University of Pisa

Department of Physics E. FERMI

Master degree in physics

Functional connectivity measures using machine learning techniques

Supervisors:

Prof. Alessandra Retico

Prof. Piernicola Oliva

Candidate:

Federico Campo

Academic year: 2021/2022

Contents

Abstract

Autism spectrum disorder (ASD) is a neurodevelopmental disorder manifesting from the early ages and involving behavioral and cognitive impairments. So far, no univocal biomarker have been detected to identify this disorder, and the most accurate diagnostic tool remains a behavioural and attitudinal test. In this work we focus on brain functional connectivity data to study the difference between patients with ASD and healthy controls. To tackle this challenge, we employed machine learning and deep learning models. Our database are Magnetic Resonance scans provided by the ABIDE dataset consisting on structrural and functional scans of approximately 2200 subjects collected from different medical centers. One half of them belong to normal individuals and the other half to people with diagnosed ASD.

Starting from these data, the workflow of this thesis can be summarised in the following steps:

- MRI and fMRI data preprocessing and normalization to a common template, and extraction of temporal series of different brain areas for each patient from the fMRI scans.
- Creation of a connectivity map for each patient. This connectivity matrix is created computing a correlation coefficient for every pair of timeseries of a single patient. Correlation coefficients were computed using two different approaches: a linear correlation of the two timeseries known as Pearson correlation, and a correlation based on time-frequency analysis of a timeseries pairs by means of wavelet transform.
- Implementation of a harmonization procedure to correct for potential biases and distinctive traits of data, linked to acquisition scan procedures employed in a medical center which can differ between different centers.

- Study and classification of connectivity matrices with deep learning methods. The two different connectivity matrices described above are compared to assess which one is able to yield the best separability between Controls and ASD. Classification of raw data and harmonized data implemented with different procedures are compared, and in addition a harmonization implemented inside a deep learning model through an adversarial learning technique was tested.

- Deep learning model is compared to a Random Forest classifier, a standard machine learning model, to determine if a deep model brings a benefit to this kind of analysis.

- After the definition of the appropriate harmonization procedure, we implemented an interpretation method called SHAP to explain which features contributed the most to the final outcome of machine learning classification, and from them we determined the most relevant brain areas that allow a discerning between controls and ASD

Organization of contents

Background: In this chapter we explain relevant theory arguments that lay the groundwork to all the thesis work. In chapter 1 we outline the main problems associated with the Autism Spectrum Disorder, from the biological development to the social isseues it generates. We briefly review some studies carried on so far to study this challenging issue. We also provide a brief overview of the main techniques employed to study this issue in this work: from data acquisition to data analysis.

Chapter 2 provides an overview of the basic physical principles of MRI imaging and the main acquisition sequences. It also includes a description of the physical and biological principles underlying fMRI diagnostic tool which is the means through which data we are going to employ are collected.

Chapter 3 provides a brief explaination of machine learning and deep learning models diving into the mathematical formulation of the main concepts. We aim to give the idea of the structure of a machine learning models, the learning procedure, and the important metrics for a proper evalutation of model performances. Ultimately, we describe a common way to reduce data dimensionality keeping as much information as possible and reducing the number of uninformative or reduntant features from data.

Chapter 4 describes the theory behind an important strategy to inspect a deep model and understand the processes that brought it to a certain output given some input data.

MATERIALS & METHODS: This section describes the instruments we used to run the analysis: dataset, preprocessing steps, and data preparation employed to obtain correlation matrix.

Chapter 5 describes and analyze ABIDE dataset: a publicly available collection of more than 2000 scans of healthy and ASD patients, collected from different medical centers.

Chapter 6 provides an overview of the most important preprocessing steps to normalize and align all these scan to a common template to run meaningful statistical analysis and to exctract brain timeseries form different brain areas aligned to a common coordinate system. We describe C-PAC: the software employed to this image preprocessing and how we used it to conduct our processing.

Chapter 7 describes the two methods we used to calculate connectivity coefficients and create a functional connectivity matrix for each patient. We describe the computation of correlation using Pearson correlation, and illustrate the formalism of wavelet transform for time-frequency analysis and the process to extract a correlation coefficient from time-frequency data of two timeseries.

Chapter 8 describes the regression model upon which harmonization procedure is based. This is an analytical method to remove inter-site variability from our data and avoid biases towards the acquisition source.

Chapter 9 describes the structure and the general idea beyond a domain adversarial neural network: a deep neural network that aims to make Control/ASD classification following a procedure that makes data site-independent to classify following a procedure that makes the classification Control/ASD output sites-independent.

Chapter 10 describes the main aspects of an important algorithm for machine learning model explaination called SHAP. We discuss the general idea behind its implementation and important quantities to determine the contribution of a feature to the output of a machine learning model.

Implementation & RESULTS: In this last section, the results obtained applying methods to our materials are presented. We illustrate the results of harmonization applied to our data and classification result of Pearson-based correlation coefficients and wavelet-based coefficients and in the end we study the most relevant feature from our data, that played an important role in discrimination of Control/ASD. Chapter 12 Presents the results of the analytical harmonization to our data to have a visual understanding of how this process affects a feature distribution. Chapter **??** shows all the classification results obtained with a deep neural network on harmonized and non-harmonized (raw) data and the results of the domain adversarial neural network and compares them with results obtained using a simpler Random Forest classifier. Results of data dimensionality reduction are presented as well to assess if this is a meaningful procedure to reduce dimensionality on

this dataset. At the end, chapter 15 shows the implementation of a feature importance assessment procedure and the results obtained with machine learning in the previous chater. We show what feature contributed the most to the outcome of a prediction, and then to the discrimination of healthy and ASD patients, and subsequently we study what brain areas are mainly involved in this discernment.

Chapter 1

Introduction

1.1 Autism spectrum disorder

Autism spectrum disorder (ASD) is a neurological disorder that is becoming a growing social issue, drawing more and more attention especially during the last decades, for its social impact on families and society and the raise in number of diagnosed cases. ASD is classified as a neurodevelopmental behavioural disorder [25] [35] and refers to a broad range of conditions manifesting as deficits in social communication and interaction such as reduced sociability or empathy, repetitive behaviours, resistance to changes, and sometimes even speech difficulties.[40] To the present days, ASD is diagnosed through a comportamental assessment test since there is not a definite biological test. Early signs start appearing by the age of 2 - 3, ages at which children usually start interacting with parents and with other children. Parents are asked to answer a set of questions about every-day behaviour of their children like the checklist for Autism in Toddler [45] consisting on a list of question such as "When you point at something, does your child watch in that direction? "or "does your children show interest towards other children? ". However, are not uncommon cases where ASD is discovered only after the adolescence. Currently ADI-R (Autism Diagnostic Interview - Revised) and ADOS (Autism Diagnostic Observation Schedule) are considered the 'gold standard' tools for diagnosis of ASD [36] [13].

Nevertheless, besides these behavioural test, to achieve a more accurate diagnosis, medical and biological informations should be taken into account as well. For a better treatment of this condition,

it would be of great importance to make an early diagnosis, in order to provide in time the help and support people need. Hopefully, treating children when this disorder is still it in its early stage, could bring to a regression of this condition.

The main biological factors that bring to the developing of autism aren't very clear so far, and is acquiring more consensus between neuroscientists and medical doctors that there is not just a single cause, but a concatenation and coexistence of various triggering factors. Among them, the most strongly suspected are identified to be genetic and environmental, where environmental is to be intended as exposure to air pollution. Regarding these environmental causes, it is suspected that during pregnancy, especially during the first days of embrional development (day 20-24 of gestation), exposition of the mother to air pollution is linked to an increased risk for her child to develop this disorder.[42]

Genetic origin of this disorder is assiciated with a rare gene mutation, such as a deletion, duplication or inversion, and even though most of the mutation that increase the risk of developing autism are not been traced. It has been assessed though, that it has an high inheritance trait. From studies on twins it has been assesses that between omozygotes twins there is around 90 % or probability that both of them develop ASD, while for heterozygote this probability falls to around 7% for men and 1-2% for women [22]

From a neurological point of view, ASD may affect a brain both in its structure and in its functionality.

Structural studies usually focus on volumentric and morphometric analyzes to examine differences in brain anatomy. It has been studied how autism could alter the symmetry between the two emispheres [**?**] of the brain. In children it has been observed an increase in total brain volumne as well asan enlargment of the left superior temporal gyrus, but this trend is not well defined for older ages.[44].

Functional neuroimaging researches mainly focus on impaired functionality. Different studies pointed out a reduced information processing due to synaptic disfunction that manifests in a reduced or altered brain functional connectivity. Functional connectivity measures aim to describe statistical dependencies between brain areas in time, by studying temporal correlation between dif-

ferent brain parts even between those which are spatially separated. Altered brain activity is found by different but controversal studies because both hyper- functional connectivity and hypo-FC were detected in ASD patients, while some found a the coexistence of an hyper- and hypo-functionality between different areas of the brain. In particular it appears functionality abnormalities to have an age-dependent trend since over connectivity is usually observed in young children while under connectivity in adolescences and adults.[49] [48].

1.2 fMRI as investigation tool

In the last decades, different approaches to study ASD have been proposed, to find a relation between different brain areas involved in lower or higher functional connectivity. Different diagnostic tools are being used to investigate this matter, from different perspectives and at different scales, such as magnetic resonance imaging (MRI), functional-MRI (fMRI), or electroencephalography (EEG).

Magnetic resonance imaging is employed to obtain images of brain both for structural studies and for functional through the technique of functional MRI (fMRI). An other tool employed to study temporal signal is the EEG which differs from fMRI because of the type of signal it aims to detect and because of the different resolution, since with EEG is possible to study signal in scale time comparable with the neuronal activation time (timescale $\sim \mu s$), while with fMRI we look for a signal due to the hemodynamic response following a neuronal activation (timescale $\sim s$).

fMRI is a diagnostic tool that investigates physiological changes linked to blood flow variations across the whole brain structure. Measures of blood properties are strictly linked with measures of the neuronal activity: an increase of neuronal activity leads to a boost in blood flow and an increase of vessel's size within a specific brain area, because of the bigger demand of oxigen and other nutrients to the neurons in that area.

The different neural connections are identified by measuring the BOLD fluctuations, (acronym for blood-oxigen-level dependent) from different areas of the brain, using the signal difference between oxy (diamagnetic) and deoxy-hemoglobin (paramagnetic). These spontaneous fluctuations in the BOLD signal during resting state are considered a strong indicator for the assessment of the properties of the brain system.

fMRI can be performed during the resting state of a patient or during a task. We refer to these procedures as rs-fMRI and task-based fMRI, and while task-based fMRI is a good instrument to investigate the local activation of a single brain area during a task, resting state fMRI is drawing more attention for the investigation of ASD. It was proven to be suitable for examining functional connectivity among all the brain regions, and highlight the difference between a normal brain and one affected by this disorder.

Resting state fMRI is an acquisition carried out while the patient is in a relaxed state, and is not performing any active task, but he is simply asked to stay still with eyes closed or open while fixating a cross. Following this setup, the brain is at rest and its activity is not perturbed by active tasks, such as moving an arm, or passive actions, like being exposed to different visual stimuli, which are common activities for task-based functional connectivity, a different acquisition setup to investigate BOLD fluctuations between subjects in a task-stimulated state and control states. Resting state setup revealed to be a powerful tool to investigate the intrinsic generated brain activity and study the altered functional connectivity networks in subjects with mental disorder

1.3 Machine learning to deal with non-linear problems

During the study of characteristic traits between control and ASD patients, different attributes are involved and affect data in a strong and evident way, the most important of which are age, sex and FIQ (full intelligence quotient). Age, for example, affects both structural and functional data, with an observed overgrown of the brain volume and hyper-connectivity, in toddlers, that are both traits that tend to decrease going in on age. Moreover, on control subjects, even though it is not well characterized, seem to show a decreasing of brain structure and functional connectivity in older ages [57]. Sex is the other feature that gives characteristic traits to data. Studies show that some brain areas exhibit an increased functional connectivity in female. And, from a combined analysis of both age and sex, it appears that in men some brain structures show more pronounced aging effect than women [11].

In addition, if we limit our focus just to functional data, the eye status at scan plays an important role. Functional connectivity with open steady eyes results to be different from closed eyes,

with strong differences and higher connectivity in different brain areas between the two cohorts of subjects.[12].

Furthermore, an other aspect to be take into account regarding closed eye patients, is that during the scan acquisition, they may fall asleep, and this would heavily modify the functional brain activity, resulting in a modified functional connectivity pattern.

It appears clear that the distinction of ASD patients among normal ones is not a straightforward task and a simple univariate analysis would not be sufficient, because of the presence of different confounding factors contributing to the characterization of data.

To take on this challenge, one of the most promising tools that allow us to deal with complex, non-linear problems is the use of machine learning algorithm to study data extracted from fMRI such as pairwise correlations between different regions of the brain, or image of brain themselves from MRI scans to studies with convolutional neural networks (CNN). Machine learning belongs to artificial intelligence tools and make use of algorithms that learn from data, and modify their parameters with the aim of recognizing distinctive properties from data and make predictions or classification, on new unseen data, trying to gradually reduce the error and make a more accurate predictions.

A relevant aspect that makes machine learning and artificial neural networks so popular is the ability to deal with non-linear problems, by introducing several non-linear functions during training. This allows to obtain non-linear outputs from each input data, which would hopefully help in the distinction between characteristic traits of healthy people and of ASD patients.

As a drawback, a restraint of machine learning is that, to perform well, an algorithm needs to be trained on datasets of large dimension, because the bigger the dataset, the more the algorithm is able to generalize information and the better are its performances on new data. This is true for all the kind of machine learning algorithms and especially for Deep Neural Networks which are the principal kind of algorithms we are going to employ in this work.

1.4 The need for a harmonization procedure

To achieve the goal of obtaining a large dataset, particularly in medical field where data are not easily available from a single center or are not in a sufficient amount to perform a large-scale analysis, data from different acquisition centers need to be put together. In the last years, this issue is becoming increasingly popular and several multicenter medical datasets such as the Human Connectome Project, the Alzheimer's Disease Neuroimaging Initiative (ADNI) or Autism Brain Imaging Data Exchange (ABIDE) were created to obtain large collections of data to make significant statistical analysis.

Unfortunately, this procedure brings with it a downside regarding the unavoidable bias towards the site that data belong to. This bias is a result of hardware and scan procedure differences between centers and it is not feasible to ask to all the medical centers across the world to uniform to a single common acquisition protocol.

To work out this problem and try to uniform inter-scan variability, we need a harmonization procedure to remove site-dependent information, leaving all the rest of important information unchanged.

In this work, two different approaches are proposed: an analytical harmonization and a deep learning approach.

Analytical harmonization is a procedure that modifies data with shifts and rescaling to remove only inter-site related effects in multi-site data, while trying to preserve all other information as biological-related effects.

The second approach, the deep learning one, tries to make the classification of control/ASD data extracting from input data both information related to the class (Control/ASD) and to site. Then it uses the latter to remove the bias linked to sites before making a prediction.

Chapter 2

Principles of Magnetic Resonance Imaging

In this chapter we report some basic principles that underlie the Nuclear Magnetic Resonance Imaging. Since the matter can become very complex, we only give some glimpses of what are the main principles behind this technique and the main acquisition sequences employed in diagnostics.

2.1 Physical principles

Nuclear magnetic resonance imaging, is a non-invasive imaging technique widely employed in radiology to obtain anatomical images or to study physiological processes, taking advantage of its flexible sensitivity to different tissues.

The physical principle of this technique is the interaction of a nuclear spin of any tissue molecule, with an external magnetic field $\mathbf{B}_0$. Molecules that contribute the most to the signal are water, and specifically, hydrogen nuclei are the dominant source of signal in MRI.

