL et al., 2023), the developed models achieved a respectable 73% accuracy. More notably, the 93% accuracy achieved in classifying disease progression as fast or slow using the fMRI pipeline suggests that the work holds promise and is worth pursuing further. This particular result gains interest, as it is equivalent or even surpasses other reported prognosis predictions.

Hothorn and Jung (2014) (HOTHORN; JUNG, 2014) did not reach encouraging accuracies using clinical data and indicates that the past ALSFRS slope and onset site are the only worthy candidates for predicting disease progression in clinic. Elamin et al. (2015) (ELAMIN et al., 2015) classified patients according to the ALS risk development using clinical data and obtained high values for Positive and Negative Prediction Values (73.3-85.7% and 93.3-100% respectively). As for the best results from our functional pipeline, Positive Prediction Value was 80-100% and Negative Prediction Value was 100%, which are, thus, comparable to state-of-the-art. Also, the highest prognosis accuracy obtained in Van Der Burgh et al. (2017) (van der Burgh et al., 2017) using both clinical and MRI data was of 84.4%, evidencing that the accuracy that we achieved is very significant.

In this project, we observed a comparative analysis of the performance of fMRI and T1 pipelines in distinguishing ALS patients from healthy controls, predicting disease progression, and classifying the onset site. Both pipelines exhibited similarities and differences depending on the tested models.

Table 14 compares the best test accuracies obtained in each pipeline for different dimensions (H0, H1, H2). Overall, the fMRI pipeline showed a slight edge over the T1 pipeline in certain tasks. A statistical comparison using the Mann-Whitney U test

(COMMUNITY, 2008-2024) was applied to each pair of accuracies (fMRI and T1) obtained from the supervised models across all dimensions. The results indicated that in most cases, there was no significant difference between the performances of the two imaging modalities. However, for dimension 0 in control vs. patient classification and progression rate prediction, the fMRI pipeline significantly outperformed the T1 pipeline (p-values of 0.036 and 0.044, respectively).

Pipeline	**Dimension**	**Best Test Accuracy in Diagnosis**	**Best Test Accuracy in Progression Classification**	**Best Test Accuracy in Onset Classification**
fMRI	H0	0.73 ± 0.13	0.93 ± 0.13	0.83 ± 0.21
	H1	0.62 ± 0.14	0.77 ± 0.20	0.92 ± 0.04
	H2	0.55 ± 0.07	0.60 ± 0.23	0.83 ± 0.21
T1	H0	0.53 ± 0.06	0.60 ± 0.21	0.87 ± 0.17
	H1	0.63 ± 0.11	0.59 ± 0.09	0.87 ± 0.17
	H2	0.55 ± 0.18	0.60 ± 0.14	0.87 ± 0.17

Table 14 – **Comparison between accuracies obtained for classification in fMRI and T1 pipelines.**

Support Vector Machine (SVM) and Random Forest models consistently provided the highest average accuracies across most classification tasks. Both KMeans and Hierarchical Clustering performed similarly in clustering tasks, while regression models yielded poor coefficients of determination. Linear regression, in particular, was the least adequate for predicting ALSFRS-R scores over time.

B Inclusion and non-inclusion criteria for each of the groups included in the cohort PULSE

Group	Inclusion Criteria	Non-Inclusion Criteria
ALS patients	- Having a possible/probable/definite SLA diagnosis following the El Escorial - Since first symptom or first deficit - Patient aged over 18 years old, man or woman - Ability to freely and clearly consent - Having social security	- Patient aged under 18 years old - Inability to consent - Presence of another serious pathology interfering on vital or functional prognostic
Healthy control	- Healthy subject aged over 18 years old paired by age and sex to the patient population. Ideally motivated and involved - Neurological tests and examination showing no neurological disorders - No serious pathology interfering on vital or functional prognostic - Having social security	- Subject under 18 years old - Inability to consent - Neurological tests showing neurological disorders - Serious pathology interfering on vital or functional prognostic - Not having social security - Contraindication to MRI
Neurological control	- Patient aged over 18 years old paired by age and sex to the patient population - Presenting another neurodegenerative disease (Parkinson's disease, Alzheimer's disease, etc.) - No other serious pathology interfering on vital or functional prognostic - Having social security	- Patient aged under 18 years old - Inability to consent - Another serious pathology interfering on vital or functional prognostic - Not having social security - Contraindication to MRI

C Bash code for running fMRI preprocessing

```
#!/bin/bash

directory=$1

# preprocessing
singularity run --cleanenv ./fmriprep/fmriprep.sif \
                $directory/data \
                $directory/fmriprep participant \
                --fs-license-file $directory/license.txt \
                --random-seed 42 \
                --omp-nthreads 1 \
                --skull-strip-fixed-seed

# postprocessing
singularity exec --nv postprocessing/postprocessing.sif \
        python3 postprocessing.py $directory/fmriprep/
```

D Python code for confound regression and functional connectivity matrices computation

```
from nilearn import image as nimg
from nilearn import datasets
from nilearn.input_data import NiftiLabelsMasker
from nilearn.connectome import ConnectivityMeasure

import os
import sys
import numpy as np
import nibabel as nib
import pandas as pd
import bids
import json

def postprocessing(sub, atlas_schaefer, fmriprep_dir, sessions):
        print(sub)

        for session in sessions:
                layout = bids.BIDSLayout(fmriprep_dir, validate=
                   False, config=['bids', 'derivatives'])
                print(session)
                # Get the file paths (preprocessed fMRI, mask
                   and confounds)
                func_files = layout.get(subject=sub,
                                        session=session,
                                        datatype='func', task='
                                           resting',
                                        desc='preproc',
                                        space='
                                           MNI152NLin2009cAsym',
                                        extension='nii.gz',
                                        return_type='file')

                mask_files = layout.get(subject=sub,
                                        session=session,
                                        datatype='func', task='
                                           resting',
                                        desc='brain',
                                        suffix='mask',
                                        space='
                                           MNI152NLin2009cAsym',
                                        extension='nii.gz',
                                        return_type='file')

            confound_files = layout.get(subject=sub,
                                            session=session,
                                            datatype='func',
                                               task='resting',
                                            desc='confounds',
                                            extension='tsv',
                                            return_type='file')
            try:
                    func_file = func_files[0]
                    mask_file = mask_files[0]
                    confound_file = confound_files[0]

                    # Select confounds to be considered in
                       confound regression
                    confound_df = pd.read_csv(confound_file,
                        delimiter='\t')
                    confound_vars = ['trans_x', 'trans_y', '
                       trans_z',
                                        'rot_x', 'rot_y', 'rot_z
                                           ',
                                        'csf', 'white_matter', '
                                           global_signal']
                    confound_df = confound_df[confound_vars]

                # Discarding the first four volumes of each
                   run to allow for T1-equilibration effects
                    raw_func_img = nimg.load_img(func_file)
                    func_img = raw_func_img.slicer[:,:,:,4:]
                    drop_confound_df = confound_df.loc[4:]

                # Get repetition time for subject from json
                    func_files_json = layout.get(subject=sub
            , session=session , datatype='func ',
            task='resting ', desc='preproc ', space
           ='MNI152NLin2009cAsym ', extension='
            json ', return_type='file ')

        f = open(func_files_json[0])
        data = json.load(f)
        repetition_time = data['RepetitionTime']
        f.close()

# Confound regression
        confounds_matrix = drop_confound_df.
            values
# parameters from Yeo 2011
        high_pass = 0.009
        low_pass = 0.08

        clean_img = nimg.clean_img(func_img,
            confounds=confounds_matrix, detrend=
            True,
                                                        st




                                                        t_
```

```
clean_dir = f'{fmriprep_dir}CleanImages/
    '
if not os.path.exists(clean_dir):
        os.makedirs(clean_dir)
clean_img_dir = f'{clean_dir}sub-{sub}
    _ses-{session}_clean.nii.gz'
clean_img.to_filename(clean_img_dir)

atlas_filename_schaefer = atlas_schaefer
    .maps
labels_schaefer = atlas_schaefer.labels

# Get connectivity matrix
connectivity_matrices_dir = f'{
    fmriprep_dir}
    ConnectivityMatrix_Negative/'
if not os.path.exists(
    connectivity_matrices_dir):
        os.makedirs(
            connectivity_matrices_dir)

kind = 'correlation'
atlas = 'schaefer'
masker = NiftiLabelsMasker(labels_img=
    atlas_filename_schaefer, standardize=
    True, verbose=5)
time_series = masker.fit_transform(
    clean_img_dir, confounds=
    confounds_matrix)

correlation_measure =
    ConnectivityMeasure(kind = kind)
correlation_matrix = correlation_measure
```

```
                    .fit_transform([time_series])[0]
                # correlation_matrix = np.absolute(
                    correlation_matrix)
                # correlation_matrix[correlation_matrix
                    < 0] = 0
                subject_connectivity_matrix = pd.
                    DataFrame(data=correlation_matrix,
                    index=labels_schaefer, columns=
                    labels_schaefer)
                subject_connectivity_matrix.to_csv(f'{
                    connectivity_matrices_dir}sub-{sub}
                    _ses-{session}_atlas-{atlas}_kind-{
                    kind}_connectivity_matrix.csv')
            except:
                print(f'Session␣{session}␣unavailable␣
                    for␣subject␣{sub}')
                continue

def postprocessing_for_all_subjects(directory):

    # Import the brain atlas
    atlas_schaefer = datasets.fetch_atlas_schaefer_2018(
        n_rois=300)
    subjects = [x[4:] for x in os.listdir(directory) if 'sub
        ' in x and 'html' not in x]
    for sub in subjects:
        sessions = [x[4:] for x in os.listdir(directory
            + 'sub-' + sub) if 'ses' in x]
        postprocessing(sub, atlas_schaefer, directory,
            sessions)

directory = sys.argv[1]
postprocessing_for_all_subjects(directory)
```