When a nucleus is subjected to an external magnetic field, the interaction would make the nuclei align with the magnetic field, but, since protons and nuclei have an

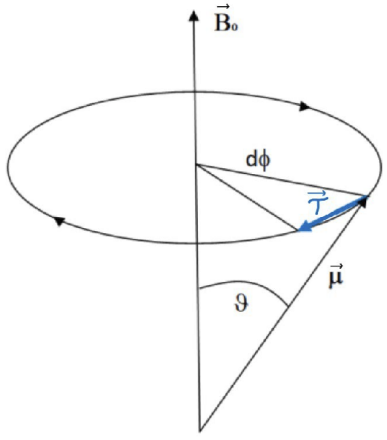


Figure 2.1: *Representation of a angu-*

intrinsic spin, and hence an angular moment $\overrightarrow{\mu}$, they start precessing.

Precession occurs because of the torque created by the interaction of a static magnetic field with the angular moment of a spinning nucleus. The relation is given by

$$\overrightarrow{\tau} = \overrightarrow{\mu} \times \overrightarrow{B_0} \tag{2.1}$$

Precession occurs around the symmetry axis given by the direction of the magnetic field $\overrightarrow{B_0}$ as schematically shown in figure **??**.

When a nucleus starts this rotatory movement, the characteristic frequency ω_0 is uniquely determined by the properties of the nucleus itself and by the strength of the magnetic field; the relation is given by the Larmor equation and the frequency ω_0 is called **Larmor frequency**

$$\omega_0 = \gamma B_0 \tag{2.2}$$

where γ is a constant called *gyromagnetic factor* and is it characteristic of each nucleus. The gyromagnetic factor of a proton in water has a value of $\gamma = 2.7 \cdot 10^8 \frac{rad \cdot s}{T}$. For a typical scanner with a magnetic field of 2-3 T, the Larmor frequency assumes values of $\sim MHz$ just below radio waves.

When protons in a body are exposed to a magnetic field, they can assume two states in relation to the direction of the external field: parallel and anti-parallel (as an example in figure **??** a parallel configuration is represented).

Proton with different orientation with respect to the magnetic field have different energies. The potential energy of a particle with magnetic moment $\vec{\mu}$ immersed in a magnetic field $\vec{B_0}$ is given by

$$U = -\vec{\mu} \cdot \vec{B_0} \tag{2.3}$$

If we suppose $\vec{B_0}$ along the z axis, it will have only component along z, so the scalar product will just concern the z component of the magnetic moment μ_z. Recalling the relation between the angular

moment and the magnetic moment $\vec{\mu} = \gamma \vec{J}$, and the quantization of the angular moment $J_z = m_s \hbar$. m_s represents the spin of the proton and can assume values $-\frac{1}{2}$ or $+\frac{1}{2}$. The potential energy of a proton when subjected to a magnetic field can be written as

$$U = -\vec{\mu} \cdot \vec{B_0} = -\mu_z B_z = -\gamma m_s \hbar B_z = -m_s \hbar \omega_0 = \pm \frac{1}{2} \hbar \omega_0 \tag{2.4}$$

. In the formula above, energy with negative sign (associated to proton with spin $+1/2$) corresponds to proton aligned in parallel with the magnetic field, while energy with positive sign belongs to proton in an anti-parallel state. For a thermodynamic system, with a given temperature T, the probability to find particles with energy ϵ is given by the Boltzmann distribution

$$P(\epsilon) = \frac{1}{Z} e^{-\frac{\epsilon}{kT}} \tag{2.5}$$

where k is the Boltzmann's constant, and Z is the partition function of the system $Z = \sum_\epsilon e^{-\epsilon/kT}$. Using this relation we can compute the difference between the number of proton in the two states parallel and anti-parallel

$$\Delta N = N^+ - N^- = \frac{N}{Z}(e^{-\frac{\hbar\omega_0}{kT}} - e^{\frac{\hbar\omega_0}{kT}}) \approx \frac{N}{2}\frac{\hbar\omega_0}{kT} \tag{2.6}$$

where the approximation of the exponential since for human body is $\hbar\omega_0 \ll kT$. What we obtain with this formula is that the number of parallel and anti-parallel protons is not the same, and the reason behind this comes from energetic considerations since protons with spin parallel to the magnetic field have a lower potential energy. This imply that when a body is exposed to a magnetic field, this small excess of number of protons in an parallel state cause the body to have an intrinsec magnetization $\vec{M}$.

This net magnetization can be represented by a single vector pointing parallel to the direction of the external field and if we want to detect a signal related to this magnetization, we have to perturb this vector from its equilibrium state. The perturbation would cause the precession of this magnetization vector around the direction of the magnetic field, with its own Larmor frequency. This precession will produce a changing magnetic flux which can be detected by an external coil. To kick this vector away from its equilibrium state, a second external magnetic field is employed under the form of a radiofrequency impulse, with a frequency resonating with the Larmor frequency

of the precessing spins. This radiofrequency impulse is usually applied to rotate spins in a direction ortogonal to the static magnetic field $\vec{B_0}$ which we can suppose to be along the z axis $\vec{B_0} = B_0\hat{z}$ We create this way a magnetization component onto the x-y plane, aligned with the direction of the radiofrequency pulse. This magnetization vector on the transverse (x-y) plane is called *transverse magnetization*,and can be denoted as $\vec{M_\perp} = M_x\hat{x} + M_y\hat{y}$. After this radio frequency pulse, protons will start rotating in the x-y plane, but, since this impulse is not persistent, after it is switched off they gradually lose their initial energy and tend to realign along the z-axis.

This process of realigning along the z-axis is called longitudinal relaxation (longitudinal in relation to the original $B_0\hat{z}$ direction).

The evolution of the longitudinal and transverse magnetization are modeled with the introduction of two time constant T_1 and T_2 respectively. Longitudinal relaxation occurs because the system goes from an higher energy state (when it is flipped on the x-y plane) to a state of thermodynamic equilibrium with its surrondings, regaining its previous magnitude.

This process follows an exponential decay in time shown in equation 2.7, with a time constant T_1, which for the physical reason underlying it, is also called spin-lattice relaxation time.

$$M_z(t) = M_0(1 - e^{-t/T_1}) \tag{2.7}$$

The transverse magnetization vector follows a different evolution. When studying the evolution of $\vec{M_\perp}$, an other effects has to be taken into account: the spin-spin interaction between protons. If we consider a physical system where protons are immersed in a magnetic field, each proton interacts with this external field plus all the small fields created by the surrounding protons and their associated spins. This leads to the presence of a local field for each proton, slightly different from the external magnetic field $\vec{B_0}$. These different local fields affect the evolution of the transverse magnetization because they cause a relative dephasing of protons among each other. According to this, if immediately after the RF-impulse all the spins can be imagined as aligned to the x axis, in time they start to fan out. This effect speeds up the loss of the total transverse magnetization on the x-y plane.

If we study this evolution from the rotating reference frame, (rotating with the same Larmor

frequency of protons) the module of the transverse magnetization follows an exponential decay in the form

$$M_{\perp}(t) = M_{\perp}(0)e^{-t/T_2} \tag{2.8}$$

.

Since T_2 constant comprises the spin-spin interaction process plus the spin-lattice interaction which cause the realignment of spins along the $\hat{z}$ axis, it is always smaller than T_1. For protons in human tissue, T_1 ranges from 10 to 100 milliseconds, while T_2 is usually of the order of 10 milliseconds.

In a real phyisical system, however, there is and additional dephasing source, coming from external field inhomogeneities. This effect is often taken into account by the introduction of a different decay time T_2' which along with T_2 bring to a overall time costant given by

$$\frac{1}{T_2*} = \frac{1}{T_2} + \frac{1}{T_2'} \tag{2.9}$$

2.2 Image acquisition and k-space

The physical process underlying signal acquisition is Faraday induction, according to which an electrical potential, called electromotive force, is generated by the variation of a magnetic flux over time $emf = -\frac{d\Phi}{dt}$ being Φ the varying flux through the receiving coil.

A simple setup in MRI to generate a varying flux over time is the application of a single RF in order to flip protons by an angle of $\pi/2$ with respect to the direction of the magnetic field $\vec{B_0}$. The variation of magnetic flux occurs while tipped spins are rotating in the x-y plane and the magnetization vector is relaxing towards the longitudinal axis. This induced emf signal can be detected by properly oriented and tuned coils. This simple experiment is referred as Free Induction Decay (FID) and is usually performed in any MRI scan to tune RF coils and optimise system response.

Usually the varying flux we are interested in is the one along the transversal plane. The reasons behind this are mainly two: 1. the weak signal along the longitudinal axes would be saturated by the strong magnetic field of the static magnetic field $\vec{B_0}$. 2. The signal coming from the transverse

magnetization $M_\perp$ is representative of all the information we need for the analysis of a material: T_1, T_2 and the proton density ρ.

To properly create an image based on these information we need to spatially encode each signal received by the changing magnetic flux. In order to relate a signal with a spatial position, in addition to the initial static magnetic field $\vec{B_0}$ we have to place a second field which causes a controlled local modification of the magnetic field $\vec{B_0}$. This additional field $\vec{B'}$ needs to be non-uniform and to have a lower magnitude of $\vec{B_0}$ for each point.

Its distribution follows a spatial gradient so that the total magnetic field along $\hat{z}$-axis is given by the sum of this two contributes, and the signal contains space-varying frequency components according to equation 2.2 which can be rewritten as

$$\omega(z) = \gamma B(z) \tag{2.10}$$

being z the spatial coordinate and where $B(z)$ is the total magnetic field now given by the relation

$$B(z,t) = B_0 + z \cdot G(t) \qquad G(t) = \partial B'/\partial z \tag{2.11}$$

This gradual changes in magnetic field causes different parts of the body along the z axis to have different Larmor frequencies. This gradient along the longitudinal direction of the static field $\vec{B_0}$ is usually referred as Slice Selection gradient G_{ss}. Choosing the z axis as the direction of the slice selection, we acquire images on the transversal x-y plane.

When acquiring an image, data are collected under the form of a matrix called k-space. K-space is a coordinate system used to store spatial frequencies information. From these informations we can retrieve the usual MRI image (containing spatial, anatomical informations) by applying the inverse Fourier 2D transform.

In a k-space matrix, low spatial frequencies, corresponding to large object across the whole real image, are encoded at the center of the matrix and high spatial frequencies corresponding to small objects and finer details, are encoded in the peripheries.

The construction of k-space is done step by step in relation to the gradient applied time by time, producing a trajectory on the k-space.

To acquire spatial information in the x-y plane, we need to introduce two new gradients (one for each direction): a readout (or frequency encoding) gradient and a phase encoding gradient.

The frequency encoding gradient acts on one direction in the transversal plane; let's identify this direction as the x axis for a clearer visualization. Just like the slice selection gradient, it consists of a linear changing magnetic field to modify the Larmor frequency of the spins along the x axis. According to the acquisition technique we want to perform, it can be applied forward and after a while, reversed. This allows a refocusing of all the dephased spins due to spin-spin interaction; we will see more in detail this concept when we will discuss acquisition sequences, for now, we just need to know that the refocusing of all the dephased spins occurs after a time interval called echo time (TE).

The phase encoding gradient acts the exact same way of the previous one, but it acts on the y axis. It is switched on and acts by modifying the Larmor frequency but it is just a temporary change. When it is switched off, all the spin along this direction contrinue precessing all at the same frequency, but with a relative phase between them.

The combined action of these two gradients allow us to acquire different lines of the k-space.

For further and more detailed information we refer to the book [8]

2.3 Acquisition sequences

Depending on the type of analysis we want to carry out, we would like to focus on some magnetic properties rather than others. For example, for a an anatomical, structural image, we may be interested in a good distinction of different tissue, while in a functional scan we are interested in detecting chemical fluctuations.

When talking about acquisition sequences, two main parameters drive the main differences between different techniques. For the moment we define them, but a contextualized use can help to clarify their meaning. This can be found in the following sections describing different acquisition sequences.

- **Repetition time (TR)** is the time interval between one RF pulse and the next.

- **Echo time (TE)** is the time interval between a RF pulse and the echo peak

Changes in TR, TE and RF pulses characteristics (such as angle or numbers), allow to perform different acquisition sequences and to emphasise one of the three fundamental parameters T_1, T_2 or the proton density ρ of different tissues.

As a general rule

- T_1-weighted images are acquired with small values of TR. This avoid that all nuclei are back to their longitudinal position, allowing a better tissue contrast and TE has to be very short to avoid contribution due to T_2-related effects.
- T_2-weighted images are acquired with high values of TR ($TR \gg T_1$) and values of TE comparable with T_2 time constant.
- To enhance the spin density contrast, TR has to be chosen as long as possible while TE needs to be short.

2.3.1 Spin-echo sequence

Spin echo sequence is one of the most used one. It lets us retrieve lost information due to spin dephasing caused by field inhomogeneities.

The spin echo method employs two RF pulses: the first flips protons by an angle of $\pi/2$ with respect to the $\vec{B_0}$ direction and the second of π in relation to the direction of the first impulse. All the sequence can be summarized by the following steps:

1. The first radio-frequency pulse is applied with an angle of $\pi/2$ in relation to the direction of the external magnetic field $\overrightarrow{B_0}$. This process as mentioned in section 2.1 causes the spin to flip onto the transversal plane x-y. Just as an example we imagine they are flipped into the x direction. They gradually start dephasing both because of spin-spin interactions and small external field inhomogeneities, which are different from point to point. Because of this, they start fanning out, because some rotate faster and some are delayed with respect to the average magnetization vector.

2. At the instant t = τ the second rf impulse is sent. It is created in order to flip protons by angle of π with respect to their direction after the first impulse, so if the first impulse flips them along the $\hat{x}$ direction, this overturns them along $-\hat{x}$. This has the effect of turning all spins with a phase ϕ to a phase $\pi - \phi$.

3. Dephasing spins continue accumulating a phase but since they were flipped by 180 degrees, the same process according to which they were previously fanning out, push them to converge. In fact, in an interval Δt two spins have accumulated a phase $\Delta\phi(t + \Delta t) = -\gamma\delta B\Delta t$, due to an inhomogeneity in magnetic field δB_0, after they are turned over, this relative phase becomes $\Delta\phi(t + \Delta t) = +\gamma\delta B\Delta t$. δB_0, though, continue causing a dephasing, that after an further interval Δt results in a total dephasing $\Delta\phi(t + 2\Delta\phi) = +\gamma\delta B\Delta t - \gamma\delta B\Delta t = 0$. This rephasing results in a recreated transverse magnetization vector in the opposite direction of the first one created after the $\pi/2$ impulse.

With this procedure, it is possible to recover the loss of transversal magnetization due to inhomogeneities of the magnetic field. The time interval covering the application of the first impulse $\pi/2$ to the instant at which phases turn back to zero is called **echo time** TE. Between the application of the π pulse and the echo time, the acquisition sampling of k-space starts.

This method is an effective strategy to remove the nuisance effect due to field inhomogeneities associated to T_2', but do not reduce the effect of T_2 due to local fields and spin-spin interactions. The reason behind this is that static field inhomogeneities remain the same even after the π impulse, so they act the same way even after the impulse, while local fields which cause spin-spin interactions change and the rate at which they accumulate phase changes with them. In general no refocus strategy is possible to correct for this effect. Fortunately though, this is not a critical issue for liquids since the time interval over which data are collected is usually smaller than the T_2 time constant.