E Container definition for confound regression and functional connectivity matrix computation

```
BootStrap: docker
From: ubuntu:bionic

%labels
    APPLICATION_NAME fMRI postprocessing
    AUTHOR_NAME Fernanda Furukita
    AUTHOR_EMAIL fernanda.namie@gmail.com
    YEAR 2024

%help
    Container for postprocessing fMRI images and get functional
       connectivity matrices.

%environment
    # Set system locale
    PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local
       /sbin
    RDBASE=/usr/local/share/rdkit
    CUDA=/usr/local/cuda/lib64:/usr/local/cuda-10.1/lib64:/usr/
       local/cuda-10.2/lib64
    LD_LIBRARY_PATH=/.singularity.d/libs:$RDBASE/lib:$CUDA
    PYTHONPATH=modules:$RDBASE:/usr/local/share/rdkit/rdkit:/usr
       /local/lib/python3.6/dist-packages/
    LANG=C.UTF-8 LC_ALL=C.UTF-8

%post
    # Change to tmp directory to download temporary files.
    cd /tmp

    # Install essential software, languages and libraries.
    apt-get -qq -y update

    export DEBIAN_FRONTEND=noninteractive
    apt-get -qq install -y --no-install-recommends tzdata apt-
       utils
```

```
ln -fs /usr/share/zoneinfo/America/New_York /etc/localtime
dpkg-reconfigure --frontend noninteractive tzdata

apt-get -qq -y update
apt-get -qq install -y --no-install-recommends \
    autoconf \
    automake \
    build-essential \
    bzip2 \
    ca-certificates \
    cmake \
    gcc \
    g++ \
    gfortran \
    git \
    gnupg2 \
    libtool \
    libjpeg-dev \
    libpng-dev \
    libtiff-dev \
    libatlas-base-dev \
    libxml2-dev \
    zlib1g-dev \
    libcairo2-dev \
    libeigen3-dev \
    libcupti-dev \
    libpcre3-dev \
    libssl-dev \
    libcurl4-openssl-dev \
    libboost-all-dev \
    libboost-dev \
    libboost-system-dev \
    libboost-thread-dev \
    libboost-serialization-dev \
    libboost-regex-dev \
    libgtk2.0-dev \
    libreadline-dev \
    libbz2-dev \
    liblzma-dev \
        libpcre++-dev \
        libpango1.0-dev \
        libmariadb-client-lgpl-dev \
        libopenblas-dev \
        liblapack-dev \
        libxt-dev \
        neovim \
        openjdk-8-jdk \
        python \
        python-pip \
        python-dev \
        python3-dev \
        python3-pip \
        python3-wheel \
        swig \
        texlive \
        texlive-fonts-extra \
        texinfo \
        vim \
        wget \
        xvfb \
        xauth \
        xfonts-base \
        zip

    export LANG=C.UTF-8 LC_ALL=C.UTF-8

# Update python pip.
    python3 -m pip --no-cache-dir install --upgrade pip
    python3 -m pip --no-cache-dir install setuptools --upgrade
    python -m pip --no-cache-dir install setuptools --upgrade
# Install libraries
    python3 -m pip --no-cache-dir install numpy pandas nilearn
       nibabel pybids

# Cleanup
    apt-get -qq clean
    rm -rf /var/lib/apt/lists/*
    rm -rf /tmp/mpi
```

F Bash Script for adding SliceTiming to json files with fMRI acquisition parameters

```
#!/bin/bash

subjects=`ls -d ~/Documents/pulse_functional/data/*/`
slice_timing='␣␣␣␣"SliceTiming":␣[\n␣␣␣␣␣␣␣␣0.0,\n␣␣␣␣␣␣␣␣
    0.37333,\n␣␣␣␣␣␣␣␣0.74667,\n␣␣␣␣␣␣␣␣1.12,\n␣␣␣␣␣␣␣␣1.44,\n␣␣␣
    ␣␣␣␣␣1.76,\n␣␣␣␣␣␣␣␣2.08,\n␣␣␣␣␣␣␣␣0.053333,\n␣␣␣␣␣␣␣␣
    0.42667,\n␣␣␣␣␣␣␣␣0.8,\n␣␣␣␣␣␣␣␣1.1733,\n␣␣␣␣␣␣␣␣1.4933,\n␣␣␣
    ␣␣␣␣␣1.8133,\n␣␣␣␣␣␣␣␣2.1333,\n␣␣␣␣␣␣␣␣0.10667,\n␣␣␣␣␣␣␣␣
    0.48,\n␣␣␣␣␣␣␣␣0.85333,\n␣␣␣␣␣␣␣␣1.2267,\n␣␣␣␣␣␣␣␣1.5467,\n␣␣
    ␣␣␣␣␣␣1.8667,\n␣␣␣␣␣␣␣␣2.1867,\n␣␣␣␣␣␣␣␣0.16,\n␣␣␣␣␣␣␣␣
    0.53333,\n␣␣␣␣␣␣␣␣0.90667,\n␣␣␣␣␣␣␣␣1.28,\n␣␣␣␣␣␣␣␣1.6,\n␣␣␣␣
    ␣␣␣␣1.92,\n␣␣␣␣␣␣␣␣2.24,\n␣␣␣␣␣␣␣␣0.21333,\n␣␣␣␣␣␣␣␣0.58667,\
    n␣␣␣␣␣␣␣␣0.96,\n␣␣␣␣␣␣␣␣1.3333,\n␣␣␣␣␣␣␣␣1.6533,\n␣␣␣␣␣␣␣␣
    1.9733,\n␣␣␣␣␣␣␣␣2.2933,\n␣␣␣␣␣␣␣␣0.26667,\n␣␣␣␣␣␣␣␣0.64,\n␣␣
    ␣␣␣␣␣␣1.0133,\n␣␣␣␣␣␣␣␣1.3867,\n␣␣␣␣␣␣␣␣1.7067,\n␣␣␣␣␣␣␣␣
    2.0267,\n␣␣␣␣␣␣␣␣2.3467,\n␣␣␣␣␣␣␣␣0.32,\n␣␣␣␣␣␣␣␣0.69333,\n␣␣
    ␣␣␣␣␣␣1.0667\n␣␣␣␣],'


for subject in $subjects
do
        dirs=`ls -d $subject/*/func`
        for dir in $dirs; do
                fichiers=`ls $dir/*.json`
                for fichier in $fichiers
                do
                        if jq -e . $fichier 2>&1 >> json_file;
                           then
                                if ! grep -q SliceTiming
                                   $fichier; then
                                        sed -i "2s/^/
                                           $slice_timing\n/" "
                                           $fichier"
                                fi
                        fi
                done
        done
done
```

G Bash Script for preprocessing T1 images

```
#!/bin/bash
dir='/home/lesourd/Documents/FPII_raw/rawdatases'
T1_folder='/home/lesourd/Documents/FPII_raw/T1_corrected'
cd $dir
subjects=`ls -d */ | cut -f1 -d'/'`

for subject in $subjects
do
        sessions=`ls $subject`
        for session in $sessions
        do
                if [ ! -f $T1_folder/${subject}_${session}
                   _T1w_brain_seg.nii.gz ]; then
                        IFS=' ' read -ra file <<< `ls $dir/
                           $subject/$session/anat | grep nii.gz`
                        echo "=================================="
                        echo ${file[0]}
                        echo "=================================="
                        # bias correction
                        N4BiasFieldCorrection -i $dir/$subject/
                           $session/anat/${file[0]} -o [
                           $T1_folder/${subject}_${session}
                           _corrected_T1w.nii.gz, T1w_BiasField.
                           nii.gz ]
                        rm T1w_BiasField.nii.gz
                        # skull-stripping
                        hd-bet -i $T1_folder/${subject}_${
                           session}_corrected_T1w.nii.gz -o
                           $T1_folder/${subject}_${session}
                           _T1w_brain.nii.gz #-device cpu -mode
                           fast -tta 0
                        rm $T1_folder/${subject}_${session}
                           _T1w_brain_mask.nii.gz
                        # segmentation WM/GM
                        fast -o $T1_folder/${subject}_${session}
                           _T1w_brain $T1_folder/${subject}_${
                           session}_T1w_brain.nii.gz
                        rm $T1_folder/${subject}_${session}
```

```
                            _T1w_brain_pveseg.nii.gz
                        rm $T1_folder/${subject}_${session}
                            _T1w_brain_pve_0.nii.gz
                        rm $T1_folder/${subject}_${session}
                            _T1w_brain_pve_1.nii.gz
                        rm $T1_folder/${subject}_${session}
                            _T1w_brain_pve_2.nii.gz
                        rm $T1_folder/${subject}_${session}
                            _T1w_brain_mixeltype.nii.gz
                fi
        done
done
```