2.3.2 Gradient Echo sequences

The sequence employed to acquire functional imaging is the Echo Planar Imaging (EPI) which is a particular sequence from the family of gradient echo sequences. Gradient echo sequences differ

from spin echo because they allow a faster acquisition. They use a single excitation impulse with a flip angle less than 90 deg, this allows a minor time interval to make spins back to their longituginal component as they are not completely flipped on the transverse plane.

The transverse component just created decay and dephase according to T_2*. At this point in an spin echo sequence, the π impulse would be applied; in its place, in GE sequences, the gradient along the x axis is reversed. This gradient inversion causes the spins to rephase but what is relevant is that this gradient inversion does not act neither on dephasing due to field inhomogeneities nor on dephasing due to chemical shifts, but just on dephasing due to the gradient field. This is the most important aspect that makes GE different from SE: the relaxation is dictated by T_2* and not just by T_2 In an gradient echo sequence, the echo peak lies on T_2* decay curve while in a spin echo, it lies on T_2 decay curve. Thanks to this property, this image acquisition is susceptible to any chemical variation such as that due to hemoglobin shift.

2.3.3 Echo Planar Imaging (EPI)

EPI refers to a technique to acquire all the 2D k-space after a single rf impulse. This is achieved by applying intermittent small phase encoging gradients called blips: triangular gradients that are switched on and off between gradient echoes in a multi echo acquisition.

To discuss this method is can be useful to give a look at figure 2.2a where all the gradients in play are illustrated, and discuss line by line what they represent

1. A single rf impulse with an angle of 90 degrees is applied to the body under examination
2. Together with the rf impulse, a slice of the body is selected, corresponding to a position along the z axis, by applying the slice selection gradient
3. Now we focus on the single 2D slice to begin the k-space data acquisition starting from the bottom line of the diagram in figure 2.2b: both gradient of phase encoding and frequency encoding are switched on.
4. The frequency encoding (readout) gradient is reversed to create the first echo

5. When the first echo occurs, the first line of k-space (along the kx direction) is acquired. In figure 2.2b this lines corresponds to the horizontal bottom line

6. Now we have to move to higher values of ky: to accomplish so, the phase encoding gradient performs a small blip: it is switched on for a brief time and then switched off. This dephase spins a little further so that we move up along the ky direction in figure 2.2b and the system is ready to acquire the other line of k-space

7. the frequency encoding gradient is reversed again and the second horizontal line along kx is acquired.

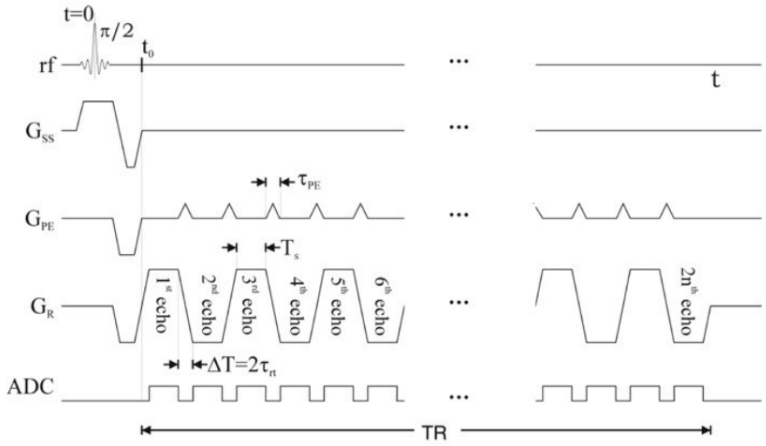
8. This process is iterated: these last two steps are repeated until all the k-space is covered and the slice data is fully acquired.

This acquisition technique allows the acquisition of a single slice in a short time (around 50-100 ms). The whole k-space for a single slice is acquired sequentially, following a snake-like pattern and the number of k-space line acquired after each single rf impulse is named differently according to the scan brand factory: examples are *EPI factor* or *Echo train lenght*.

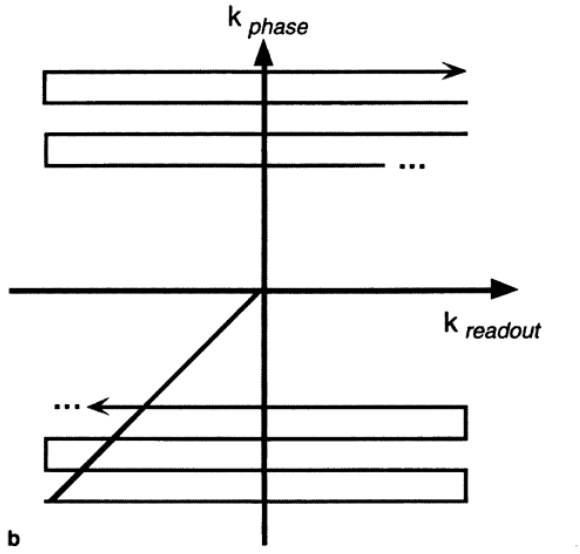
Since slice data are collected after gradient echo inversions, this imaging technique if sensitive to the T_2^* time constant which makes it suitable for functional imaging where we are interested in chemical shifts within a body. For this reason this kind of imaging is often called T_2^*-weighted imaging. In fact, the period of the readout gradient is the echo time TE, and is modulated such that is sensible to T_2^* setting $TE \approx T_2^*$.

2.4 Functional MRI (fMRI)

Functional Magnetic Resonance Imaging is a non-invasive diagnostic tool which aims to measure the neural activity of different parts of the brain. The vast majority of scan acquisition is carried out using EPI sequences (section 2.3.3) and consists of several 3D volumes acquired each 1.5-2 seconds (corresponding to the repetition time TR) for the entire duration of a scan which lasts from 5 to more than 8 minutes, depending on the scanner model.



(a) *Gradients activation sequences for a EPI acquisition where a complete image is acquired after a single RF impulse. G_{SS} is the slide selecting gradient, G_{PE} is the phase encoding gradient, with tiny blips to dephase spins along the y direction, G_R the readout gradient, or frequency encoding gradient, reversed each time for the acquisition of a line along the x axis. Finally, the ADC is switched on to record signals during the echo peack.*



(b) *Acquisition trajectory on the k-space for a single slice. Acquisition starts to negative values of k_x ($k_{readout}$) and k_y (k_{phase}), then, changing the direction of the readout gradient, the acquisition proceeds back and forth, gradually moving towards higher value of k_y at each blip of the phase encoding gradient.*

Figure 2.2: *Gradients of EPI acquisition sequence*

The detected signal comes from the blood oxygenation level fluctuations following a neuronal activation, and for this reason this kind of analysis is referred as BOLD (blood oxygenation level-dependent) imaging.

Blood can be portrayed as a colloidal mixture where blood cells constitute around 40-45 % of volume. [8]

From a physical point of view the variation in oxygen content in blood affects the local magnetic susceptibility of the blood. Signal coming from blood cells is mainly due to the presence (in each blood cell) of several molecules of hemoglobin: a protein containing 4 heme groups each one including an iron atom which binds an oxygen molecule and carries it throughout veins and capillaries. Hemoglobin can be in two states: oxy-hemoglobin and deoxy-hemoglobin. These two molecules differ in the presence of the bounded oxygen molecule, which reflects in differences in their magnetic susceptibility as oxy-hemoglobin has diamagnetic properties while deoxy-hemoglobin is paramagnetic. This difference is due to unpaired iron electrons in the deoxy-hemoglobin which lead to an unshielded molecule against the external magnetic field. The presence of deoxy-hemoglobin causes local field inhomogeneities, leading to a reduction of the main signal coming from water molecules.

This signal weakening is due to spins going out of phase between each other more quickly, because of these additional field inhomogeneities, which results in reducing the total magnetization. This means that nuclei would lose their magnetization faster than the typical T_2^* decay constant. For this reason if the scanner is tuned to the T_2^* relaxation time, it would be possible to appreciate this chemical shift between oxygenated and deoxygenated areas. This gives the alternative name of BOLD-images as T_2^*-weighted images. [26]

This way, since the presence of deoxy-hemoglobin causes a weakening of the signal, when a brain area is activated, it demands a greater amount of oxygen carried by oxy-hemoglobin molecule which results in a signal increase.

The process that, from a neuronal stimulus leads to a measurable blood signal is governed by the hemodynamic response function (HRF) shown in figure 2.3. This figure represent the blood response immediately after a neuronal stimulus (which occurs on a timescale of $\approx$ 10-100 μs). The BOLD signal takes $\approx$ 4-6s to reach a peak and a total of 20-30 s to return to its zero baseline.

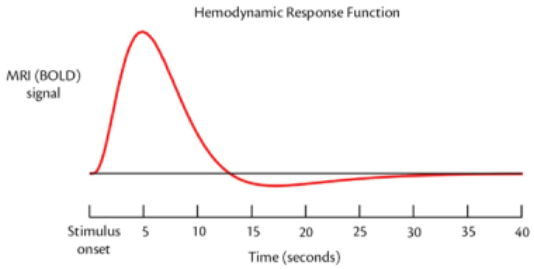


Figure 2.3: *Evolution of BOLD signal of an adult brain during time. At the instant t=0 the neuronal stimulus occurs; after that, it takes around six seconds to reach a maximum of signal, and a total of more than twenty-five seconds to return to its baseline value*

Since we are interested in these slow signal trends, the TR of 1.5-2 seconds is the minimum necessary to have an analysis sensitive to these signals. According to the Shannon-Nyquist sampling theorem, the maximum detectable frequency in a signal is equal to 1/2 of the sampling frequency. For this reason with a TR of 1.5 seconds for example, we are able to detect signals with a period ≥ 3 seconds.

During an MRI and fMRI scan session, data are usually acquired slice by slice, and the thinner is the slice, the more spatial resolution we can accomplish. But there's a trade-off between spatial and temporal resolution: decreasing the slice's thickness would lead to a better space resolution but at the cost of increasing repetition time to mantain the same Signal to Noise ratio (SNR). Thie main reason is that signal is proportional to the number of hydrogen nuclei, which is proportional to the slice volume. In order to obtain a strong BOLD signal, echo time plays a significant role as well: to obtain the maximum strenght signal echo time has to be set $TE \approx T_2^*$ that means TE around 30 ms. Images acquired using a shorter echo time have a weaker BOLD signal because of the lack of signal to detect. [28]

Chapter 3

Machine learning

Machine learning is a branch of Artificial Intelligence (AI) that aims to learn from data and apply this knowledge on new, unseen data. It is possible to make predictions or perform classification of data and, through the training process, gradually improve accuracy on the task a model performs. Regarding the training process of an algorithm, as a rule of thumb, the bigger and homogeneous is the dataset to train it, the better accuracy and generalization can be achieved.

According to the analysis we aim to perform, and the type of dataset we are working on, machine leraning algorithms can be divided into two macro areas: supervised and unsupervised learning.

- From a dataset, consisting on a collection of data, each one containing a certain number of features, an **unsupervised** algorithm goal is to learn the principal properties from the dataset structure and to extract information from data without human labour to annotate and label each data. Analysis with unsupervised algorithms include clustering or dimensionality reduction.

  Clustering is maybe the most simple and intuitive example of an unsupervised learning: it is basically a classification process through which unlabeled data are reorganized and classified into subgroups according to some common properties or distance measures that arise from the analysis. After this process, data of each group, called cluster, share a certain degree of similarity in feature probability distribution. Dimensionality reduction is another important example of unsupervised learning process: it is performed with the task of finding a different

representation of the data, with a lower dimensionality, and preserve as much information as possible from them. This can be accomplished either by compressing data into a lower-dimensional space, or by searching the main source of variance across the data and create a representation in such a way that the dimensions of the new representation are statistically independent.

- On the other hand, when we are dealing with **supervised learning**, we work with a dataset where each data is associated with a label, specifying the class data belong to. Thus, the algorithm is trained to learn the common features for each class and to predict the correct label associated to each input data.

  Generally speaking, given an input data x and an associated label y, a model tries to estimate the probability of obtaining y given x: $p(y|x)$. These algorithms are referred as supervised because of the human labour necessary to label each input data.

A subset of Machine Learning is Deep Learning, the main difference lies on the structure of its algorithms and on the learning techniques. Traditional machine learning methods consist on algorithms like Random Forest, Support Vector Machines (SVM), K-nearest neighbor (KNN) and several more models, while deep learning includes models as Artificial neural networks (ANN), Convolutional Neural Networks (CNN) and more.

3.1 Random forest

Random Forest classifier is a common machine learning algorithm that belongs to the category of *ensemble* classifiers. It combines multiple decision tree models to reduce overfitting of data (see section 3.2) and to create a more powerful model that achieve good generalizability. Therefore, to properly understand a random forest algorithm, we need to start understanding how a Decision Tree classifier works.

Tree classifiers

A decision tree aims to learn distictive traits from input data by asking yes/no questions. Focusing on one single input feature, the classifier splits the dataset based on the value of that feature

according to an if/else statement like "if feature_i > a ". Thus, each branch of a decision tree consists of a question that causes the split of the dataset into two smaller sub datasets. In a dataset containing multiple features, the process is recursively repeated until it reaches an end point called *leave*, corresponding to the ultimate partition, containing data points belonging to a single class.

The goal of a tree is to construct branches so that the partition are informative about the class labels. In a practical way, given a dataset X made of different data samples, each one containing n features, the algorithm construct a tree following these simple steps: starting from the top node called *root*, it searches among the n features the one which allows the best split between classes and split the dataset into two subsets, each one constituting a node. Then, for each node, the process is repeated until a single leaf is left.

The best split is performed according to a pre-defined objective function we want to maximize throughout the construction of a tree. Usually these functions regard measures of the homogeneity of class labels in a node, usually referred as impurity measures. By lowering the impurity of a node, it seeks for the maximization of the information gain, defined as the difference between the parent node impurity and the weighted sum of the child nodes impurities.

A commonly used impurity measure is Giny Impurity. [41]. It can be interpreted as the probebility to misclassify an observation. If a node contains a sub-dataset Q with a total number of data n, belonging to k different classes, the Gini Impurity measure H(Q) is obtained by the simple relation

$$H(Q) = \sum_k p_k(1 - p_k) = 1 - \sum_k p_k^2 \tag{3.1}$$

where p_k is the fraction of data in Q belonging to class k.

Following this principle, the splitting process is iterated and the tree is grown.

The structure of a tree can be visualized in a plot that contains all the informations necessaries to understand the tree, such as the feature according to which each split is made, and the impurity measure of each node. This property is one of those that make decision trees so popular: they are easy to interpret since their structure and the information of each node can be visualized in a clear plot. Other positive sides of decision trees are, their easy to use implementation which requires little data preparation and their rapidity since their complexity goes like $O(n_{features} n^2_{samples} log(n_{samples}))$

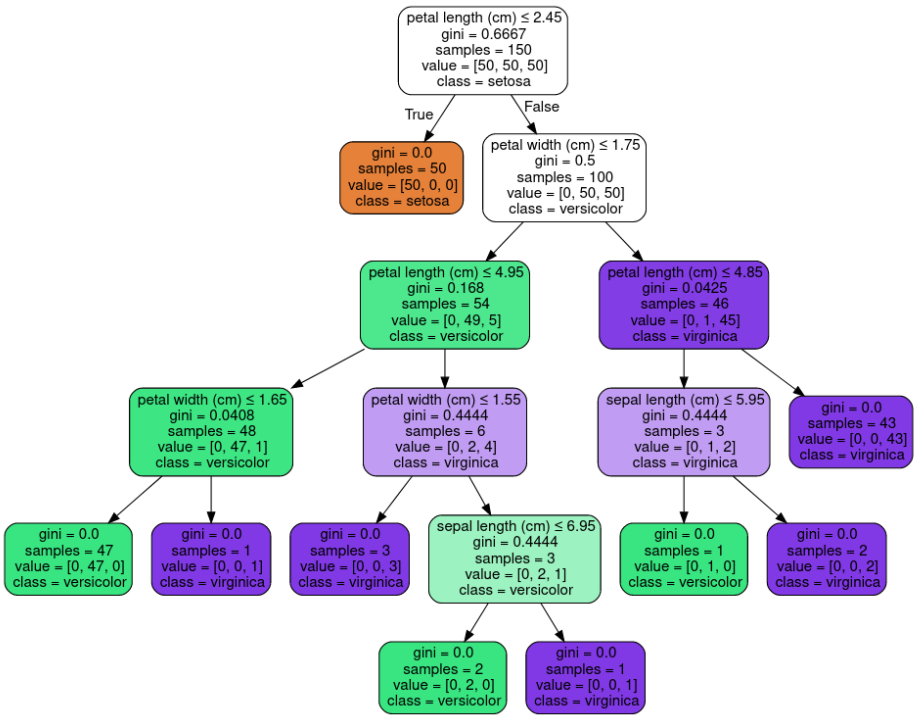


Figure 3.1: *A graphical representation of a decision tree for classification of the Iris dataset consisting on data belonging to three different classes of iris: Setosa, Versicolour, and Virginica. Each node of the tree shows the criterion used for its split, the Gini values, the total number of samples in that node and the main class that data belong to*

[37]

An example of a decision tree dealing with flower classification is shown in figure 3.1. We choose this example because is easier to understand how a decision trees works with this data where each feature is labeled with an intuitive name and contains a physical quantity easily comparable with everyday life such as petal lenght expressed in centimeters. This example is taken from the scikit-learn website [1]

Instead of looking at the whole tree, we may be interested in what feature contributed the most to the final output. To this end, a decision tree usually weights feature with a number from 0 to 1 according to how much a feature contributed to the impurity decrease along the tree. The importance of a feature is calculated as the decrease in node impurity weighted by the probability

[1] https://scikit-learn.org/stable/modules/tree.html#

of reaching that node, where this probability is just the number of samples that reach that node divided by the total number of samples. [41] [33]

Unfortunately though, one of the main drowbacks of decision trees is that the iterating process towards the reaching of a leaf, can bring to the creation of a over-complex model that overfits the training data. That is, the model accurately learns the training dataset but is not able to generalize well to a test dataset. To prevent overfitting, one possible strategy is to early stop the iteration towards the leaf, and leave a node with more than a single sample left. An other strategy, which leads to the construction of a Random Forest, is the creation of different trees and put information together, in order to obtain stronger model, more prone to generalize information and reducing overfitting.

Random Forest

A Random Forest is a collection of decision trees where each tree is different from the others, each tree tends to overfit, but following different patterns. Taking advantage of this process, we could reduce overfitting by averaging different results. Random forest gets its name because of the insertion of randomness during the construction of different trees. Given an input dataset X with N different samples, there are two main steps where randomicity plays an important role: when selecting the number of samples to grow each tree on, and when selecting the number of features to consider when looking for the best split of a node.

In a random forest, one of the main parameters is the number of tree to grow. Once selected this number, the process can start where each tree is constructed according to the following steps:

1. Randomly select $n \leq N$ samples from the input dataset X, this operation is called bootstrap.
2. Grow a decision tree from this bootstrap sample. For each node, the algorithm randomly limit the number of feature available and compute the best split on this remaining subset of features. This avoids correlations between trees and results in better performances.
3. Repeat steps 1) and 2) as many times as the number of trees to build.
4. Once all the trees are created, the forest is ready to make predictions. Each tree is used to make a prediction and all the results are collected. The final prediction of the forest is

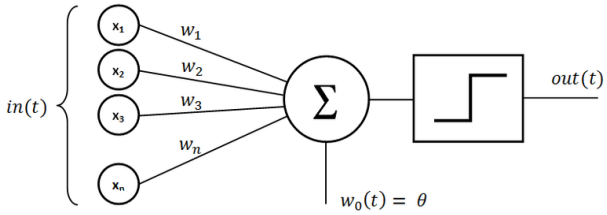


Figure 3.2: *A schematic representation of a perceptron: this perceptron receives an input vector t, with n features $x_1, ...x_n$, each one weighted with a different weight $w_1, ...w_n$ and a offset w_0, and compute a linear combination of them and returns an output, activated by a step function: it returns 0 if the linear combination's value doesn't reach a certain threshold, or returns the value itself if it does.*

assessed by majoring vote: the predicted class will then be the one predicted by the majority of classifiers.

Just as we discussed for tree classifiers, it is possible to extract information about features for a random forest as well. Important features for a random forest are assessed by extracting important features from each tree and by averaging these results.

3.2 Deep Learning: ANN

In the previous paragraph we discussed one of the most important machine learning algorithm, but, as mentioned before, there is a subset of machine learning algorithm with a common typical structure, thanks to which they get their name of deep neural networks. The structure of algorithms belonging to this family are inspired by a brain neuronal structure, and even if in a simplified way, they try to emulate the learning process of a brain. These algorithms own the adjective "deep "to their structure: they are organised in layer, each one containing several fundamental unit called neuron. Neurons between layers are connected to each other like synapsis transmit a signal between neurons in a biological brain. Thanks to this structure which reminds a neuronal brain structure,they are called artificial neuronal networks

The foundamental unit which constiture a neuronal network is an artificial neuron, one of the

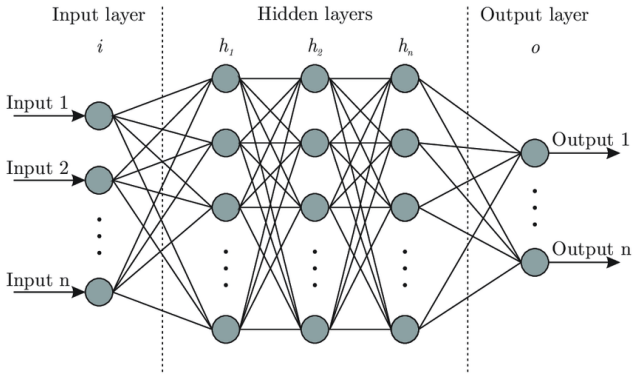


Figure 3.3: *Schematic representation of an artificial neural network containing an input layer, three hidden layers and an output layer: a single vector contained n features (input 1, ... input n) is given as input to the first Input layer i. All the inputs are linked to every neurons of the first hidden layer, which would compute a linear combination of its inputs. All the hidden layers are linked to the other hidden layers, and at the end, n different outputs are returned.*

most relevant example of artificial neurons is the *perceptron* shown in figure 3.2.

A perceptron is the foundamental unit of a supervised deep learning algorithm: it takes as input a data, for example an n-dimensional array $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ and returns an output: a real number computed by applying a function to a linear combination of all the inputs $y = f(z) = f\left(\sum_i w_i x_i + b\right)$. The function that determines the output is called *activation function*, and can be either a simple step function, or a more complex function. We will briefly discuss some of the most important activation functions in section 3.3.

A deep neural network is a hierarchical organization of neurons into layers, connected to each other. Input data are passed to the first input layer where each neuron acting like a perceptron, produces an output using the activation function. All the neurons in a layer have the same activation function, but it can differ from the activation function of other layers. Once collected every output from the first layer neurons, they are passed to the second layer becoming the input to each neuron belonging to this layer. This is reiterated through all the layers up to the final output layer. As schematized in figure 3.3 an artificial neural network is mainly composed of three parts: an input layer, some middle layers also called *hidden layers* and a final output layer. Each neuron of a layer is linked to all the neurons of the previous and next layer and each connection is weighted. At

the beginning, weights are randomly set, but during training they are updated in order to improve network performances.

Using matrix formalism the entire input-output process can be written as

$$\mathbf{y} = \sum_{j}^{n} w_{ij} x_j + b_i \tag{3.2}$$

where $\mathbf{x} \in \Re^n$ $\mathbf{y} \in \Re^m$ being m the output dimension determined by the number of neurons in the output layer. $\mathbf{y}$ represents the output, or prediction of the model. The process through which given an input data $\mathbf{x}$ we obtain an output $\mathbf{y}$ is called *forward propagation*.

In order to train a model, the prediction is compared with the actual value of the label associated to the input data point, and during a process called *backpropagation* the algorithm modifies its weights in order to minimize the difference between the actual and the predicted value.

The goal of a machine/deep learning algorithm is to learn from data using a train dataset, and generalize information in order to perform well on an unseen dataset called test dataset. Performing well means to produce a low error on the test dataset after minimizing the error on train dataset. To fully define a neural network we just need some parameters called hyperparameters, the principals of which are:

- Number of layers
- Number of neuron for each layer
- Activation function for each layer
- Number of epochs (or iterations)
- Learning rate

Number of layers and numbers of neurons are connected to one of the most important concept in machine learning that characterize a network: the concept of *capacity*. Capacity qualitatively refers to the level of complexity that a model is able to learn. This is strictly linked to two critical issues in machine learning: the concept of underfitting and overfitting.

- Underfitting occurs when the model is not able to learn the required amount of information during training. This usually happens for shallow network, when the model has a small number of parameters in regard to the number needed to explain the input features. To have low parameters results in poor performances because the model is not able to learn the underlying structure of a complex data set.

- On the contrary, overfitting occurs when the model has too many parameters compared to those required to learn the input features. What happens then is that the model memorizes all the data it sees during train but it is not able to generalize informations. As a result, the model performs well on the train dataset and has low performances in the unseen test dataset.

To understand this concept is can be helpful to provide an example in a two-dimensional space. If we have to fit a dataset consisting of some points sampled from a quadratic curve, we can choose different functions, for example a linear function, a quadratic function and a polynomial function with grade equal to the number of data points. As shown in figure 3.4 the linear model is not able to describe the data distribution while the polynomial function has too many parameters, and it is able to fit the train distribution, but it does not suit to describe data points belonging to the test, sampled from the same quadratic distribution. Furthermore, if the number of parameters (the grade of the polynomial in this case) is greater than the number of data points, we obtain that an infinite number of different curves are suitable to fit our data, and finding the one which performs well even on the test dataset is an hard task. We should therefore pay attention when choosing the number of parameters on a model to avoid under- or more likely over- fitting.

To monitor the evolution of the training process of a model it is a common practice to split the train dataset into two subsets: one that will actual constitute the train dataset, and a smaller one, called validation dataset used to make constant checkups on how the model is learning with training data. The validation dataset is used for the fine tuning of hyperparameters and it typically consists of 10-30% of the train dataset. Just in this paragraph to distinguish between the two train dataset we will denote as “Train ”, the whole train dataset and as “train ”the subset of the Train dataset left after its split into a train and a validation set. The whole dataset will then be divided into approximately 70% Train and 30% test, and the Train dataset itself will be divided into 70% actual

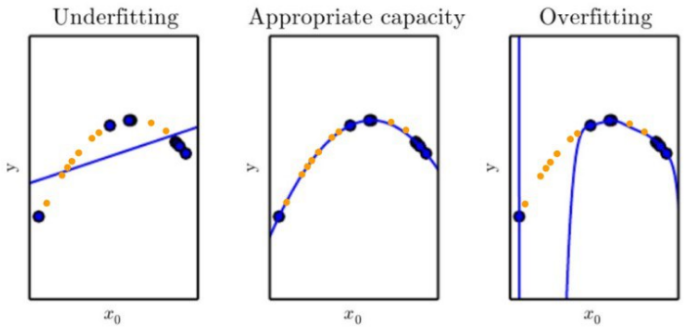


Figure 3.4: *Underfitting, overfitting and appropriate fit in a 2D dataset: three models with different capacities are trained on blue data, if the model is too simple, in this case if the fit function has a few free parameters, it is not able to properly fit data, on the other hand, if it has too much parameters in respect to how much we would need to fit our data, it perfectly match train data distribution, passing through each data point, but performs very poorly on the test dataset (orange points). With an appropriate number of parameters, (middle figure) the model is able to fit train data, and generalizes well to test data.*

train and 30% validation. When the best combination of hyperparameters is found, it is possible to actually train the model using the entire Train dataset.

Regularization strategies

Some regularization procedures are often implemented to avoid overfitting. The strategy is to build a model with a capacity slightly higher than the necessary in order to perform well on the training dataset, and then, to avoid overfitting, implement some regularization techniques to achieve good generalization performances.

Some of the most popular are Dropout and Batch-Normalization

- Dropout is a regularization strategy that inserts randomness during the training of a model. It is usually applied to the neurons of hidden layers and it randomly drop a certain fraction of hidden neurons and their connections, during each training cycle. The percentage of neurons to drop, corresponding to the probability for a single neuron to be dropped, is set by the user. A visual representation of what occurs is illustrated in figure 3.5. When discarding

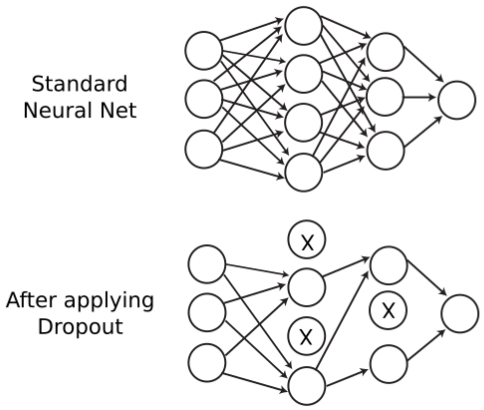


Figure 3.5: *Schematic representation of a 0.5 dropout procedure: in a standard neural network, each neuron is linked to all the others, but if we implement a 0.5 dropout, for each iteration, neurons on hidden layers have a 50% probability to be dropped and excluded from the computation of outputs*

some neurons in a layer, the remaining neurons needs to rescale their weights to account the missing connection; doing so, every neuron cannot rely on the the input of all the preceding neurons and the network is forced to learn more robust patterns from the data. With this strategy we achieve the same results as we would obtain by training different models with different structures and average all the outputs.

- BatchNormalizzation is a regularization scheme that has been commonly adopted since its introduction in 2015 [**?**]. It is based on the observation that a neural network works and performs better when its input are normalized because this prevents the saturation of its neurons. A neuron can in fact saturate and settle to a certain value because of an high input, this causes the neuron outputs value to be always close to the asymptotic end of its activation function (see section 3.3), resulting in a biased and less accurate prediction. What is essentially done is then a simple scaling of each neuron input: for a layer l with d neurons its input $\mathbf{x} = \{x_1^l, x_2^l, ..., x_d^l\}$ is normalized by removing the mean value across all the input data, and by dividing for their variance $x_i^l \rightarrow \tilde{x}_i^l = \frac{x_i^l - \mathbb{E}[x_i^l]}{\sqrt{Var(x_i^l)}}$

Cross validation procedures Da passare nella parte di Implementation and results?? When dealing with small datasets, dividing them into train and test can be a non trivial issue because of the shortage of data for a proper train procedure.

To work this out, a procedure called k-fold cross validation can be implemented, at the cost of increasing computational costs. It consists on the creation of k different partitions of the main whole dataset (before splitting it into train + test). From these partitions, k-1 are used as train, and the last is used as test. Every subset of partitions is used so that each different partition is used as test, and the others as train. Thus, for each k-th run there are two subsets (a train and a test) of the original dataset. Following this procedure, at each iteration, the train dataset is different, and the model is tested on a different dataset each time. An image showing this procedure is 3.6. In practice what is usually done, is not a sequential partitioning the dataset as shown in figure 3.6 but a shuffled partition of data. This avoids creating folder containing all the same label in case the dataset was ordered, but randomly picking data in order to create subsets containing the same proportion between classes as in the main dataset.