H Container definition for T1 preprocessing

```
BootStrap: docker
From: ubuntu:bionic

%labels
    APPLICATION_NAME T1 preprocessing
    AUTHOR_NAME Fernanda Furukita
    AUTHOR_EMAIL fernanda.namie@gmail.com
    YEAR 2024

%help
    Container for preprocessing T1-w images (correction, skull-
       stripping and WM/GM segmentation).

%files
    /home/amael-broustet/Documents/neurotda/T1MRI/singularity/
       fslinstaller.py /usr/local/fslinstaller.py

%environment
    # Set system locale
    PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local
       /sbin
    RDBASE=/usr/local/share/rdkit
    CUDA=/usr/local/cuda/lib64:/usr/local/cuda-10.1/lib64:/usr/
       local/cuda-10.2/lib64
    LD_LIBRARY_PATH=/.singularity.d/libs:$RDBASE/lib:$CUDA
    PYTHONPATH=modules:$RDBASE:/usr/local/share/rdkit/rdkit:/usr
       /local/lib/python3.6/dist-packages/
    LANG=C.UTF-8 LC_ALL=C.UTF-8
    export PATH="/usr/local/fsl/bin:$PATH"
    export PATH="/usr/local/ants/bin:$PATH"
    export LD_LIBRARY_PATH="/opt/ants/lib:$LD_LIBRARY_PATH"
        export ANTSPATH="/opt/ants/bin/"
        export PATH="/opt/ants/bin:$PATH"
        export LC_ALL=C

%post
    # Change to tmp directory to download temporary files.
    cd /tmp
```

```
# Install essential software, languages and libraries.
apt-get -qq -y update

export DEBIAN_FRONTEND=noninteractive
apt-get -qq install -y --no-install-recommends tzdata apt-
   utils

ln -fs /usr/share/zoneinfo/America/New_York /etc/localtime
dpkg-reconfigure --frontend noninteractive  tzdata

apt-get -qq -y update
apt-get -qq install -y --no-install-recommends \
    autoconf \
    automake \
    build-essential \
    bzip2 \
    ca-certificates \
    cmake \
    gcc \
    g++ \
    gfortran \
    git \
    gnupg2 \
    libtool \
    libjpeg-dev \
    libpng-dev \
    libtiff-dev \
    libatlas-base-dev \
    libxml2-dev \
    zlib1g-dev \
    libcairo2-dev \
    libeigen3-dev \
    libcupti-dev \
    libpcre3-dev \
    libssl-dev \
    libcurl4-openssl-dev \
    libboost-all-dev \
    libboost-dev \
        libboost-system-dev \
        libboost-thread-dev \
        libboost-serialization-dev \
        libboost-regex-dev \
        libgtk2.0-dev \
        libreadline-dev \
        libbz2-dev \
        liblzma-dev \
        libpcre++-dev \
        libpango1.0-dev \
        libmariadb-client-lgpl-dev \
        libopenblas-dev \
        liblapack-dev \
        libxt-dev \
        neovim \
        openjdk-8-jdk \
        python3.10 \
        python-pip \
        python-dev \
        python3-dev \
        python3-pip \
        python3-wheel \
        swig \
        texlive \
        texlive-fonts-extra \
        texinfo \
        vim \
        wget \
        xvfb \
        xauth \
        xfonts-base \
        zip

    export LANG=C.UTF-8 LC_ALL=C.UTF-8

# install ANTs for correction
        ANTSVERSION="2.4.3"

        apt-get update
```

```
apt-get install -y --no-install-recommends \
        apt-transport-https \
        bc \
        build-essential \
        ca-certificates \
        gnupg \
        ninja-build \
        git \
        software-properties-common \
        wget \
        zlib1g-dev

wget -O - https://apt.kitware.com/keys/kitware-archive-
    latest.asc 2>/dev/null | apt-key add -
apt-add-repository -y 'deb https://apt.kitware.com/
    ubuntu/ bionic main'
apt-get update
apt-get -y install cmake=3.18.3-0kitware1 cmake-data
    =3.18.3-0kitware1

mkdir -p /ants
cd /ants
git clone https://github.com/ANTsX/ANTs.git --branch=v${
    ANTSVERSION}
mv ANTs source
cd /ants/source



mkdir -p /ants/build
cd /ants/build
mkdir -p /opt/ants
git config --global url."https://".insteadOf git://
cmake \
        -G Ninja \
        -DBUILD_TESTING=ON \
        -DRUN_LONG_TESTS=OFF \
        -DRUN_SHORT_TESTS=ON \
        -DBUILD_SHARED_LIBS=ON \
        -DCMAKE_INSTALL_PREFIX=/opt/ants \
                /ants/source
        cmake --build . --parallel
        cd ANTS-build
        cmake --install .

# install hd-bet for skull-stripping
        pip3 install --upgrade pip
        pip3 install --upgrade setuptools
        pip3 install opencv-python opencv-contrib-python
        cd /usr/local
    git clone https://github.com/MIC-DKFZ/HD-BET
    cd HD-BET
    pip3 install -e .

 # install fsl for WM/GM segmentation
        python3 /usr/local/fslinstaller.py -d "/usr/local/fsl"
        rm /usr/local/fslinstaller.py
        FSLDIR=/usr/local/fsl
        PATH=${FSLDIR}/share/fsl/bin:${PATH}
        export FSLDIR PATH
        . ${FSLDIR}/etc/fslconf/fsl.sh

# Cleanup
    apt-get -qq clean
    rm -rf /var/lib/apt/lists/*
    rm -rf /tmp/mpi
```