Overall, we can imagine this process like the creation of k different models, each one trained on a different partition (k-1 folds) and tested on the remaining k-th fold. We evaluate each of these models, and we take as a result the average score across all the outputs.

3.3 Activation functions

The activation function of a neuron, and consequently of a layer defines the output of each neuron belonging to that layer. There are different activation functions, linear or non-linear. A linear activation function outputs a value $f(z) = a \cdot z + b$ where we denote as z the scalar product between an input vector $\mathbf{x}$ and the weights vector $\mathbf{w}$ plus an eventual offset w_0, and as $\mathbf{z}$ the matrix-vector product between the matrix of weights $\mathbf{w}$ and the input vector $\mathbf{x}$ and an offset vector $\mathbf{w_0}$.

In practice, however, it would not be very useful to introduce a linear activation function since the combination of linear functions is a linear function itself. In fact, in a neural model, we seek to introduce non-linearity in order to deal with more complex tasks. There are several non-linear functions that can be employed depending on what data we are working on, or on what kind of

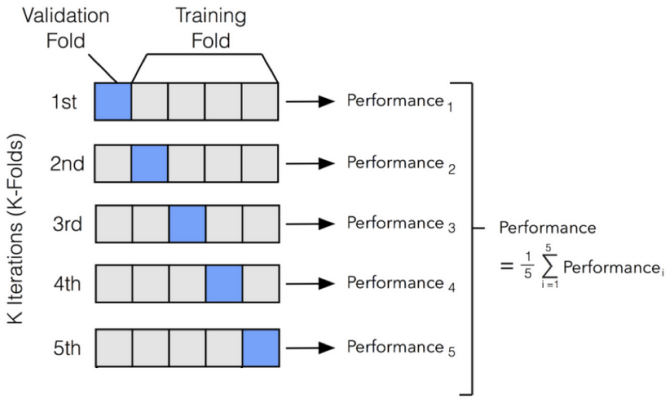


Figure 3.6: *Schematic representation of a k-fold cross validation procedure with $k = 5$: Dataset is partitioned into 5 subsets, four of which are used for training and the remaining one for the testing dataset. during each iteration a model is trained over 4 different folds and tested on the remaining one, until each fold has been used at least once as part of training and once as test.*

classification task we are performing. Some of the most popular are:

- The ReLU function shown in figure 3.7a is defined as

$$\phi(z) = max(0, z) = \begin{cases} 0 \ for \ z < 0 \\ \\ z \ for \ z > 0 \end{cases} \tag{3.3}$$

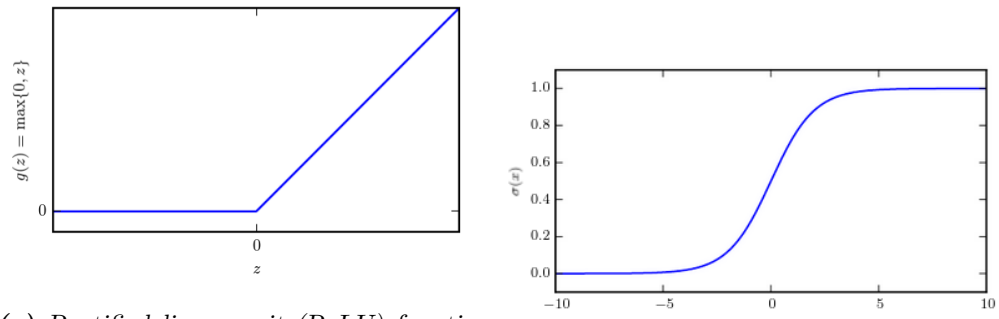
  and returns the maximum value between the input and zero, it essentially puts to zero all negative inputs and leaves the positives untouched.

  - A modified version of the ReLU function called Leaky ReLU was introduced defined as $\phi_{leaky}(z) = \begin{cases} \alpha z \ for \ z < 0 \\ \\ z \ for \ z > 0 \end{cases}$ where α is a coefficient usually < 1, typically of the order of 10^{-2}

- The sigmoid function, or logistic function show in figure 3.7b is defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.4}$$

  and outputs a real number between 0 and 1. For this reason is a common choice when we have to model a probability. For example in a binary classification task, when it is employed



(a) *Rectified linear unit (ReLU) function: it returns zero if its argument is negative and returns the argument itself if it is positive, according to equation 3.3.*

(b) *Sigmoid function: returns a value between 0 and 1, according to equation 3.4*

Figure 3.7

as activation function of the last layer, it can be interpreted as the probability of the input data to belong to the class 0 or 1.

- The softmax function, is a generalization of the logistic function, and is commonly used for multiclass classification. It is defined as

$$\phi(z) = p(y = i|z) = \frac{e_i^z}{\sum_{j=1}^{M} e^{z_j}} \tag{3.5}$$

It is applied to the vector $\mathbf{z}$ and describes the probability of $\mathbf{x}$ to belong to the class i over a total of M classes. With this function, classes are regarded as mutually exclusives, so if the probability to belong to class i is p, the probability to belong to one of the other classes is $1 - p$. To hold this property, softmax can't be applied independently to each input z_i, since it depends on all elements of $\mathbf{z} = \{z_1, ..., z_M\}$

3.4 Loss functions

In order to train a network and improve its performances we compare the predicted output value with the value of the label associated to a data, and compute an error, or distance between these two values.

This error is computed by using a *loss function*. In common classification tasks, the goal of a

network is to gradually reduce the error, looking for a minimum of these functions, according to a process called backpropagation, that we discuss in section 3.5

The most common loss function for classification problems is the cross-entropy loss. It relies on the concept of cross-entropy between two distributions $\hat{y}$ *and* y defined as

$$H(y, \hat{y}) = -\sum_{m=0}^{M-1} y_i \cdot log(\hat{y}_i) \tag{3.6}$$

Where the sum is intended over all the M possible values a variable y can assume (all the M possible classes in a classification problem).

If classification only concerns two classes is called a binary classification, and its associated label usually have values $y = \{0, 1\}$. If we explicit the sum of equation 3.6 for M = 2, cross entropy for a binary classification, for one observation can be calculated as

$$\ell_i = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)) \tag{3.7}$$

If we have N data belonging to two classes, we can write the *binary cross-entropy* loss as

$$L = \frac{1}{N} \sum_{i}^{N} \ell_i = -\frac{1}{N} \sum_{i=1}^{N} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i) \tag{3.8}$$

Binary crossentropy loss can be generalized to the case of multi-class classification. In this case, if data belong to M classes, labels y can assume values $y \in \{0, 1, ..., M - 1\}$. To define a loss, each label y_i must be one-hot encoded (see section 13) in order to create a binary vector of dimension M, with all but one entry equal to zero, and the position of the only entry equals to 1 specifies the class. For example if a datapoint belongs to class m, its corresponding one-hot vector will be defined as $y_{im} = \begin{cases} 1 \text{ if } y_i = m \\ 0 \text{ otherwise} \end{cases}$ In this case the loss function is called *categorical cross-entropy*

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{m=0}^{M-1} y_{im} log(\hat{y}_{im}) + (1 - y_{im}) log(1 - \hat{y}_{im})) \tag{3.9}$$

In equations 3.8 and 3.9 $\hat{y}$ are probability of the input data $i - th$ to belong to a class. For binary classification, this probability is usually modeled with a sigmoid function, while for multi-class classification, a softmax function is usually employed.

Relation of the loss function with a Maximum Likelihood Extimation

The estimation of the minimum of the loss function can be seen as a process of Maximum Likelihood Extimation. Given a probabilistic model depending on m different parameters $\theta_1 ... \theta_m$, the likelihood is a function that describes the probability of observing a value $\hat{y}_i$ as a function of set of parameters $\{\theta_i\}_1^m$. If we set values of these parameters we obtain the probability of observing the value $\hat{y}_i$ under the given model [3].

$$\mathcal{L} = (\theta_1, ..., \theta_m | \hat{y}_i) = P(\hat{y}_i | \theta_1, ... \theta_m). \tag{3.10}$$

For a subset of observed value we can write the likelihood as $\mathcal{L} = P(\hat{y}_1, ..., \hat{y}_N | \theta_1, ... \theta_m)$. If all the observed values are independent, we can write the total probability as the product of single probabilities of observing each sample $\hat{y}_i$ given a model with parameters $\{\theta_i\}_1^m$:

$$\mathcal{L} = \prod_{i=1}^{n} p(\hat{y}_i | \theta_1, ... \theta_m). \tag{3.11}$$

We consider the problem of a binary classification using a deep neural network. With this model, parameters $\{\theta_i\}_1^m$ correspond to the inner weights of the networks $\mathbf{w} = \{w_i\}_1^m$. Input data are a set of N datapoints $\{x_1 ... x_N\}$ each one associate with a label $\{y_1 ... y_N\}$ which can be either 0 or 1. Overall, the whole dataset can be denoted as D = $\{(\mathbf{x_i}, y_i)\}$.

Given an input data $\mathbf{x_i}$ and an activation function (sigmoid, for example, since we are performing binary classification) we model the probability of $\mathbf{x_i}$ to belong to the class $y_i = 1$ as:

$$P(y_i = 1 | x_i, \mathbf{w}) = \sigma(z_i) = \frac{1}{1 + e^{-z_i}} \tag{3.12}$$

where z is defined as in section 3.3 $\mathbf{z_i} = \mathbf{x_i w}$ And since we are dealing with a binary classification, where y_i can be either 0 or 1, the probability to belong to one class is 1 minus the probability to belong to the other:

$$P(y_i = 0) = 1 - P(y_i = 1) \tag{3.13}$$

Using this result for a set of data $D = \{(\mathbf{x_i}, y_i)\}_1^N$ and substituting the formula for a single probability in the likelihood function 3.11 we obtain the likelihood of observing the dataset D under

a model with parameters $\mathbf{w}$

$$\mathcal{L} = P(D|\mathbf{w}) = \prod_{i=1}^{N} \sigma(\mathbf{x_i w})^{y_i}(1 - \sigma(\mathbf{x_i w}))^{1-y_i} \tag{3.14}$$

To perform a Maximum Likelihood Estimation we look for a maximum in this function by computing partial derivatives. However, since computing the derivative of a product is a nontrivial task, we can consider the logarithm of $\mathcal{L}$ and compute its maximum since the logarithm is a monotonic function and compute its maximum corresponds to computing the maximum of the function itself. The logarithm of equation 3.14 is

$$log(\mathcal{L}) = \sum_{i=1}^{N} y_i log(\sigma(x_i \mathbf{w})) + (1 - y_i) log(1 - \sigma(x_i \mathbf{w})) \tag{3.15}$$

which taken with a negative sign, and averaged over all the N data samples corresponds to the cross-entropy function in equation 3.8.

Thus, minimizing the binary-crossentropy loss is the equivalent of a maximum likelihood estimation for the parameters of our model.

3.5 Gradient descent and Backpropagation

During the training of a model, after the calculation of the loss function, the network moves to the estimation of the best parameters through an algorithm called *backpropagation*. To perform the minimization of a loss function, even if it would be theoretically possible to find a minimum by means of an analytical way, in practice, usually the number of weights is so huge that numerical methods must be employed. The most popular method to compute gradients and optimize parameters is the **gradient descent**.

We denote a generic loss function as $L(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, \hat{y}_i(\mathbf{w}))$ where $\mathbf{w}$ is a vector of weights, the minimum of this function corresponding to the vector $\mathbf{w}^0$ can be found by following the subsequent steps:

1. Choose a random initial guess for $\mathbf{w}^0$ and start iterating

2. At iteration $i+1$ we reach a weights vector $\mathbf{w}^{i+1}$ given by the formula

$$\mathbf{w}^{i+1} = \mathbf{w}^{i} - \eta \nabla_{\mathbf{w}} L(\mathbf{w}^{i}) \tag{3.16}$$

where the $\nabla_{\mathbf{w}}$ indicate the gradient of the cost function with respect to $\mathbf{w}$ components, and η is a parameters called *learning rate*. It specifies the dimension of each step during the descent toward the minimum of the cost function.

A drawback of this gradient descent algorithm is that it works by computing all the gradients for all the data points before updating the weights. So the weight is updated only after the whole dataset is been seen, resulting in a huge computational cost.

An optimized version of this algorithm is called **Stochastic Gradient Descent** (SDG). It is often used because it brings some advantages such as decreasing computational cost and, introducing stochasticity. The insertion of randomicity during the gradient computation results in a reduced chance for the algorithm to get stuck in local minima.

SGD works by approximating the gradient of the cost function calculated with all the input data, with a gradient computed using only a small subset of input data called *mini-batch*.

With a dataset comprising N input data, we can create subsets containing m elements and obtain N/m minibatches. The gradient is computed on a mini-batch and weights are updated. This process is repeated and when the gradient is computed over all the mini-batches it is said that the training proces completed an *epoch*. The number of epochs is one of the hyperparameters to set when choosing a training strategy of a model.

Even though SGD performs quite well, it can be further improved by introducing the concept of momentum. Momentum is represented by a parameter $0 \leq \gamma \leq 1$ and it takes track of the descending direction by running an average over all the preceding encountered gradients. This process helps the algorithm speeding up the descending process if in a certain direction the gradient is constant and it does not change slope.

The descending process can be further improved by taking into account even the steepness all over the dimensions. To accomplish so the algorithm has to be improved by adding the calculation of second order momenta also called *uncentered variance*. This procedure, would ideally bring to the

calculus of the hessian matrix but at the cost of increasing computational costs. Recently introduced algorithm can accomplish this task by approximating the calculus of the second momenta.

With this improvement, the algorithm keeps track of the curvature of the loss function in the space of parameters, and takes big leaps in steepest direction and small steps in flatter ones, allowing us to adaptively change the descending step size.

One of the most popular among these algorithms is **Adam**: it accomplish this computation by making use of two different optimization algorithms: AdaGrad and RMSProp. It is a powerful algorithms since it adapts the learning rate taking into account both the first and the second momenta. This leads it to perform better and quicker in finding the minima than simpler SGD with momentum algorithm.

The process of computing gradients is an important step of the backpropagation algorithm, which is the process through which the weights of a netwrork are updated. A neural network with L layers, given an input data, produces an output through the feedforward propagation, and with this value, the loss is calculated. Denoting with z_j^l the weighted input to the $j-th$ neuron of the $l-th$ layer, the first step of backpropagation is to compute the error

$$\Delta_j^L = \partial L / \partial z_j^L$$

On the last layer. Then, using this quantity it is possible to calculate Δ_j^l for all the layers by exploiting the chain rule of derivatives

$$\Delta_j^l = \sum_k \frac{\partial L}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

Once computed all the errors, it is possible, for each layer, to compute the loss with respect to the weights of the networks and modify them according to equation 3.16.

3.6 Dimensionality reduction: PCA

Principal Components Analysis is a method to perform dimensionality reduction. It is an unsupervised learning algorithm which aims to reduce the dimensionality of input data, preserving as much information as possible from it. It does so by learning a new representation of data with lower

dimensionality than the initial one and whose element have no linear correlation with each other. It performs then a projection of data on to this new space, created such that its axes lie on the directions along which data variance is the biggest.

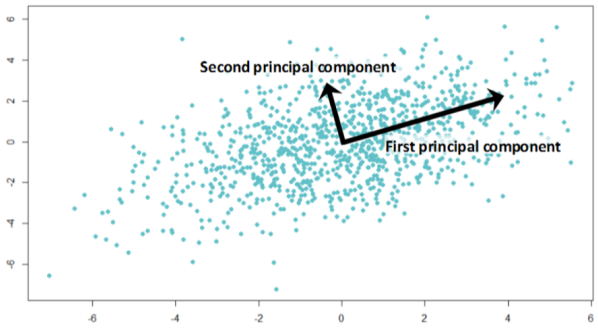


Figure 3.8: *Example of the first two principal components in a 2D dataset: the first principal component is placed along the direction where variance is greater, and the second component is in the orthogonal direction.*

PCA seeks to find an orthonormal basis so that the first base vector corresponds to the direction along which the variance of data is the greatest. This vector is called first principal component (PC). The second principal component is defined as the vector orthogonal to the first, that explains the most variance left once the first component is removed. And so on, the $i-th$ PC is the direction orthogonal to the first $(i-1)-th$ vectors that maximize the left variance. In figure 3.8 is reported a graphic example of two PC extracted from a set of two-dimensional data.

Principal components are calculated as the eigenvectors of the covariance matrix, and the most popular way to implement this calculation is through Singular Value Decomposition (SVD) of the matrix of input data. SVD is preferred over the simpler calculation of the covariance matrix and its eigendecomposition because there are algorithms that can deal more quickly with SVD and avoid the explicit calculation of the covariance matrix.

Concretely, if we consider a set of n input data vectors lying in a space $\Re^m$, we can represent them as a matrix $X \in \Re^{n \times m}$ where n is the total number of input data and m is the dimensionality of each data (the number of features in each data vector).

We can suppose without loss of generality that feature distributions across data have zero mean,

so that for each feature f_i with i = 1, ..., m, $\mu_i = \mathbb{E}[f_i] = 0$.

The covariance matrix $\Sigma \in \Re^{m \times m}$ of input data X is given by

$$\Sigma = \frac{1}{n-1} X^T \cdot X \tag{3.17}$$

because each entry can be written as $\sigma_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} (x_j^{(i)} - \mu_j)(x_k^{(i)} - \mu_k)$, and, in the hypotesis of zero mean $\mu_j = \mu_k = 0$, we obtain $\sigma_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} (x_j^{(i)} x_k^{(i)})$.
Since the covariance of a variable with itself is its variance (Cov(a,a)=Var(a)), in the main diagonal of Σ we actually have the variances of each feature. And since the covariance is commutative (Cov(a,b)=Cov(b,a)), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal.

We are interested in finding a new basis such that covariance matrix is diagonal in this basis and then perform a rotation of data using this basis. To find it, PCA makes use of Singular Values Decomposition of X.

SVD is basically a factorization of a n x m matrix X that (in the case X is a real matrix), allows to rewrite it as $X = USW^T$ where U and V are respectively m x m and n x n orthogonal matrices whose columns are called *left* and *right singular vectors* of X, and S is a m x n rectangular diagonal matrix. Diagonal values of S: s_i are uniquely determined by X and are called *singular values* of X.

Using this decomposition it is possible to write

$$\begin{aligned} X^T X &= (USW^T)^T (USW^T) \\ &= WS^T U^T USW^T \\ &= WS^2 W^T \end{aligned} \tag{3.18}$$

where we used the definition of orthogonal matrix for U $U^T U = \mathrm{I}$.

We therefore rewrite the covariance matrix

$$\Sigma = \frac{1}{n-1} WS^2 W^T. \tag{3.19}$$

which means that the singular values of X: s_i are related to eigenvalues of the covariance matrix by the relation $\lambda_i = s_i^2/(n-1)$, while the right singular vectors of X represents the eigenvectors of the covariance matrix.

Using this result, we consider the transformation of data given by Z = XW, to show that with these rotated data, the covariance matrix is diagonal.

$$\begin{aligned} Var[Z] &= \frac{1}{n-1} Z^T Z \\ &= \frac{1}{n-1} W^T X^T X W \\ &= \frac{1}{n-1} W^T W S^2 W^T W \\ &= \frac{1}{n-1} S^2 \end{aligned} \tag{3.20}$$

This shows that if we use right-singular vector of X to perform the transformation, the covariance matrix of the transformed data is in a diagonal form.

To actual reduce the dimensionality of our data, we need to extract the first $\tilde{m} \leq m$ eigenvalues from the covariance matrix, order them in a descending order of magnitude, and collect the corresponding eigenvectors from the matrix W.

With them we can construct the projection matrix $\tilde{W}_{\tilde{m}} \in \Re^{m \times \tilde{m}}$ of the first $\tilde{m}$ eigenvectors, and use it to project data in order to obtain a new dataset $Y = X\tilde{W}_{\tilde{m}}$ made of n new data of dimensionality $\tilde{m}$.

An important parameters to quantify the amount of variance that PCA is able to explain, is the *variance explained ratio*, given by the ratio of an eigenvalue of the covariance matrix, and the sum of all the eigenvalues.

$$\frac{\lambda_i}{\sum_{k=0}^{m} \lambda_k} \tag{3.21}$$

3.7 How to assess network performances

After the model has been trained, it is evaluated on the test set to assess its performances on a new dataset. A comomon metric for evaluating model performances is the *accuracy*: it is simply defined as the ratio between the total number of correct prediction and the total number of predictions (total number of data in the test dataset). For a binary classification, indicating as T and F, true and false, and P and N, positive and negative, it is possible to give some important definitions:

- TP: data classified as P, belonging to class P
- TN: data classified as N, belonging to class N
- FP: data classified as P, actually belonging to class N
- FN: data classified as N, actually belonging to class P

Using these definitions, accuracy is defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.22}$$

Even though accuracy may seem a good parameter, it is not the most accurate one, when we're dealing with a unbalanced test dataset. As an example, in a binary classification concerning classes A and B, we can be in a situation where a model is not able to make distinctions between the two classes, and classifies all the data as belonging to A. If we test this model using a dataset consisting on 10 data, with just 2 samples belonging to B and the rest to A, the model predicts every data belonging to the class A, yet we would get a score of 80 % accuracy, which is not a thruthful result.

For this reason there are other ways to evaluate performances of a network that overcome this problem. One of this is based on the introduction of two quantities: precision and recall. Precision is the ratio between true positive and total positive classified cases (TP + FP), which is a measure of how many positive predicted cases are actually positive. Recall measures the percentage of actual positives that were correctly classified, or in other words it represents the true positive rate.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \tag{3.23}$$

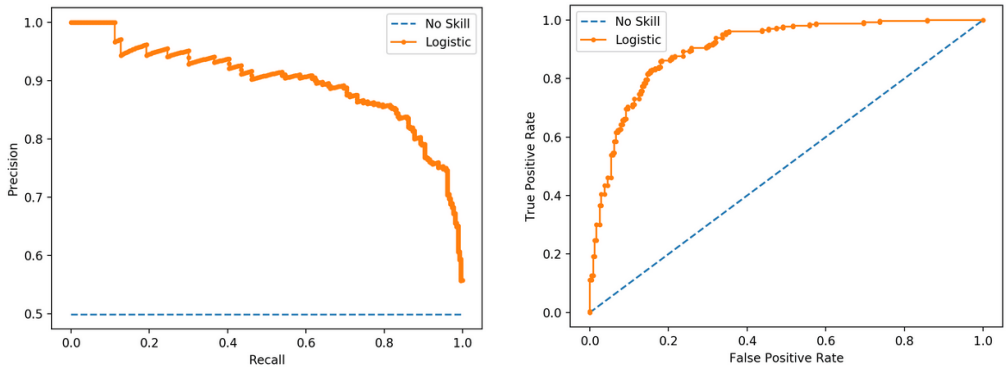
Unfortunately precision and recall are linked in a sense that often enhancing one leads to the decrease of the other and vice-versa. It can be set a trade-off between recall and precision according to what quantity is more crucial for the classification task we are performing. For example we could be willing to take the risk to have more false positive, in order to achieve a greater number of true positive.

To reach a threshold of positive missing $< x\%$ means set the recall to (100-x)%, this operation of establish a threshold is usually referred as "setting the operating point". This threshold though

is not always set at the beginning, since the best operating point is not always clear a priori. For this reason, what is done is to study the model under all the possible thresholds, and plot results creating a curve called precision-recall curve, an example of which is reported in figure 3.9a. The closer a curve lies on the upper right corner (high precision and high recall), the more correctly the model is working. One way to summarize the information of a precision-recall curve can provide us, is to compute the area under the curve, known as "average precition "

Similarly to the precision-recall curve, a other curve is usually employed to study the effect of different thresholds. It is called the *Receiver Operating Characteristics curve*, usually referred as ROC curve. The ROC curve is constructed using the true positive rate (recall) and the false positive rate (FPR) and shows the evolution of TPR vs FPR for diefferent thresholds. The ideal curve would be close to the top left (high TPR and low FPR) and the less accurate is the model, the more this curve tend to lay down to the bisector line. To summarize the model's performances with a single number using ROC curve information, we compute the Area Under the Curve AUC. The reference value for an AUC is 0.5 which is obtained when a model is just randomly predicting and it corresponds to a curve laying on the bisector line.

The AUC can be regarded as the probability that a randomly picked point from the positive class, will have an higher score (according to the model) than a randomly picked point from the negative class. In other words, the percentage of the AUC value is an estimate of the probability that the model is able to distinguish between the two classes. An example of ROC curve is shown in figure 3.9b



(a) *Precision-recall curve representing values of precision and recall for different thresholds values*

(b) *ROC curve representing values of True positive and False positive rates for different thresholds values*

Figure 3.9

Chapter 4

Explainable AI: a game theory approach

As methods to learn pattern from data becomes more complex, they become harder to interpret. Deep learning represents an example of a technique to search for nonlinear relations between data, but introducing nonlinearity, makes results difficult to be explained. Because of this, it is not uncommon to consider a machine learning model as black box where results are collected without any knowledge of the inner processes that produced them. However, in order to obtain more reliable results, it is a crucial issue to get rid of the black box idea and provide an explanation of the main features that characterized the output of a model One state-of-art explainatory algorithm is called SHAP, and it is built making use of an important result from game theory. For this reason, in the following section, we briefly discuss some important concepts related to this field, that were subsequently readapted, for the implementation of this important machine learning explainatory model.

4.1 Shapley values

Game theory is a branch of mathematics related to the study of mathematical models to conceive social situations among competitive players [46]

It had a great development in the XX century especially during and after the Second World

War thanks to the contribution of mathematicians like John Von Neumann, John Nash and Lloyd Shapley. Game theory mainly focuses on two major research areas: non-coperative and cooperative games.

- Non-cooperative games concern competition between individual players who don't cooperate each other and can't form coalitions. The main task on non-cooperative games can be summarised as the search for a good strategy for each player. Each player's objective is in fact the maximization of his own utility function. A big contribution to the development of this theory came from von Neumann, or John Nash with the concept of Nash equilibrium [34].

- Cooperative games (or coalition games) concern competition between groups of player forming coalitions. Each coalition plays as a single participant and earns a payoff. One of the the main tasks related to cooperative games is to find a way to divide the total utility among the member of a coalition in an equally way proportionally of how much they contributed to the final score.

In 1951 Lloyd Shapley introduced a way to compute the exact amount of payoff for each player, making use of what were named after him: Shapley values [**?**][**?**]. Shapley introduced those values in the field of coalition games, therefore to properly understand his work, we need to briefly illustrate what a coalition game consists of.

A coalition game involves N players and different subsets, called coalitions, created from these N players. Each subset of players S gains a payoff at the end of the game. A function ν maps every subset to its payoff $\nu(S) = \text{payoff(S)} \in \Re$. On the basis of which player had more influence in this final score, we ask how to split this payoff in a fair way between each player of the subset, where "fair "is to be intended as, proportional to his own contribution. A solution for this problem comes from **Shapley values** $\phi_i(\nu)$, specific for each player $i \in N$ in a coalition $S \subseteq N$. The idea behind them is the marginal contribution of that player to the final score, where marginal contribution is defined as the difference on the score of the coalition when player i joins the coalition. In other words they are the difference between the coalition's score with player i and the coalition's score without him marginal contribution $= \nu(S \cup \{i\}) - \nu(S)$. [1]

Shapley defined these coefficients (Shapley values) for a player i as a weighted average of marginal contribution values, over all the possible subsets that include player i.

The mathematical formulation that Shapley introduced for $\phi_i(\nu)$ is

$$\begin{aligned}\phi_i(\nu) &= \frac{1}{N} \sum_{S \subseteq N \setminus \{i\}} \binom{N-1}{|S|}^{-1} [\nu(S \cup \{i\}) - \nu(S)] \\ &= \sum_{S \subseteq N \setminus \{i\}} \frac{S!(N-1-S)!}{N!} [\nu(S \cup \{i\}) - \nu(S)]\end{aligned} \tag{4.1}$$

which exactly represent the amount of reward for each player i.

MATERIALS AND METHODS

Chapter 5

Dataset: ABIDE I & II

Data we are going to work on belong to the ABIDE dataset (Autism Brain Images Data Exchange): a project founded with the aim of investigating ASD using magnetic resonance structural images and resting state fMRI scans.

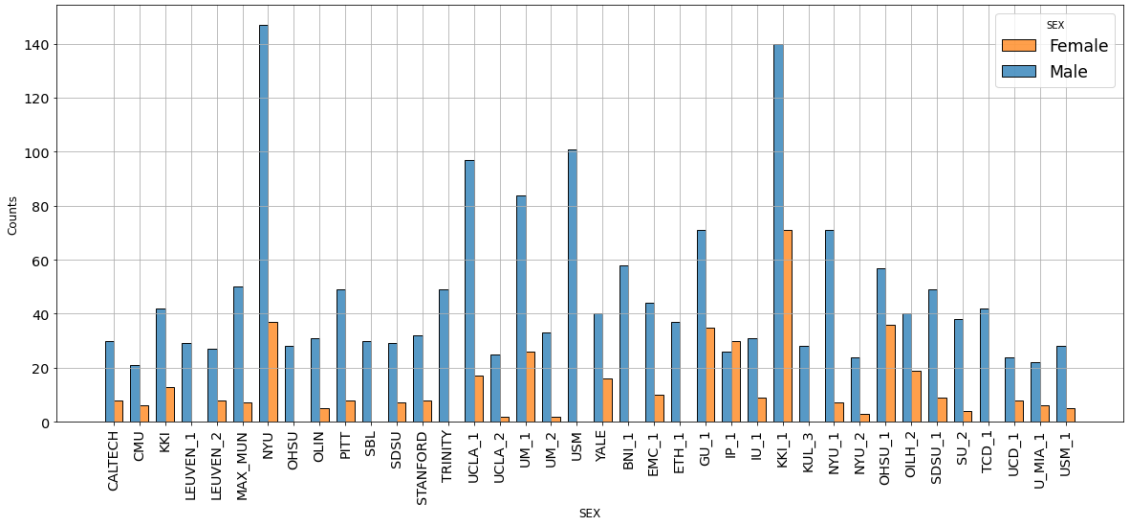
The whole ABIDE dataset was published in two releases: ABIDE I released in August 2012 and containing 1112 patient scans, and ABIDE II released in June 2016 containing 1114 scans. For each site autism was diagnosed either by gold standard diagnostic tests, clinical judgment or a combination of clinical gold standard procedures.

ABIDE I includes scans collected from 17 different sites, and the 1112 patients consist on 539 patients with ASD and 573 typical control patients. ABIDE II includes scans collected from 19 different sites, and the 1114 patients consist on 593 patients with ASD and 521 typical control patients. Not every site belonging to ABIDE II is different from those of ABIDE I, but, even though some medical centers are the same, the scanner type, or acquisition pipeline and parameters may have been changed during the time interval between the two releases, so in all our analysys, they are regarded as different acquisition sites.