I Python code for functions used in TDA and in Model Training

```python
import numpy as np
import pandas as pd
import gudhi as gd
from pylab import *
import nibabel as nib
import time
import psutil
import os

import tensorflow as tf
```

```
from sklearn.preprocessing import MinMaxScaler
import gudhi.representations as gdr
import gudhi.tensorflow.perslay as prsl
from persim import PersImage, PersistenceImager
from gtda.diagrams import PersistenceImage as PIgtda,
   PersistenceLandscape
from gudhi.representations import Landscape, PersistenceImage as
    PIgudhi

from sklearn.model_selection import KFold
from sklearn.metrics         import accuracy_score

from sklearn.svm             import SVC
from sklearn.ensemble        import RandomForestClassifier
from sklearn.neighbors       import KNeighborsClassifier
from sklearn.linear_model    import LogisticRegression
from sklearn.linear_model    import LinearRegression
from sklearn.ensemble        import GradientBoostingRegressor
from sklearn.ensemble        import RandomForestRegressor

# Parameters
# files = all preprocessed images
def get_M000_M006_files(files):

    # Gets which sessions each subject has
    subjects = {}
    for file in files:
        pos_sub = file.find('sub')
        pos_ses = file.find('ses')
        sub = file[pos_sub:pos_sub+12]
        session = file[pos_ses+4:pos_ses+8]
        if sub not in subjects:
            subjects[sub] = []

        subjects[sub].append(session)

    # Gets subjects that have each combination of sessions
       pairwise
    M000_M003 = []
```

```
M000_M006 = []
M003_M006 = []
M000_M012 = []
M003_M012 = []
M006_M012 = []

for sub in subjects.keys():
    if ('M000' in subjects[sub] and 'M003' in subjects[sub])
        :
        M000_M003.append(sub)

    if ('M000' in subjects[sub] and 'M006' in subjects[sub])
        :
        M000_M006.append(sub)

    if ('M006' in subjects[sub] and 'M003' in subjects[sub])
        :
        M003_M006.append(sub)

    if ('M000' in subjects[sub] and 'M012' in subjects[sub])
        :
        M000_M012.append(sub)

    if ('M012' in subjects[sub] and 'M003' in subjects[sub])
        :
        M003_M012.append(sub)

    if ('M006' in subjects[sub] and 'M012' in subjects[sub])
        :
        M006_M012.append(sub)


print(f'Patients␣that␣have␣M000␣and␣M003:␣{len(M000_M003)}')
print(f'Patients␣that␣have␣M000␣and␣M006:␣{len(M000_M006)}')
print(f'Patients␣that␣have␣M003␣and␣M006:␣{len(M003_M006)}')
print(f'Patients␣that␣have␣M000␣and␣M012:␣{len(M000_M012)}')
print(f'Patients␣that␣have␣M003␣and␣M012:␣{len(M003_M012)}')
print(f'Patients␣that␣have␣M006␣and␣M012:␣{len(M006_M012)}')
```

```
    # Gets files for both M000 and M006 only for the subjects
        that have both sessions simultaneously
    files_M0 = []
    files_M6 = []

    for file in files:
        pos_sub = file.find('sub')
        pos_ses = file.find('ses')
        sub = file[pos_sub:pos_sub+12]
        session = file[pos_ses+4:pos_ses+8]

        if session == 'M000' and sub in M000_M006:
            files_M0.append(file)
        elif session == 'M006' and sub in M000_M006:
            files_M6.append(file)

    # files are sorted so that they have the same order of
        patients
    files_M0.sort()
    files_M6.sort()
    return files_M0, files_M6

# Parameters
# path = localises where correlation matrices are
# files = all preprocessed images
def get_first_session_files(path, files):
    # Gets which sessions each subject has
    subjects = {}
    for file in files:
        pos_sub = file.find('sub')
        pos_ses = file.find('ses')
        sub = file[pos_sub:pos_sub+12]
        session = file[pos_ses+4:pos_ses+8]
        if sub not in subjects:
            subjects[sub] = []

        subjects[sub].append(session)

    first_session_files = []
    for sub in subjects.keys():
        if 'M000' in subjects[sub]:
            first_session_files.append(path + sub + '_ses-
               M000_atlas-schaefer_kind-
               correlation_connectivity_matrix.csv')
        elif 'M003' in subjects[sub]:
            first_session_files.append(path + sub + '_ses-
               M003_atlas-schaefer_kind-
               correlation_connectivity_matrix.csv')
        elif 'M006' in subjects[sub]:
            first_session_files.append(path + sub + '_ses-
               M006_atlas-schaefer_kind-
               correlation_connectivity_matrix.csv')
    return first_session_files

# Parameters
# files = T1 images for which the persistence will be calculated
def calculate_cubical_persistence(files, title):
    persistences_H0 = []
    persistences_H1 = []
    persistences_H2 = []
    order_subjects = []

    for file in files:
        print(file)
        sub = file[file.find('sub'):file.find('sub') + 12]
        img = nib.load(file)
        data = img.get_fdata()

        cubical = gd.CubicalComplex(
            top_dimensional_cells = data
        )
        cubical.compute_persistence(homology_coeff_field = 2,
           min_persistence = 0)

        persistences_H0.append(cubical.
           persistence_intervals_in_dimension(0))
        persistences_H1.append(cubical.
           persistence_intervals_in_dimension(1))
        persistences_H2.append(cubical.
            persistence_intervals_in_dimension(2))
        order_subjects.append(sub)

    np.savez(title, persistences_H0=np.array(persistences_H0,
        dtype='object'), persistences_H1=np.array(persistences_H1
        , dtype='object'), persistences_H2=np.array(
        persistences_H2, dtype='object'), order_subjects=
        order_subjects)

# Parameters
# files = fMRI images for which the persistence will be
    calculated
def calculate_simplex_persistence(files, title):
    persistences_H0 = []
    persistences_H1 = []
    persistences_H2 = []
    order_subjects = []

    for file in files:
        print(file)
        sub = file[file.find('sub'):file.find('sub') + 12]
        matrix = pd.read_csv(file, header=0, index_col=0).values
        matrix[matrix == 0] = np.finfo(float).smallest_normal
        inverse_matrix = 1 / (matrix)

        rips_complex = gd.RipsComplex(distance_matrix=
            inverse_matrix, max_edge_length=1e+3)
        simplex_tree = rips_complex.create_simplex_tree(
            max_dimension=3)
        simplex_tree.compute_persistence(homology_coeff_field =
            2)

        persistences_H0.append(simplex_tree.
            persistence_intervals_in_dimension(0))
        persistences_H1.append(simplex_tree.
            persistence_intervals_in_dimension(1))
        persistences_H2.append(simplex_tree.
            persistence_intervals_in_dimension(2))
        order_subjects.append(sub)

    np.savez(title, persistences_H0=np.array(persistences_H0,
       dtype='object'), persistences_H1=np.array(persistences_H1
       , dtype='object'), persistences_H2=np.array(
       persistences_H2, dtype='object'), order_subjects=
       order_subjects)

# Parameters
# pers = list with the persistences calculated to which Perslay
   will be applied
def get_perslay_vectors(pers, dim):
    # Preprocessing of persistences
    # DiagramSelector: Selects points that are non infinite
    # Diagram Scaler: Normalisation
    # Padding: Adds points in (0,0) so all persistences have the
        same size
    diagrams = gdr.DiagramSelector(use=True).fit_transform(np.
       asarray(pers, dtype='object'))
    diagrams = gdr.DiagramScaler(use=True, scalers=[([0,1],
       MinMaxScaler())]).fit_transform(diagrams)
    diagrams = gdr.Padding(use=True).fit_transform(np.asarray(
       diagrams, dtype='object'))

    # Padding returns 3D data (third dimension = point added
       during padding or not)
    diagrams_2D = []
    for diag in diagrams:
        diag_2D = []
        for point in diag:
            diag_2D.append([point[0], point[1]])
        diagrams_2D.append(np.array(diag_2D))

    start = time.time()
    # ==================================
    # PERSLAY
    # ==================================
    diagrams = tf.RaggedTensor.from_tensor(tf.constant(
    diagrams_2D, dtype=tf.float32))
rho = tf.identity
phi = prsl.GaussianPerslayPhi((100, 100), ((-.5, 1.5), (-.5,
     1.5)), .1)
phi = prsl.TentPerslayPhi(np.array(np.arange(-1.,2.,.001),
    dtype=np.float32))
weight = prsl.PowerPerslayWeight(1., 1.)
perm_op = tf.math.reduce_sum

perslay = prsl.Perslay(phi=phi, weight=weight, perm_op=
    perm_op, rho=rho)
vectors = perslay(diagrams)

# ===================================
# PERSIM - PersImage
# ===================================
pim = PersImage(pixels=[100,100], spread=1)
vectors = pim.transform(diagrams_2D)

# ===================================
# PERSIM - PersistenceImager
# ===================================
pimgr = PersistenceImager(pixel_size=0.005)
vectors = pimgr.fit_transform(diagrams_2D)
print('Resolution =', pimgr.resolution)

# ===================================
# GIOTTO-TDA
# ===================================
diagrams_3D = []
for diag in diagrams_2D:
     diagrams_3D.append([np.append(pair, dim) for pair in
         diag])
PI = PIgtda(sigma=0.1, n_bins=1000)
vectors = PI.fit_transform(diagrams_3D)

PL = PersistenceLandscape(n_layers=5, n_bins=100)
vectors = PL.fit_transform(diagrams_3D)
```

```
    # ===================================
    # GUDHI
    # ===================================
    vectors = Landscape(resolution=1000).fit_transform(
        diagrams_2D)
    vectors = PIgudhi(bandwidth=1.0, weight=lambda x: x[1]**2,
        im_range=[0, 1, 0, 1], resolution=[100,100]).
        fit_transform(diagrams_2D)

    print(f'Total␣time␣(Persistence␣Images):␣{time.time()␣-␣
        start}')
    print(f'RAM␣used␣(GB):␣{psutil.Process(os.getpid()).
        memory_info().rss␣/␣10**9}')
    print(f'Total␣Available␣RAM␣(GB):␣{psutil.virtual_memory().
        total␣/␣10␣**␣9}')
    print(f'Percentual:␣{psutil.Process(os.getpid()).memory_info
        ().rss␣/␣psutil.virtual_memory().total}')
    return np.array(vectors)

def test_models_classification(vectors_M0, vectors_M6,
   true_labels):
    X = []
    for x1, x2 in zip(vectors_M0, vectors_M6):
        X.append(np.concatenate([x1.flatten(), x2.flatten()]))
    test_models(np.array(X), np.array(true_labels))

# Parameters
# pers_group1 = first group of persistences to be compared
# pers_group2 = second group of persistences to be compared
# true_labels = expected labels (fast or slow progressor)
def test_models(X, y):

    start = time.time()
    model_SVC = SVC(gamma='auto', random_state=42)
    model_RF = RandomForestClassifier(random_state=42)
    model_KN = KNeighborsClassifier(n_neighbors=2)
    model_LR = LogisticRegression(random_state=42)

    n_splits = 5
```

```
kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)
acc_train = np.empty((4, n_splits))
acc_test = np.empty((4, n_splits))
idx = 0
for train, test in kf.split(X, y):
    X_train, X_test = X[train], X[test]
    y_train, y_test = y[train], y[test]

    model_SVC.fit(X_train, y_train)
    y_pred_test = model_SVC.predict(X_test)
    y_pred_train = model_SVC.predict(X_train)
    acc_train[0][idx] = accuracy_score(y_train, y_pred_train
        )
    acc_test[0][idx] = accuracy_score(y_test, y_pred_test)

    model_RF.fit(X_train, y_train)
    y_pred_test = model_RF.predict(X_test)
    y_pred_train = model_RF.predict(X_train)
    acc_train[1][idx] = accuracy_score(y_train, y_pred_train
        )
    acc_test[1][idx] = accuracy_score(y_test, y_pred_test)

    model_KN.fit(X_train, y_train)
    y_pred_test = model_KN.predict(X_test)
    y_pred_train = model_KN.predict(X_train)
    acc_train[2][idx] = accuracy_score(y_train, y_pred_train
        )
    acc_test[2][idx] = accuracy_score(y_test, y_pred_test)

    model_LR.fit(X_train, y_train)
    y_pred_test = model_LR.predict(X_test)
    y_pred_train = model_LR.predict(X_train)
    acc_train[3][idx] = accuracy_score(y_train, y_pred_train
        )
    acc_test[3][idx] = accuracy_score(y_test, y_pred_test)

    idx += 1

print(f'Total␣time␣(model␣training):␣{time.time()␣-␣start}')
```

```
    print(f'RAM␣used␣(GB):␣{psutil.Process(os.getpid()).
        memory_info().rss␣/␣10**9}')
    print(f'Total␣Available␣RAM␣(GB):␣{psutil.virtual_memory().
        total␣/␣10␣**␣9}')
    print(f'Percentual:␣{psutil.Process(os.getpid()).memory_info
        ().rss␣/␣psutil.virtual_memory().total}')

    print('====================SVC====================')
    print(f'Train␣Accuracy␣=␣{acc_train[0].mean()}␣+/-␣{
        acc_train[0].std()}')
    print(f'Test␣Accuracy␣=␣{acc_test[0].mean()}␣+/-␣{acc_test
        [0].std()}')

    print('====================RandomForest====================')
    print(f'Train␣Accuracy␣=␣{acc_train[1].mean()}␣+/-␣{
        acc_train[1].std()}')
    print(f'Test␣Accuracy␣=␣{acc_test[1].mean()}␣+/-␣{acc_test
        [1].std()}')

    print('====================KNeighbors====================')
    print(f'Train␣Accuracy␣=␣{acc_train[2].mean()}␣+/-␣{
        acc_train[2].std()}')
    print(f'Test␣Accuracy␣=␣{acc_test[2].mean()}␣+/-␣{acc_test
        [2].std()}')

    print('====================LogisticRegression====================')
    print(f'Train␣Accuracy␣=␣{acc_train[3].mean()}␣+/-␣{
        acc_train[3].std()}')
    print(f'Test␣Accuracy␣=␣{acc_test[3].mean()}␣+/-␣{acc_test
        [3].std()}')

def test_regression(pers1, pers2, deltas, dim):
    vectors1 = get_perslay_vectors(pers1, dim)
    vectors2 = get_perslay_vectors(pers2, dim)

    start = time.time()
    x = []
    for vector1, vector2 in zip(vectors1, vectors2):
        x.append(np.concatenate([vector1.flatten(), vector2.
            flatten()], 0))
    x = np.array(x)
    y = np.array(deltas)

    lr_model = LinearRegression()
    gb_model = GradientBoostingRegressor(random_state=42)
    rf_model = RandomForestRegressor(random_state=42)

    n_splits = 5
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)
    train_acc = np.empty((3, n_splits))
    test_acc = np.empty((3, n_splits))
    idx = 0
    for train, test in kf.split(x, y):
        X_train, X_test = x[train], x[test]
        y_train, y_test = y[train], y[test]

        lr_model.fit(X_train, y_train)
        train_acc[0][idx] = lr_model.score(X_train, y_train)
        test_acc[0][idx] = lr_model.score(X_test, y_test)

        gb_model.fit(X_train, y_train)
        train_acc[1][idx] = gb_model.score(X_train, y_train)
        test_acc[1][idx] = gb_model.score(X_test, y_test)

        rf_model.fit(X_train, y_train)
        train_acc[2][idx] = rf_model.score(X_train, y_train)
        test_acc[2][idx] = rf_model.score(X_test, y_test)
        idx += 1

    print(f'Total␣time␣(model␣training):␣{time.time()␣-␣start}')
    print(f'RAM␣used␣(GB):␣{psutil.Process(os.getpid()).
      memory_info().rss␣/␣10**9}')
    print(f'Total␣Available␣RAM␣(GB):␣{psutil.virtual_memory().
      total␣/␣10␣**␣9}')
    print(f'Percentual:␣{psutil.Process(os.getpid()).memory_info
      ().rss␣/␣psutil.virtual_memory().total}')
    print('===============Linear␣Regression================')
    print(f'Train␣Accuracy␣=␣{train_acc[0].mean()}␣+/-␣{
        train_acc[0].std()}')
    print(f'Test␣Accuracy␣=␣{test_acc[0].mean()}␣+/-␣{test_acc
        [0].std()}')
    print('===============Gradient␣Boosting␣Regression
        ===============')
    print(f'Train␣Accuracy␣=␣{train_acc[1].mean()}␣+/-␣{
        train_acc[1].std()}')
    print(f'Test␣Accuracy␣=␣{test_acc[1].mean()}␣+/-␣{test_acc
        [1].std()}')
    print('===============Random␣Forest␣Regression
        ===============')
    print(f'Train␣Accuracy␣=␣{train_acc[2].mean()}␣+/-␣{
        train_acc[2].std()}')
    print(f'Test␣Accuracy␣=␣{test_acc[2].mean()}␣+/-␣{test_acc
        [2].std()}')
```