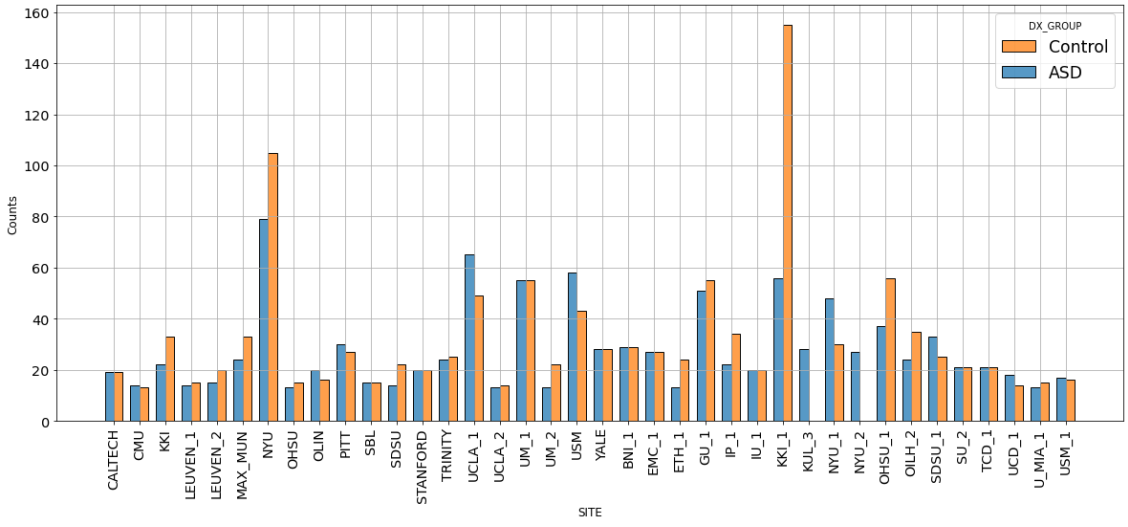
In addition to scan images, ABIDE provides every information related to each patient, as age, sex, intelligence quotient (FIQ), eye status during the scan (open or closed), and every additional clinical information provided by patients.

The vast majority of patients are males as shown in figure 5.1a, for a total amount of 1804 males

and 422 females. The number of males is greater than the number of females as a consequence of the greater probability for male to be affected by ASD, the ratio between males and females has been estimated to be approximately 4:1 [?]. Control/ASD patient number is more or less balanced for every site, with the exception of KKI_1 (Kennedy Krieger Institute) that provided two times more controls than ASD patients, and KUL_3 (Katholieke Universiteit Leuven) and NYU_2 (NYU Langone Medical center, Sample 2) that only provided ASD cases. For a visual comparison, the number of controls/ASD for each site is displayed on the histogram in figure 5.1b.



(a) *Histogram of male and females subjects per site from the whole ABIDE I & II dataset*



(b) *Histogram of control and ASD subject per site from the whole ABIDE I & II dataset*

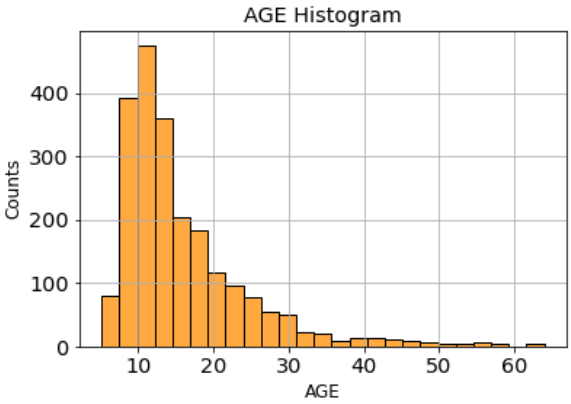
Figure 5.1

Patients in ABIDE dataset have ages ranging from 4 to > 50 years old, but as shown in the distribution in figure 5.2a the vast majority of participant are younger than 40, precisely > 97 % of participant are under 40 y.o., also, the majority of sites provide only young patients in a restricted age range, but as can be seen from figure 5.2b there are some sites that acquired patients with a wide age range (MAX_MUN or BNI_1 for example).

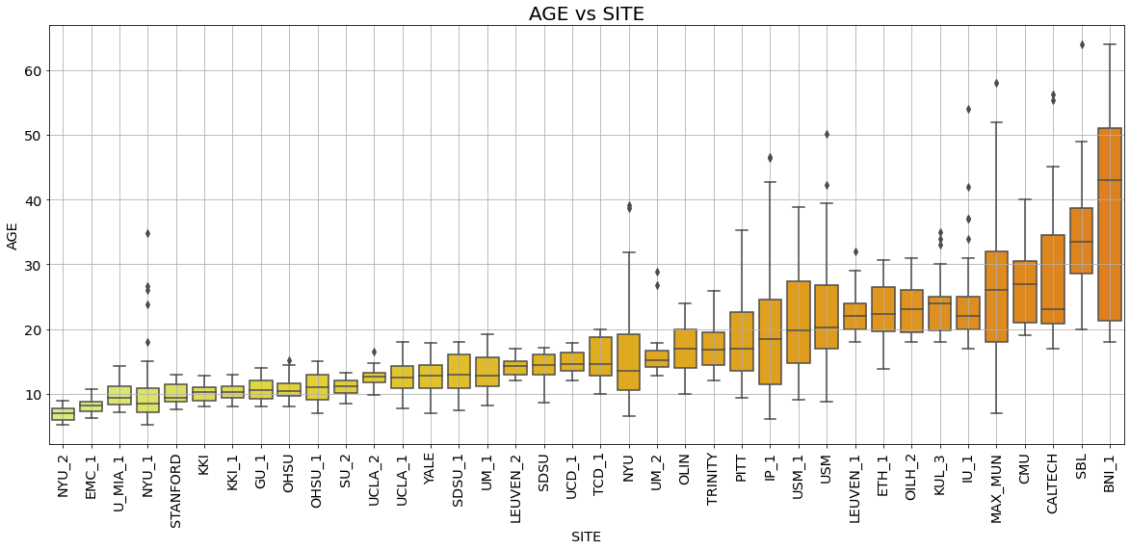
Intelligence quotient, whose distribution is shown in figure 5.3a, was not provided for every participant, in fact 171 patients out of 2226, coming from sites UM1 and EMC1 (figure 5.3b) lack this information. In our further analysis, before proceeding these values were replaced by the average value of all the other provided values.

The lack of a common acquisition protocol is also evident from the eye status at scan feature: as shown in figure 5.4a each site acquired scans either with open eyes or with closed eyes, without a common procedure, and sometimes this information is not even specified. Overall, the whole dataset consists of more than 70% of patient acquired with open eyes, as shown in figure 5.4b.

Considering all the information above, we can limit our further analysis in order to work on a more homogeneous dataset. In this way, we try to remove from the very beginning some source of variability due to unavoidable difference due to sex or age. We have then carried out our analysis on a dataset consisting on only male patients with an age within 5 and 40 years. Some further analysis were performed with further cut on eye status at scan, selecting only patients with open eye.

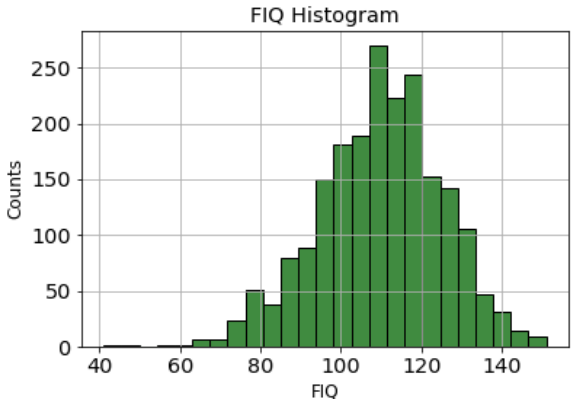


(a) ***Age*** *distribution of the whole ABIDE I & II dataset*

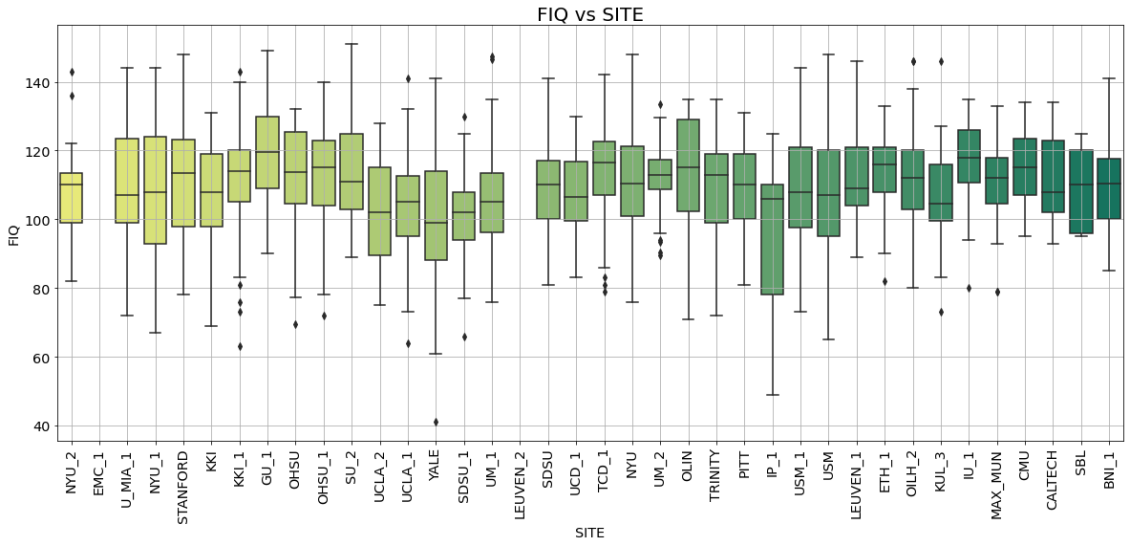


(b) *Boxplot of* ***age*** *of patients per site*

Figure 5.2: *Histogram distribution and boxplot of AGE (jointly for Controls and ASD) per site, from ABIDE I + II*

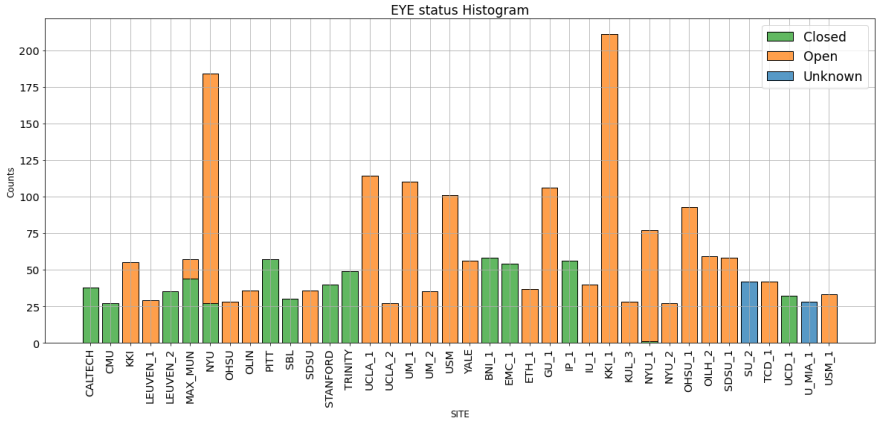


(a) ***FIQ*** *distribution of the whole ABIDE I & II dataset*

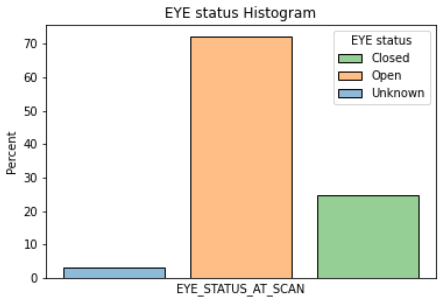


(b) *Boxplot of patient's* ***FIQ*** *per site*

Figure 5.3: *Histogram distribution and boxplot of FIQ (jointly for Controls and ASD) per site, from ABIDE I + II*



(a) *Eye status distribution per site of the entire ABIDE dataset*



(b) *Overall eye status distribution on the entire ABIDE dataset*

Figure 5.4: *Histograms of patients with open (orange), closed (green) and unknown (blue) eye status in the whole ABIDE I & II dataset*

Chapter 6

Image preprocessing

6.1 Common preprocessing steps

Data acquired after a MRI and fMRI session are not ready to be analyzed to perform functional connectivity measures. In fact they are usually affected by different sources of noise and artifacts. There are two primary noise sources: the first is due to normal biological functions of patients such as heartbeat and breathing, commonly referred as physiological noise. For example, breath rate, can affect BOLD signal because of the induced local motion of the brain's vessels, or the change in blood oxygenation and pressure. The second noise source is due to movements of the patients during the scan, such as head movements or small body adjustments. Other common artifacts usually derive from the hardware scan as well. One of the main artifact intrinsic to the signal nature is distortion and field inhomogeneities, deriving from making the scan sensitive to the BOLD signal which intrinsically is the detection of a signal loss due to field distortion. This could be corrected by employing small coils inside the scan to smooth magnetic field differences, this process called shimming still left some artifacts, therefore common data preprocessing pipelines use extra acquisitions to create a field map of the remaining field inhomogeneities and a shift and stretching voxel to correct for them.[5]

In the following lines, we list the most important causes of noise and how they are corrected during common preprocessing steps implemented in the main software of analysis.

- Head motion effect is due to the physical movement of the patient inside the scanner. It results in a misalignment from one acquired volume to the next. To correct this effect, **motion correction** steps are performed at the beginning of image preprocessing. Motion correction works by spatially applying transformations as rotation or translation volume by volume, aiming to overlap every acquired slice to a chosen reference volume, like the first or the middle one.

- For EPI data, **slice timing** correction is usually performed as well. It aims to correct artifacts deriving from the sequentiality of acquisition for each slice of the brain, due to which each slice is acquired at different time. In EPI images, the entire time elapsed to acquire all the slices is called repetition-time, and it usually ranges from 1 to 3 seconds, and during this interval signal may lose its initial strength. Slice timing correction uses interpolation in time to shift the BOLD timeseries of each voxel, in order to align them to a reference starting time. A downside of the use of interpolation, though, is the signal smoothing it performs that can lead to a slight loss of high frequencies information.

- A further common step is **spatial smoothing** of both structural and functional data. It operates calculating a weighted average of each pixel over neighbored voxels. To this end, a gaussian kernel with a chosen FWHM is applied to create the weights. Spatial smoothing is useful to avoid abrupt changes of signal between two neighbouring voxels.

- **Band-pass temporal filtering** is commonly applied to BOLD data, aiming to reduce artifacts from hardware such as an effect called "drift "namely the slow change of the BOLD baseline signal over time. This is accomplished by applying a high-pass filter to remove very low frequency components. This filtering operation is usually referred as a removal of linear trends, since we are removing from a signal only components with a frequency lower than the low-frequency fluctuations of BOLD signal. At the same time, a low-pass filter is applied to remove all frequencies above a cut off frequency, it is commonly applied in processing resting state fMRI data because the physiological signal is driven by low frequencies oscillations while high frequencies are associated to noise.

All the preprocessing steps cited above are performed for each patient, and the main objective of them is to enhance signal to noise ratio of each image. However, if our task is to perform a group analysis, and compare different patient's images, one of the most relevant step is **registration**. Structural (T1-weighted) data are aligned over a standard coordinates system space to universally describe location of the different brain parts, to make sure that the same voxel coordinate corresponds to the same brain area for all the subjects. Registration occurs for functional images as well: they are firstly aligned and adapted to the structural image and then registered to the same standard template. The most common template, provided by packages like FSL is Talairach and Monreal Neurological Institute's MNI-152, adopted by the International Consortium of Brain Mapping (ICBM) as the international standard, replaced the previous Talairach and Tournoux template. Because the adoption as international standard it is also called ICBM152 but the most common name remains MNI-152.