J Python code for functions used in Clustering

```
import numpy as np
import pandas as pd
from pylab import *
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from kneed import KneeLocator
from sklearn.metrics import silhouette_score,
    adjusted_rand_score
from sklearn.cluster import AgglomerativeClustering

def acp(vectors):
    pca = PCA(n_components=2, random_state=42)
    size = 1
    for i in range(len(np.shape(vectors)) - 1):
        size *= np.shape(vectors)[i+1]
    pers_pca = pca.fit_transform(np.asarray(vectors).reshape((
        len(vectors), size)))
    explained_variances = [pca.explained_variance_ratio_[0], pca
       .explained_variance_ratio_[1]]
    return pers_pca, explained_variances

def clustering(vectors, order_subjects, name, qtte_cases,
   true_labels, use_elbow=True, predefined_n_clusters=2):
    pers_pca, explained_variances = acp(vectors)

    if use_elbow:
        sse = []
        for k in range (1,11):
            kmeans = KMeans(n_clusters=k, random_state=42, init=
               'random', n_init=10, max_iter=300)
            kmeans.fit(pers_pca)
            sse.append(kmeans.inertia_)

        kl = KneeLocator(range(1,11), sse, curve='convex',
           direction='decreasing')
        n_clusters = kl.elbow
    else:
        n_clusters = predefined_n_clusters

    model_clustering = KMeans(n_clusters=n_clusters, init='k-
       means++', n_init=50, max_iter=500, random_state=42)
    trained_model = model_clustering.fit(pers_pca)

    pcadf = pd.DataFrame(pers_pca, columns=['component_1', '
       component_2'])
    pcadf['predicted_cluster'] = trained_model.labels_

    plt.rcParams["text.usetex"] = False
    plt.style.use('fivethirtyeight')
    plt.figure(figsize=(8,8))
    scat = sns.scatterplot(data=pcadf, x='component_1', y='
       component_2', hue='predicted_cluster', palette='viridis')
    scat.set_xlabel(f'Component␣1␣({explained_variances[0]:.2f})
       ')
    scat.set_ylabel(f'Component␣2␣({explained_variances[1]:.2f})
       ')
    for i, txt in enumerate(order_subjects):
        scat.annotate(txt[4:9]+txt[-5:], (pers_pca[i,0],
            pers_pca[i,1]), fontsize=6)

    fig = scat.get_figure()
    fig.savefig('./results/' + name + '.png')
    plt.close()

    sil_score = silhouette_score(pers_pca, pcadf['
        predicted_cluster'])
    adj_score = adjusted_rand_score(true_labels, pcadf['
        predicted_cluster'])
    print(f'Silhouette␣Score␣=␣{sil_score}␣|␣Adjusted␣Random␣
        Score␣=␣{adj_score}')

    labels = ['case'] * qtte_cases + ['control'] * (len(vectors)
         - qtte_cases)
    plt.style.use('fivethirtyeight')
    plt.figure(figsize=(8,8))
    scat = sns.scatterplot(data=pcadf, x='component_1', y='
        component_2', hue=labels, palette='viridis')
    fig = scat.get_figure()
    fig.savefig('./results/' + name + '_ref.png')
    plt.close()

def HC_clustering(vectors, name, true_labels, order_subjects):
    pers_pca, explained_variances = acp(vectors)
    pcadf = pd.DataFrame(pers_pca, columns=['component_1', '
        component_2'])

    hierarchical_cluster = AgglomerativeClustering(n_clusters=2,
         linkage='ward')
    labels = hierarchical_cluster.fit_predict(pers_pca)

    plt.rcParams["text.usetex"] = False
    plt.style.use('fivethirtyeight')
    plt.figure(figsize=(8,8))
    scat = sns.scatterplot(data=pcadf, x='component_1', y='
    component_2', hue=labels, palette='viridis')
scat.set_xlabel(f'Component␣1␣({explained_variances[0]:.2f})
    ')
scat.set_ylabel(f'Component␣2␣({explained_variances[1]:.2f})
    ')

for i, txt in enumerate(order_subjects):
     scat.annotate(txt[4:9]+txt[-5:], (pers_pca[i,0],
         pers_pca[i,1]), fontsize=6)

fig = scat.get_figure()
fig.savefig('./results/' + name + '.png')
plt.close()

sil_score = silhouette_score(pers_pca, labels)
adj_score = adjusted_rand_score(true_labels, labels)
print(f'Silhouette␣Score␣=␣{sil_score}␣|␣Adjusted␣Random␣
    Score␣=␣{adj_score}')
```

K Results for all representations in control vs. patient classification from the rsfMRI pipeline

					H0							
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	142.73	124.9	0.82	1.0 ± 0.0	0.53 ± 0.08	1.0 ± 0.0	0.63 ± 0.21	0.75 ± 0.06	0.52 ± 0.07	0.70 ± 0.04	0.52 ± 0.07
	Persim	4.6	7.9	0.72	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.45 ± 0.07	0.73 ± 0.02	0.34 ± 0.09	0.73 ± 0.02	0.55 ± 0.07
	Giotto						No Results					
	Gudhi	4.7	40.1	0.72	0.67 ± 0.02	0.58 ± 0.11	1.0 ± 0.0	0.50 ± 0.11	0.69 ± 0.03	0.53 ± 0.06	0.66 ± 0.03	0.62 ± 0.08
PL	Perslay	5.3	4.3	0.73	0.68 ± 0.02	0.62 ± 0.05	1.0 ± 0.0	0.73 ± 0.13	0.76 ± 0.04	0.57 ± 0.10	0.72 ± 0.04	0.50 ± 0.11
	Giotto	0.1	1.3	0.72	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.57 ± 0.11	0.74 ± 0.02	0.55 ± 0.07	0.72 ± 0.04	0.64 ± 0.09
	Gudhi	0.3	12.2	0.72	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.57 ± 0.11	0.74 ± 0.02	0.55 ± 0.07	0.77 ± 0.03	0.64 ± 0.08

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H0 for classifying functional MRI as healthy control or ALS patient.

					H1							
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	211.5	206.5	0.92	0.96 ± 0.03	0.57 ± 0.08	1.0 ± 0.0	0.48 ± 0.09	0.72 ± 0.03	0.47 ± 0.05	0.82 ± 0.04	0.60 ± 0.06
	Persim	6.8	7.4	0.73	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.57 ± 0.06	0.75 ± 0.04	0.55 ± 0.10	0.55 ± 0.02	0.55 ± 0.07
	Giotto						No Results					
	Gudhi	8.6	69.0	0.73	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.49 ± 0.26	0.68 ± 0.07	0.38 ± 0.08	0.58 ± 0.03	0.45 ± 0.07
PL	Perslay	7.4	3.8	0.73	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.62 ± 0.14	0.77 ± 0.03	0.53 ± 0.09	0.61 ± 0.02	0.52 ± 0.09
	Giotto	0.1	1.4	0.72	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.52 ± 0.17	0.76 ± 0.04	0.45 ± 0.10	0.65 ± 0.06	0.52 ± 0.04
	Gudhi	0.7	7.4	0.72	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.50 ± 0.13	0.77 ± 0.04	0.45 ± 0.10	0.78 ± 0.05	0.48 ± 0.11

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H1 for classifying functional MRI as healthy control or ALS patient.

					H2							
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	350.4	202.2	1.01	0.67 ± 0.06	0.55 ± 0.14	1.0 ± 0.0	0.64 ± 0.11	0.76 ± 0.03	0.53 ± 0.15	0.72 ± 0.03	0.52 ± 0.11
	Persim	10.8	7.8	0.73	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.43 ± 0.15	0.73 ± 0.05	0.48 ± 0.19	0.55 ± 0.02	0.55 ± 0.07
	Giotto						No Results					
	Gudhi	11.5	16.4	0.73	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.59 ± 0.15	0.80 ± 0.06	0.58 ± 0.14	0.58 ± 0.03	0.57 ± 0.08
PL	Perslay	11.5	3.6	0.73	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.43 ± 0.12	0.78 ± 0.03	0.47 ± 0.07	0.56 ± 0.01	0.55 ± 0.07
	Giotto	0.2	1.6	0.73	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.55 ± 0.09	0.70 ± 0.03	0.45 ± 0.13	0.56 ± 0.01	0.55 ± 0.07
	Gudhi	0.9	5.0	0.73	0.55 ± 0.02	0.55 ± 0.07	1.0 ± 0.0	0.59 ± 0.06	0.74 ± 0.03	0.42 ± 0.10	0.60 ± 0.01	0.57 ± 0.02

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H2 for classifying functional MRI as healthy control or ALS patient.

L Results for all representations in progression classification from the rsfMRI pipeline

H0												
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	74.9	254.1	0.73	1.0 ± 0.0	0.53 ± 0.12	1.0 ± 0.0	0.77 ± 0.20	0.91 ± 0.06	0.70 ± 0.27	1.0 ± 0.0	0.47 ± 0.12
	Persim	2.3	23.0	0.69	0.63 ± 0.04	0.60 ± 0.23	1.0 ± 0.0	0.37 ± 0.22	0.69 ± 0.07	0.60 ± 0.23	0.63 ± 0.04	0.60 ± 0.23
	Giotto	23.8	281.2	0.90	0.61 ± 0.07	0.40 ± 0.23	0.61 ± 0.07	0.60 ± 0.23	0.61 ± 0.07	0.60 ± 0.23	0.61 ± 0.07	0.60 ± 0.23
	Gudhi	4.7	139.5	0.69	0.96 ± 0.05	0.87 ± 0.16	1.0 ± 0.0	0.93 ± 0.13	0.91 ± 0.06	0.70 ± 0.27	0.75 ± 0.15	0.57 ± 0.25
PL	Perslay	3.1	6.7	0.73	0.96 ± 0.05	0.87 ± 0.16	1.0 ± 0.0	0.73 ± 0.23	0.91 ± 0.06	0.70 ± 0.27	1.0 ± 0.0	0.53 ± 0.12
	Giotto	0.1	1.3	0.69	0.69 ± 0.11	0.53 ± 0.12	1.0 ± 0.0	0.83 ± 0.21	0.96 ± 0.05	0.83 ± 0.21	0.83 ± 0.08	0.53 ± 0.12
	Gudhi	0.1	130.8	0.69	0.69 ± 0.11	0.53 ± 0.12	1.0 ± 0.0	0.83 ± 0.21	0.96 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.67 ± 0.18

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H0 for classifying functional MRI by progression rate.

H1												
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	84.5	246.8	0.75	1.0 ± 0.0	0.53 ± 0.12	1.0 ± 0.0	0.60 ± 0.08	0.85 ± 0.04	0.60 ± 0.08	1.0 ± 0.0	0.63 ± 0.22
	Persim	2.7	23.8	0.70	0.65 ± 0.04	0.60 ± 0.23	1.0 ± 0.0	0.73 ± 0.23	0.88 ± 0.08	0.60 ± 0.23	0.65 ± 0.04	0.60 ± 0.23
	Giotto	21.8	240.6	1.11	1.0 ± 0.0	0.40 ± 0.23	1.0 ± 0.0	0.63 ± 0.22	0.94 ± 0.05	0.60 ± 0.08	1.0 ± 0.0	0.63 ± 0.22
	Gudhi	4.2	26.3	0.76	0.67 ± 0.07	0.53 ± 0.12	1.0 ± 0.0	0.77 ± 0.20	0.94 ± 0.05	0.77 ± 0.29	0.71 ± 0.13	0.50 ± 0.26
PL	Perslay	3.2	5.7	0.73	0.67 ± 0.07	0.53 ± 0.12	1.0 ± 0.0	0.53 ± 0.12	0.69 ± 0.05	0.70 ± 0.27	0.81 ± 0.01	0.43 ± 0.25
	Giotto	0.1	1.3	0.70	0.67 ± 0.07	0.53 ± 0.12	1.0 ± 0.0	0.53 ± 0.12	0.67 ± 0.05	0.60 ± 0.23	0.69 ± 0.06	0.53 ± 0.12
	Gudhi	0.1	24.1	0.69	0.67 ± 0.07	0.53 ± 0.12	1.0 ± 0.0	0.53 ± 0.12	0.67 ± 0.05	0.60 ± 0.23	0.87 ± 0.04	0.37 ± 0.22

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H1 for classifying functional MRI by progression rate.

H2												
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	104.0	226.2	0.74	1.0 ± 0.0	0.53 ± 0.12	1.0 ± 0.0	0.47 ± 0.12	0.79 ± 0.07	0.67 ± 0.28	1.0 ± 0.0	0.40 ± 0.08
	Persim	3.6	23.4	0.70	1.0 ± 0.0	0.40 ± 0.23	1.0 ± 0.0	0.70 ± 0.16	0.73 ± 0.08	0.60 ± 0.23	0.67 ± 0.07	0.53 ± 0.12
	Giotto	55.2	302.9	1.32	1.0 ± 0.0	0.53 ± 0.34	1.0 ± 0.0	0.53 ± 0.34	0.86 ± 0.05	0.63 ± 0.37	1.0 ± 0.0	0.27 ± 0.25
	Gudhi	5.4	14.9	0.70	0.63 ± 0.04	0.60 ± 0.23	1.0 ± 0.0	0.37 ± 0.22	0.67 ± 0.05	0.47 ± 0.12	0.81 ± 0.06	0.43 ± 0.25
PL	Perslay	3.7	5.8	0.73	0.63 ± 0.04	0.60 ± 0.23	1.0 ± 0.0	0.47 ± 0.12	0.65 ± 0.04	0.47 ± 0.12	0.77 ± 0.07	0.60 ± 0.23
	Giotto	0.1	1.4	0.70	0.65 ± 0.04	0.60 ± 0.23	1.0 ± 0.0	0.47 ± 0.12	0.67 ± 0.07	0.47 ± 0.12	0.75 ± 0.04	0.60 ± 0.23
	Gudhi	0.1	13.9	0.69	0.65 ± 0.04	0.60 ± 0.23	1.0 ± 0.0	0.47 ± 0.12	0.67 ± 0.07	0.47 ± 0.12	0.87 ± 0.07	0.47 ± 0.12

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H2 for classifying functional MRI by progression rate.

M Results for all representations in onset classification from the rsfMRI pipeline

					H0							
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	194.6	240.5	0.72	1.0 ± 0.0	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.93 ± 0.07	0.70 ± 0.27	1.0 ± 0.0	0.70 ± 0.27
	Persim	2.3	17.1	0.69	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.77 ± 0.20	0.92 ± 0.07	0.63 ± 0.22	0.85 ± 0.05	0.83 ± 0.21
	Giotto	32.7	549.8	0.90	0.85 ± 0.05	0.83 ± 0.21	0.85 ± 0.05	0.83 ± 0.21	0.61 ± 0.33	0.43 ± 0.39	0.85 ± 0.05	0.83 ± 0.21
	Gudhi	2.1	170.2	0.69	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.77 ± 0.20	0.93 ± 0.07	0.70 ± 0.27	0.93 ± 0.11	0.70 ± 0.27
PL	Perslay	5.11	15.8	0.73	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.93 ± 0.07	0.70 ± 0.27	1.0 ± 0.0	0.70 ± 0.27
	Giotto	0.1	1.3	0.69	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.77 ± 0.07	0.53 ± 0.34	0.85 ± 0.05	0.83 ± 0.21
	Gudhi	0.1	97.1	0.69	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.77 ± 0.07	0.53 ± 0.34	1.0 ± 0.0	0.83 ± 0.21

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H0 for classifying functional MRI by onset site.

					H1							
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	224.2	373.8	0.74	1.0 ± 0.0	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.92 ± 0.04	0.60 ± 0.08	1.0 ± 0.0	0.83 ± 0.21
	Persim	2.5	17.8	0.70	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.67 ± 0.18	0.81 ± 0.06	0.47 ± 0.12	0.85 ± 0.05	0.83 ± 0.21
	Giotto	32.1	370.0	1.11	1.0 ± 0.0	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.96 ± 0.05	0.67 ± 0.18	1.0 ± 0.0	0.83 ± 0.21
	Gudhi	2.0	49.8	0.79	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.73 ± 0.23	0.81 ± 0.06	0.53 ± 0.32	0.85 ± 0.05	0.83 ± 0.21
PL	Perslay	5.3	12.0	0.74	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.73 ± 0.23	0.78 ± 0.13	0.40 ± 0.08	0.85 ± 0.05	0.83 ± 0.21
	Giotto	0.1	1.3	0.70	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.71 ± 0.11	0.43 ± 0.33	0.85 ± 0.05	0.83 ± 0.21
	Gudhi	0.1	22.1	0.69	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.71 ± 0.11	0.43 ± 0.33	0.85 ± 0.05	0.83 ± 0.21

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H1 for classifying functional MRI by onset site.

					H2							
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	254.24	260.9	0.77	0.94 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.75 ± 0.14	0.47 ± 0.24	1.0 ± 0.0	0.63 ± 0.37
	Persim	2.9	16.1	0.70	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.75 ± 0.14	0.47 ± 0.24	0.85 ± 0.05	0.83 ± 0.21
	Giotto	77.5	357.9	1.32	1.0 ± 0.0	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.86 ± 0.08	0.50 ± 0.26	1.0 ± 0.0	0.60 ± 0.23
	Gudhi	2.86	30.9	0.81	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.63 ± 0.37	0.82 ± 0.12	0.57 ± 0.39	0.85 ± 0.05	0.63 ± 0.37
PL	Perslay	7.35	10.8	0.74	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.86 ± 0.10	0.53 ± 0.45	0.85 ± 0.05	0.83 ± 0.21
	Giotto	0.1	1.3	0.70	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.88 ± 0.08	0.47 ± 0.40	0.85 ± 0.05	0.83 ± 0.21
	Gudhi	0.1	17.9	0.69	0.85 ± 0.05	0.83 ± 0.21	1.0 ± 0.0	0.83 ± 0.21	0.88 ± 0.08	0.47 ± 0.40	0.92 ± 0.04	0.83 ± 0.21

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H2 for classifying functional MRI by onset site.

N Results for all representations in regression from the rsfMRI pipeline

					H0					
					Linear		**Gradient Boosting**		**Random Forest**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**
PI	PersLay	123.3	46.4	0.73	0.99 ± 3.70E-07	-241.36 ± 233.95	0.99 ± 9.65E-08	-5.04 ± 9.17	0.81 ± 0.02	-2.76 ± 4.30
	Persim	4.1	80.3	0.69	0.99 ± 3.09E-03	-52.67 ± 50.78	0.99 ± 4.89E-05	-11.98 ± 22.70	0.82 ± 0.04	-7.03 ± 12.69
	Giotto	0.4	775.9	1.09	0.0 ± 0.0	-2.64 ± 3.49	0.0 ± 0.0	-2.64 ± 3.49	-4.85E-04 ± 2.80E-04	-2.55 ± 3.23
	Gudhi	8.1	176.7	0.74	0.87 ± 0.07	-1066.43 ± 1980.18	0.99 ± 2.16E-04	-5.68 ± 6.14	0.76 ± 0.04	-2.85 ± 3.76
PL	Perslay	4.9	9.2	0.72	0.99 ± 2.69E-10	-13.89 ± 21.78	0.99 ± 6.68E-07	-2.11 ± 3.67	0.82 ± 0.03	-2.38 ± 3.79
	Giotto	0.1	3.9	0.67	1.0 ± 0.0	-4.97 ± 5.93	0.99 ± 6.35E-06	-2.59 ± 2.96	0.83 ± 0.03	-3.19 ± 4.66
	Gudhi	0.1	21.6	0.67	1.0 ± 0.0	-4.99 ± 5.92	0.99 ± 6.35E-06	-3.28 ± 3.96	0.83 ± 0.03	-3.27 ± 4.81

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H0 for the regression in the rsfMRI pipeline.

					H1					
					Linear		**Gradient Boosting**		**Random Forest**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**
PI	PersLay	157.4	48.7	0.74	0.99 ± 9.17E-12	-18.61 ± 22.64	0.99 ± 5.35E-09	-3.84 ± 3.52	0.82 ± 0.03	-3.07 ± 3.20
	Persim	5.2	78.2	0.69	1.0 ± 0.0	-266.73 ± 362.34	0.99 ± 1.70E-04	-4.24 ± 5.61	0.77 ± 0.04	-3.65 ± 4.72
	Giotto	105.2	13504.1	0.75	1.0 ± 0.0	-16.65 ± 16.84	0.99 ± 2.63E-08	-2.62 ± 2.67	0.84 ± 0.03	-2.38 ± 2.54
	Gudhi	7.7	178.6	0.69	1.0 ± 0.0	-1212.56 ± 1851.39	0.99 ± 7.13E-05	-3.21 ± 5.93	0.81 ± 0.04	-3.32 ± 5.31
PL	Perslay	5.9	6.9	0.77	0.99 ± 8.97E-13	-8.29 ± 9.40	0.99 ± 5.80E-09	-4.42 ± 3.72	0.84 ± 0.03	-3.10 ± 3.36
	Giotto	0.2	3.6	0.67	1.0 ± 0.0	-13.01 ± 14.70	0.99 ± 1.19E-09	-3.30 ± 2.79	0.84 ± 0.02	-3.10 ± 3.65
	Gudhi	0.2	21.2	0.66	1.0 ± 0.0	-13.09 ± 14.93	0.99 ± 3.68E-10	-2.03 ± 1.18	0.84 ± 0.02	-2.54 ± 3.43

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H1 for the regression in the rsfMRI pipeline.

					H2					
					Linear		**Gradient Boosting**		**Random Forest**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**
PI	PersLay	223.1	44.9	0.79	0.99 ± 5.79E-12	-23.56 ± 25.62	0.99 ± 1.92E-08	-1.60 ± 1.01	0.84 ± 0.02	-1.92 ± 2.16
	Persim	7.4	78.5	0.7	1.0 ± 4.97E-17	-4541.41 ± 8739.41	0.99 ± 1.98E-04	-12.18 ± 18.51	0.78 ± 0.02	-5.97 ± 8.52
	Giotto	94.4	9837.5	0.76	1.0 ± 0.0	-38.89 ± 36.49	0.99 ± 1.74E-08	-2.89 ± 4.01	0.84 ± 0.02	-2.37 ± 3.25
	Gudhi	12.6	109.7	0.71	1.0 ± 0.0	-6.76E+7 ± 1.35E+8	0.99 ± 1.11E-04	-13.91 ± 24.41	0.79 ± 0.03	-7.85 ± 13.89
PL	Perslay	8.2	5.6	0.76	0.99 ± 3.90E-12	-364.45 ± 676.82	0.99 ± 3.36E-09	-2.30 ± 1.99	0.84 ± 0.02	-3.44 ± 3.55
	Giotto	0.2	3.1	0.67	1.0 ± 0.0	-71.49 ± 134.60	0.99 ± 2.16E-09	-5.45 ± 4.17	0.83 ± 0.02	-4.99 ± 6.23
	Gudhi	0.3	14.6	0.56	1.0 ± 0.0	-72.81 ± 137.14	0.99 ± 5.67E-10	-2.03 ± 1.42	0.83 ± 0.01	-3.66 ± 4.61

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H2 for the regression in the rsfMRI pipeline.

O Results for all representations in control vs. patient classification from the T1 pipeline

					H0							
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay						No Results					
	Persim	249.5	6.9	0.51	0.52 ± 0.01	0.44 ± 0.03	1.0 ± 0.0	0.61 ± 0.08	0.73 ± 0.02	0.53 ± 0.08	0.55 ± 0.02	0.45 ± 0.08
	Giotto	80.0	938.9	1.61	1.0 ± 0.0	0.44 ± 0.03	1.0 ± 0.0	0.61 ± 0.08	0.73 ± 0.02	0.53 ± 0.08	0.55 ± 0.02	0.61 ± 0.08
	Gudhi						No Results					
PL	Perslay	7.6	0.8	1.09	0.63 ± 0.03	0.52 ± 0.11	1.0 ± 0.0	0.53 ± 0.06	0.80 ± 0.02	0.49 ± 0.07	0.60 ± 0.02	0.39 ± 0.13
	Giotto	4.4	1.4	1.11	0.52 ± 0.01	0.44 ± 0.03	1.0 ± 0.0	0.48 ± 0.07	0.80 ± 0.02	0.50 ± 0.11	0.58 ± 0.03	0.47 ± 0.11
	Gudhi	12.7	1.7	0.93	0.52 ± 0.01	0.44 ± 0.03	1.0 ± 0.0	0.47 ± 0.09	0.79 ± 0.02	0.52 ± 0.08	0.60 ± 0.01	0.42 ± 0.10

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H0 for classifying T1-w MRI as healthy control or ALS patient.

					H1							
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay						No Results					
	Persim	2510.6	20.2	0.99	0.51 ± 0.01	0.44 ± 0.03	1.0 ± 0.0	0.55 ± 0.14	0.70 ± 0.05	0.52 ± 0.11	0.61 ± 0.04	0.58 ± 0.10
	Giotto	118.3	1308.4	3.55	1.0 ± 0.0	0.44 ± 0.03	1.0 ± 0.0	0.55 ± 0.09	0.77 ± 0.04	0.62 ± 0.04	0.76 ± 0.04	0.67 ± 0.15
	Gudhi						No Results					
PL	Perslay	64.8	0.8	1.42	0.70 ± 0.04	0.53 ± 0.09	1.0 ± 0.0	0.55 ± 0.11	0.75 ± 0.03	0.50 ± 0.05	0.61 ± 0.04	0.63 ± 0.11
	Giotto	43.3	1.5	2.54	0.52 ± 0.01	0.44 ± 0.03	1.0 ± 0.0	0.50 ± 0.11	0.75 ± 0.03	0.44 ± 0.12	0.59 ± 0.06	0.50 ± 0.12
	Gudhi	129.6	1.9	1.23	0.52 ± 0.01	0.44 ± 0.03	1.0 ± 0.0	0.52 ± 0.13	0.76 ± 0.04	0.44 ± 0.12	0.63 ± 0.03	0.42 ± 0.09

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H1 for classifying T1-w MRI as healthy control or ALS patient.

					H2							
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay						No Results					
	Persim	2322.6	15.4	0.99	0.52 ± 0.01	0.44 ± 0.03	1.0 ± 0.0	0.52 ± 0.09	0.73 ± 0.04	0.47 ± 0.08	0.61 ± 0.02	0.61 ± 0.09
	Giotto	107.6	1565.7	4.05	1.0 ± 0.0	0.44 ± 0.03	1.0 ± 0.0	0.38 ± 0.08	0.77 ± 0.03	0.61 ± 0.12	0.72 ± 0.03	0.61 ± 0.19
	Gudhi						No Results					
PL	Perslay	63.9	0.8	1.39	0.64 ± 0.04	0.48 ± 0.11	1.0 ± 0.0	0.47 ± 0.15	0.72 ± 0.04	0.45 ± 0.13	0.61 ± 0.05	0.55 ± 0.18
	Giotto	42.7	1.5	2.53	0.52 ± 0.01	0.44 ± 0.03	1.0 ± 0.0	0.41 ± 0.08	0.75 ± 0.02	0.50 ± 0.07	0.56 ± 0.03	0.42 ± 0.15
	Gudhi	123.9	1.8	1.23	0.52 ± 0.01	0.44 ± 0.03	1.0 ± 0.0	0.41 ± 0.09	0.75 ± 0.02	0.50 ± 0.07	0.63 ± 0.03	0.37 ± 0.09

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H2 for classifying T1-w MRI as healthy control or ALS patient.

P Results for all representations in progression classification from the T1 pipeline

							H0					
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay						No Results					
	Persim	1062.9	62.0	0.99	0.55 ± 0.03	0.60 ± 0.21	1.0 ± 0.0	0.60 ± 0.26	0.77 ± 0.06	0.44 ± 0.16	0.56 ± 0.09	0.40 ± 0.14
	Giotto	65.5	426.1	1.71	1.0 ± 0.0	0.55 ± 0.12	1.0 ± 0.0	0.42 ± 0.20	0.60 ± 0.10	0.36 ± 0.16	1.0 ± 0.0	0.35 ± 0.16
	Gudhi	677.6	264.3	0.85	0.97 ± 0.03	0.37 ± 0.12	1.0 ± 0.0	0.46 ± 0.21	0.70 ± 0.02	0.38 ± 0.19	0.75 ± 0.07	0.48 ± 0.28
PL	Perslay	22.9	1.0	1.10	0.97 ± 0.03	0.6 ± 0.14	1.0 ± 0.0	0.33 ± 0.29	0.68 ± 0.06	0.41 ± 0.27	0.86 ± 0.06	0.28 ± 0.23
	Giotto	15.5	1.6	1.42	0.56 ± 0.02	0.50 ± 0.06	1.0 ± 0.0	0.23 ± 0.20	0.70 ± 0.07	0.41 ± 0.16	0.64 ± 0.09	0.37 ± 0.12
	Gudhi	71.9	140.1	0.95	0.56 ± 0.02	0.50 ± 0.06	1.0 ± 0.0	0.23 ± 0.20	0.70 ± 0.07	0.41 ± 0.16	0.84 ± 0.07	0.15 ± 0.20

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H0 for classifying T1-weighted MRI by progression rate.

							H1					
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay						No Results					
	Persim	909.5	96.5	0.99	0.57 ± 0.03	0.55 ± 0.12	1.0 ± 0.0	0.59 ± 0.09	0.65 ± 0.02	0.35 ± 0.14	0.59 ± 0.03	0.51 ± 0.18
	Giotto	64.1	447.9	1.90	1.0 ± 0.0	0.55 ± 0.12	1.0 ± 0.0	0.32 ± 0.26	0.60 ± 0.08	0.37 ± 0.12	0.99 ± 0.02	0.27 ± 0.25
	Gudhi	586.1	253.3	0.65	1.0 ± 0.0	0.47 ± 0.18	1.0 ± 0.0	0.47 ± 0.18	0.63 ± 0.04	0.50 ± 0.17	0.70 ± 0.10	0.29 ± 0.19
PL	Perslay	23.5	0.8	1.19	1.0 ± 0.0	0.51 ± 0.13	1.0 ± 0.0	0.51 ± 0.24	0.68 ± 0.02	0.50 ± 0.06	0.92 ± 0.07	0.45 ± 0.21
	Giotto	16.2	1.3	1.42	0.57 ± 0.03	0.55 ± 0.12	1.0 ± 0.0	0.36 ± 0.22	0.73 ± 0.09	0.41 ± 0.09	0.70 ± 0.04	0.37 ± 0.12
	Gudhi	48.8	160.8	0.95	0.57 ± 0.03	0.55 ± 0.12	1.0 ± 0.0	0.36 ± 0.22	0.71 ± 0.11	0.41 ± 0.09	0.91 ± 0.03	0.36 ± 0.16

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H1 for classifying T1-weighted MRI by progression rate.

							H2					
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay						No Results					
	Persim	1476.3	89.4	1.01	0.57 ± 0.03	0.55 ± 0.12	1.0 ± 0.0	0.50 ± 0.17	0.64 ± 0.03	0.36 ± 0.10	0.63 ± 0.04	0.51 ± 0.18
	Giotto	74.2	514.1	2.25	1.0 ± 0.0	0.55 ± 0.12	1.0 ± 0.0	0.32 ± 0.11	0.64 ± 0.09	0.33 ± 0.25	1.0 ± 0.0	0.27 ± 0.25
	Gudhi	995.5	264.2	0.67	1.0 ± 0.0	0.60 ± 0.14	1.0 ± 0.0	0.55 ± 0.20	0.74 ± 0.06	0.54 ± 0.30	0.75 ± 0.14	0.47 ± 0.25
PL	Perslay	39.3	0.8	1.06	0.99 ± 0.02	0.51 ± 0.13	1.0 ± 0.0	0.32 ± 0.21	0.66 ± 0.10	0.22 ± 0.13	0.97 ± 0.04	0.37 ± 0.27
	Giotto	27.7	1.3	1.41	0.56 ± 0.02	0.45 ± 0.12	1.0 ± 0.0	0.36 ± 0.22	0.75 ± 0.06	0.50 ± 0.25	0.71 ± 0.06	0.42 ± 0.20
	Gudhi	86.3	147.5	0.95	0.56 ± 0.02	0.45 ± 0.12	1.0 ± 0.0	0.36 ± 0.22	0.75 ± 0.06	0.5 ± 0.25	0.92 ± 0.06	0.37 ± 0.20

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H2 for classifying T1-weighted MRI by progression rate.

Q Results for all representations in onset classification from the T1 pipeline

H0												
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	1130.4	1.5	1.04	1.0 ± 0.0	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.90 ± 0.04	0.83 ± 0.15	1.0 ± 0.0	0.70 ± 0.20
	Persim	800.5	44.6	0.98	0.86 ± 0.04	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.91 ± 0.05	0.72 ± 0.28	0.86 ± 0.04	0.87 ± 0.17
	Giotto	176.5	1016.9	1.84	1.0 ± 0.0	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.83 ± 0.05	0.69 ± 0.17	1.0 ± 0.0	0.69 ± 0.17
	Gudhi	1045.0	295.3	0.83	1.0 ± 0.0	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.92 ± 0.04	0.78 ± 0.13	0.93 ± 0.09	0.72 ± 0.17
PL	Perslay	24.5	1.0	1.27	0.99 ± 0.02	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.90 ± 0.04	0.87 ± 0.17	1.0 ± 0.0	0.78 ± 0.13
	Giotto	16.0	1.2	1.43	0.86 ± 0.04	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.86 ± 0.05	0.78 ± 0.13	0.90 ± 0.04	0.87 ± 0.17
	Gudhi	75.4	360.8	0.96	0.86 ± 0.04	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.86 ± 0.05	0.78 ± 0.13	0.93 ± 0.02	0.78 ± 0.13

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H0 for classifying T1-weighted MRI by onset site.

H1												
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	1224.5	1.3	1.06	1.0 ± 0.0	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.86 ± 0.06	0.73 ± 0.17	1.0 ± 0.0	0.64 ± 0.19
	Persim	815.5	93.4	0.62	0.86 ± 0.04	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.92 ± 0.03	0.77 ± 0.13	0.86 ± 0.04	0.87 ± 0.17
	Giotto	112.0	647.8	2.04	1.0 ± 0.0	0.87 ± 0.17	1.0 ± 0.0	0.83 ± 0.15	0.83 ± 0.05	0.73 ± 0.17	1.0 ± 0.0	0.69 ± 0.17
	Gudhi	882.5	283.4	0.49	1.0 ± 0.0	0.87 ± 0.17	1.0 ± 0.0	0.83 ± 0.15	0.85 ± 0.08	0.72 ± 0.28	0.91 ± 0.09	0.72 ± 0.28
PL	Perslay	23.8	0.8	1.19	1.0 ± 0.0	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.90 ± 0.06	0.82 ± 0.16	1.0 ± 0.0	0.74 ± 0.15
	Giotto	15.5	1.0	1.43	0.86 ± 0.04	0.87 ± 0.17	1.0 ± 0.0	0.82 ± 0.16	0.87 ± 0.04	0.78 ± 0.13	0.91 ± 0.03	0.87 ± 0.17
	Gudhi	67.2	361.2	0.96	0.86 ± 0.04	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.87 ± 0.04	0.78 ± 0.13	0.94 ± 1.60E-03	0.78 ± 0.13

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H1 for classifying T1-weighted MRI by onset site.

H2												
					SVC		**Random Forest**		**KNeighbors**		**Logistic Regression**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**	**Train accuracy**	**Test accuracy**
PI	PersLay	1896.8	1.3	1.46	1.0 ± 0.0	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.85 ± 0.03	0.69 ± 0.17	1.0 ± 0.0	0.59 ± 0.24
	Persim	1406.8	80.0	0.65	0.86 ± 0.04	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.85 ± 0.06	0.77 ± 0.20	0.86 ± 0.04	0.87 ± 0.17
	Giotto	133.2	404.3	2.41	1.0 ± 0.0	0.87 ± 0.17	1.0 ± 0.0	0.79 ± 0.18	0.83 ± 0.05	0.73 ± 0.17	1.0 ± 0.0	0.74 ± 0.15
	Gudhi	1481.9	282.3	0.57	1.0 ± 0.0	0.87 ± 0.17	1.0 ± 0.0	0.83 ± 0.15	0.88 ± 0.04	0.69 ± 0.31	0.90 ± 0.09	0.67 ± 0.25
PL	Perslay	41.6	0.8	1.31	1.0 ± 0.0	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.90 ± 0.02	0.79 ± 0.18	1.0 ± 0.0	0.69 ± 0.17
	Giotto	27.4	1.0	1.42	0.86 ± 0.04	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.87 ± 0.04	0.82 ± 0.16	0.86 ± 0.04	0.87 ± 0.17
	Gudhi	156.3	349.8	1.0	0.86 ± 0.04	0.87 ± 0.17	1.0 ± 0.0	0.87 ± 0.17	0.87 ± 0.04	0.82 ± 0.16	0.99 ± 0.02	0.83 ± 0.15

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H2 for classifying T1-weighted MRI by onset site.

R Results for all representations in regression from the T1 pipeline

						H0				
					Linear		**Gradient Boosting**		**Random Forest**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**
PI	PersLay	315.1	6.3	21.73	0.99 ± 7.49E-10	-205.91 ± 379.31	0.99 ± 4.82E-07	-1.54 ± 1.58	0.84 ± 0.03	-1.04 ± 1.21
	Persim	246.1	134.6	0.94	0.99 ± 1.76E-14	-1.22E+04 ± 2.42E+04	0.99 ± 2.40E-04	-0.81 ± 0.61	0.84 ± 0.02	-0.72 ± 0.58
	Giotto	85.7	16258.5	1.35	1.0 ± 0.0	-280.70 ± 327.98	0.99 ± 1.48E-07	-1.25 ± 1.54	0.85 ± 0.03	-0.84 ± 1.25
	Gudhi	196.7	125.1	0.95	0.99 ± 5.10E-14	-546.26 ± 953.64	0.99 ± 5.69E-04	-1.33 ± 1.13	0.83 ± 0.01	-1.10 ± 1.28
PL	Perslay	6.4	0.8	1.58	0.76 ± 0.07	-998.95 ± 1329.81	0.99 ± 5.30E-05	-1.26 ± 1.02	0.81 ± 0.02	-0.79 ± 0.61
	Giotto	4.83	5.7	1.04	1.0 ± 0.0	-45.02 ± 79.74	0.99 ± 7.48E-05	-1.24 ± 1.16	0.81 ± 0.03	-0.72 ± 0.37
	Gudhi	29.6	35.9	0.87	1.0 ± 0.0	-43.03 ± 75.88	0.99 ± 5.49E-05	-1.36 ± 1.24	0.81 ± 0.02	-0.77 ± 0.41

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H0 for the regression in the T1 pipeline.

						H1				
					Linear		**Gradient Boosting**		**Random Forest**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**
PI	PersLay					No Results				
	Persim	2475.0	137.4	1.19	0.99 ± 3.51E-14	-6158.46 ± 1.19E+04	0.99 ± 7.24E-05	-0.73 ± 0.90	0.86 ± 0.02	-0.47 ± 0.80
	Giotto	115.4	14379.3	1.97	1.0 ± 0.0	-508.63 ± 859.89	0.99 ± 2.51E-07	-1.00 ± 0.82	0.84 ± 0.02	-0.88 ± 1.09
	Gudhi	3551.6	128.5	0.03	0.99 ± 4.96E-15	-4436.76 ± 7774.89	0.99 ± 1.70E-04	-2.58 ± 3.56	0.83 ± 0.02	-0.93 ± 1.33
PL	Perslay	61.6	0.8	8.3	0.81 ± 0.06	-80.22 ± 154.59	0.99 ± 2.59E-05	-0.91 ± 0.82	0.84 ± 0.03	-0.56 ± 0.50
	Giotto	48.3	4.4	1.90	1.0 ± 0.0	-6.15 ± 6.95	0.99 ± 3.34E-05	-1.61 ± 2.04	0.85 ± 0.03	-0.73 ± 0.59
	Gudhi	324.4	37.5	0.64	1.0 ± 0.0	-6.19 ± 6.98	0.99 ± 3.65E-05	-1.37 ± 1.92	0.85 ± 0.03	-0.78 ± 0.61

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H1 for the regression in the T1 pipeline.

						H2				
					Linear		**Gradient Boosting**		**Random Forest**	
	Library	**Time converting PD to PI/PL (s)**	**Time training models (s)**	**RAM Usage (GB)**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**	**Train coefficient**	**Test coefficient**
PI	PersLay					No Results				
	Persim	2449.8	139.3	1.19	0.99 ± 1.01E-13	-3248.05 ± 5744.43	0.99 ± 6.58E-05	-0.41 ± 0.55	0.85 ± 0.01	-0.23 ± 0.38
	Giotto	113.1	14595.2	2.07	1.0 ± 0.0	-4.06E+04 ± 8.09E+04	0.99 ± 5.5E-08	-2.70 ± 4.56	0.84 ± 0.03	-1.31 ± 2.30
	Gudhi					No Results				
PL	Perslay	58.3	0.8	8.30	0.76 ± 0.06	-44.46 ± 51.83	0.99 ± 3.00E-05	-3.34 ± 3.91	0.82 ± 0.03	-1.65 ± 2.06
	Giotto	47.4	4.5	1.89	1.0 ± 0.0	-8.73 ± 8.84	0.99 ± 1.10E-05	-2.70 ± 2.47	0.82 ± 0.03	-1.84 ± 2.25
	Gudhi	260.1	38.8	0.73	1.0 ± 0.0	-8.80 ± 8.92	0.99 ± 2.55E-05	-2.56 ± 2.18	0.82 ± 0.04	-1.93 ± 2.46

Comparison between the different libraries for converting the persistence diagrams (PD) to persistence representations, including persistence images (PI) and persistence landscapes (PL), in dimension H2 for the regression in the T1 pipeline.