To understand how MNI152 differs from the Talairach and Tournoux we have to spend a few words about the latter.[7]

From Talairach-Tournox to MNI152 template

The Teilarch and Tournoux space was published in 1988 and introduced some innovative aspects to tackle the problem of the great variability in brain anatomy between people which so far had limited the accuracy of statistical analysis. They introduced a common coordinate system to identify different brain locations, based on some anatomical landmarks, and introduced a spatial transformations to match different brains. They choose two anatomical areas as reference areas: the anterior and the posterior commissure, which are relative invariant between different brains and conjuncted them with an axis. An horizontal plane passing through this axis was chosen such that it was perpendicular with the interemishpere axis. This way they created a 3D coordinate system called the Talairach coordinate system. Afterwards, to match different brains they described a set of spatial transformation, one for each different brain quadrant, to transform a brain in order to match the principal anatomical structures of each other.[6]

From this template, a first MNI template was created, called MNI305, created by manual scaling 241 brains to the Talairach template and averaging them to obtain new template, and then transform 305 additional scans to this new template and averaging them to create a second average and final template (MNI305).

From this template, MNI152 was finally created and published in 2001, by registering 152 T1 scans to the MNI305 template and averaging them.

This template is used both for the structural and the functional registration of a MRI and fMRI scan.

Looking at picture 6.1 we can see an example of functional and structural image registration to a MNI152 template: first the functional image is registered to the structural and next, they are both registered to MNI template.

Once the structural and functional images were registered to a standard space, is possible to extract brain region information using an **atlas**.

Atlases are in the same space as the template image (MNI or Talairach space for example) and consist of a 3D standard brain template where different brain areas are marked with different color intensities to associate each area to a label. This division into areas and the subsequent labeling, is called parcellization.

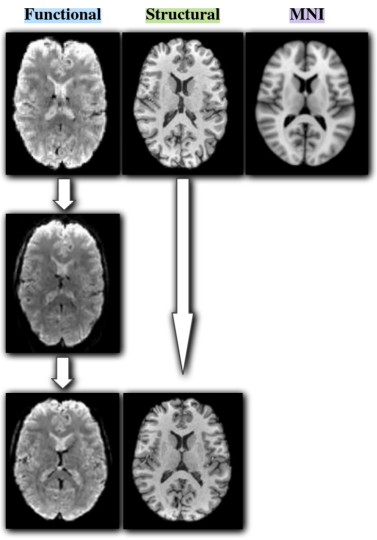


Figure 6.1: *Registration steps: starting from the first row there are the acquired functional and structural images and the MNI template respectively The second row shows the functional image registered on the structural space The third row shows the final images both registered to the MNI 152 template*

There are different atlases, each one including a different parcelization of the brain, which is obtained by divding it into N labeled ROIs (Regions Of Interest) to focus on anatomical and/or functional regions, according to the study we are carrying out. We will describe some og the most important atlases in the following section (6.2)

6.2 Preprocessing of ABIDE I & ABIDE II dataset

ABIDE-preprocessed initiative

In 2013 Neuro Bureau Preprocessing Initiative took care of data preprocessing of ABIDE I dataset and shared its results making them publicly available [9]. Thanks to this work, ABIDE I data are available as raw data, as well as preprocessed data. Four preprocessing software and approaches were employed each from a different group and every one publicly available [1] for download. The four software for image preoprocessing employed for ABIDE I are:

- Connectome Computation System (CCS)

[1] http://preprocessed-connectomes-project.org/abide/C-PAC.html

- Configurable Pipeline for the Analysis of Connectomes (C-PAC)
- Data Preprocess Assistant for Resting-State fMRI (DPARSF)
- Neuroimaging Analysis Kit (NIAK)

The preprocessing steps implemented by the different softwares are similar, they differ on the programming language on which they are rely (Python, MATLAB..) the algorithm implementation, the order of preprossessing steps and their parameters.

But since this work only concerns ABIDE I dataset, and our task is to work with the entire ABIDE I + ABIDE II dataset, we to repeat the preprocessing procedures in order to obtain a dataset including both ABIDE I and II preprocessed with the same pipeline.

C-PAC the Configurable Pipeline for the Analysis of Connectomes

In our work, we choose to preprocess data from ABIDE I and ABIDE II using C-PAC: a configurable, open source pipeline, based on Nipype platform. C-PAC was run on a Docker container installed on a computer and our hardware and software setup consisted on:

- 16-core Intel i7-5960X processor and 64 Gb RAM
- Ubuntu 20.04 operating System
- C-PAC 1.8.1 installed on Docker v. 20.10.11

We were allowed to run 3 patients in parallel, reserving 4 cores for each partecipant and up to 12 Gb memory to each patient, necessary to save intermediate-steps outputs.

We choose to employ C-PAC because, basing on machine learning classification results, [55] C-PAC prove to be the most efficient preprocessing pipeline to preprocess ABIDE dataset. C-PAC employs tools like AFNI, FSL and ANTS to perform image correction of structural MRI and rs fMRI. Data were preprocessed following the same steps as the pipeline employed with C-PAC to create the ABIDE-preprocessed dataset. This pipeline includes both antomical and functional preprocessing. Anatomical pipeline steps consist on:

- Skull removal using AFNI's 3dSkullStrip

- Tissue segmentation using FSL-FAST, to separate gray matter, white matter and CSF, uging a thresholding probability map whose threshold's values were set the same as ABIDE-preprocessed pipeline values
- Registration to a standard template using ANTS, with a spatial resolution of 2mm

Functional pipeline consists on the following steps

- Slice timing correction using AFNI-3dTshift
- Motion estimate and correction using AFNI-3dvolreg
- Distortion correction using PhaseDiff and AFNI 3dQWarp
- Create a brain-only mask of the functional data using AFNI 3DAUTOMASK

At the end of functional preprocessing steps, timeseries are extracted from each patient, making use of different atlases such as Automated Anatomical Labeled (AAL), Harvard-Oxford (HO), CC200, CC400, DesikanKilliany etc. As mentioned in section 6.1, different atlases differ in numbers and types of ROIs. Here we list some examples

The Automated Anatomical Labeled, an atlas created in 2002 [52] consisted of 90 total ROIs, 45 each emisphere, since then, different modified and improved version were released, the last of which AAL3 consisting of 166 ROIs. The AAL atlas employed in C-PAC is an intermediate version and consists on 116 ROIs[2].

The Desikan atlas [16] originally created in 2006 and consisting on 68 regions also underwent different transformation towards a finer parcellization of the brain. C-PAC employs three different Desikan atlases, the one we found more information about is a modified intermediate version named DesikanKilliany consisting on 94 ROIs of which 32 cortical regions each side, 3 ROIs belonging to cerebellar vermis and 29 subcortical regions.

CC200 and CC400 are more recent atlases created by C. Craddock in 2012 [14] for functional parcellization, consisting of 200 and 392 ROIs respectively [3].

[2]http://preprocessed-connectomes-project.org/abide/Pipelines.html

[3]http://ccraddock.github.io/cluster_roi/atlases.html

However, according to previous studies [48] the atlas that gave best classification performances was Harvard-Oxford, so this was the atlas we choose to employ for our work. The H-O atlas consists of a subcortical and a cortical atlas, with a total of 117 ROIs of which 21 subcortical and 96 corticals (48 for each emishpere). The H-O atlas employed by C-PAC is provided by FSL library [4], and as includes both subcortical and cortical atlases, even though there's only 111 ROIs, namely 14 subcortical and 97 corticals. The lacking ROI in the subcortical areas are L/R cerebral white matter, L/R cerebral cortex, L/R lateral ventricle and the brain stem [5]. Removing these regions from the 117 ROIs we should obtain 110 ROIs, while the atlas employed in C-PAC has 111 regions. The extra cortical ROI in this H-O atlas is present in the 83th position and is named 3455; however there are no information abut this ROI neither on Harvard-Oxford documentation nor on C-PAC's/FSL's documentation, and because is only made of 2 voxels it was excluded for our further analysis. In summary we are able to obtain 110 ROIs with Harvard-Oxford atlas, and for each ROI we are able to extract a timeseries using C-PAC.

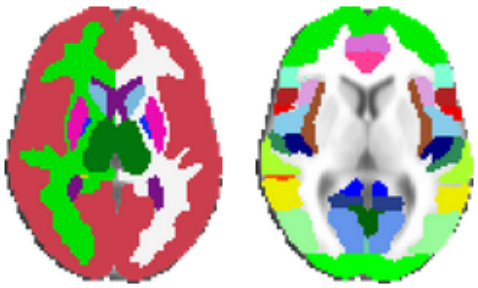


Figure 6.2: *Horizontal section of Harvard-Oxford subcortical (sx) and cortical (dx) atlased*

As is possible to notice from figure 6.3 timeseries extracted after our preprocessing pipeline do not exactly match those from ABIDE preprocessed, even if it appears clear that the trend is the same, but there are some local differencies between the two plots. This is most likely due to two main reasons: the differences in software version, and the lacking of a detailed step-by-step pipeline legend showing the value of all the sub parameters employed during the C-PAC analysis pipeline. Abide preproccessed data were obtained using one of the first version of C-PAC; nowadays,

[4]https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/

[5]Subcortical ROIs https://neurovault.org/images/1700/ Cortical ROIs https://neurovault.org/images/1705/

after more than 7 years, C-PAC and the libraries it relies on were upgraded several times and this may have slightly affected the final outcome of the process. We found though the old pipeline configuration file employed on abide preprocessed [6], but since it belongs to a previous version of C-PAC, it lacked lots of sub-parameters and sub-settings added during these years. For this reason, in our pipeline configuration file, parameters in common between our file and abide preprocessed' were set the same as abide's, and for all the others we choose to left the C-PAC's default values.

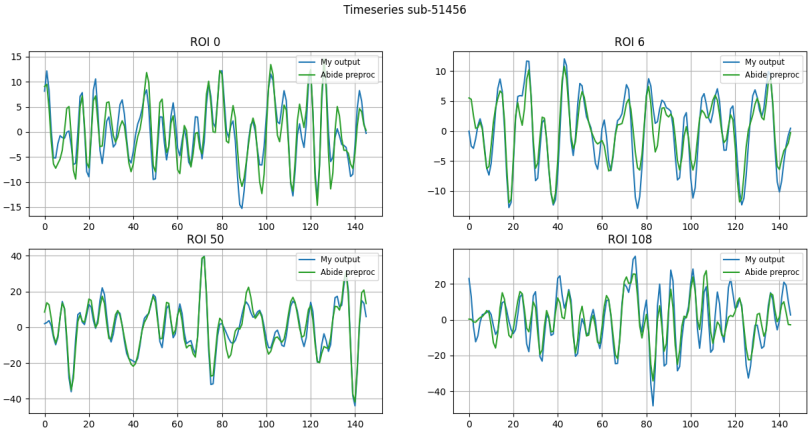


Figure 6.3: *Comparison of 4 timeseries between ABIDE-preprocessed and our timeseries. Data show 4 randomly chosen ROIs: ROI 0, 6, 50, 108, belonging to patient 51456 from ABIDE I. Corresponding to ROI 10, 26, 1901, 4702*

[6]https://github.com/preprocessed-connectomes-project/abide

Chapter 7

Connectivity measures

As described in chapter 6.1 if we choose an atlas on fMRI data, for brain parcellization, we can extract timeseries of different brain areas. If we use Harvard-Oxford atlas, we are able to extract 110 timeseries for each patient. With these data, we want to construct a correlation matrix for each patient, by pairwise comparing timeseries and extracting a coefficient representative of functional correlation between two areas.

The total number of combination achievable with n timeseries is given by

$$N_{comb} = \frac{n \cdot (n-1)}{2} \tag{7.1}$$

Therefore, employing HO atlas with 110 ROIs we obtain 5995 combination each one expressed by a correlation coefficient.

In the following section we discuss two different approaches for computing correlation coefficients: the first makes use of Pearson correlation analysis, a useful tool to quantify the linear correlation between two timeseries, and the second approach, based on wavelet analysis, performs a comparison between two signals in time-frequency domain. We start discussing Pearson correlation coefficients in the following section and in the next section we describe the main traits of wavelet analysis and how we extracted a correlation coefficients from a time-frequency analysis of two timeseries.

7.1 Pearson correlation and Z-Fisher transform

Pearson correlation often simply called correlation coefficient is the measure of a linear relation lying between two sets of data x_1 and x_2, is defined as the covariance of (x1, x2) over the product of the standard deviation of the two sets. Pearson correlation coefficients have the important property to be scale and magnitude invariant, since each timepoint is shifted by the average value of the timeseries and divided by its standard deviation.

$$Corr(x, y) = r_{xy} = \frac{Cov\,(x, y)}{\sigma_x \sigma_y} = \frac{E\left[(x - \mu_x)\,(y - \mu_y)\right]}{\sigma_x \sigma_y} \tag{7.2}$$

This correlation coefficient assumes values between -1 and +1 where the extremes correspond to exact anti-correlation or correlation respectively, so that, if a linear relation lies between the two sets, a high absolute value indicates that the two series tend to be simultaneously greater or lower than their respective means. [3]

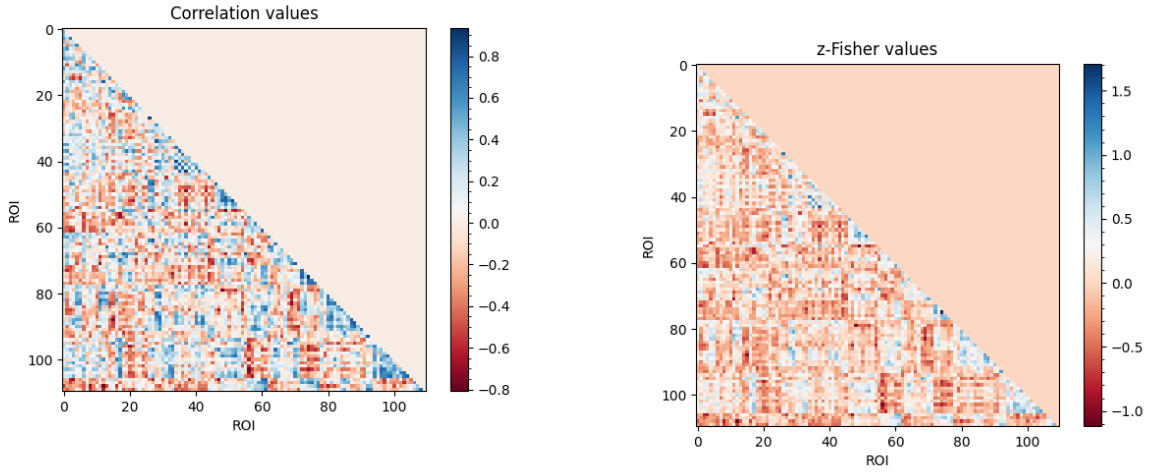
Given two series x and y, of length n correlation can be easily computed by

$$r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \mu_{x_i})\,(y_i - \mu_{y_i})}{\sqrt{\sum_{i=1}^{n} (x_i - \mu_{x_i})^2}\sqrt{\sum_{i=1}^{n} (y_i - \mu_{y_i})^2}} \tag{7.3}$$

For each patient Pearson correlation coefficient was computed for all timeseries pairs. Before further analysis, a common way to proceed is to transform each coefficient with Fisher z-transformation.[48] The reason behind this common operation appears clear when we are dealing with high correlate variables, when Pearson correlation distribution results in an highly skewed distribution, Fisher's transform sought to transform it into a normal distribution of which the standard error is approximately constant equal to $\sigma = \sqrt{N - 3}$ where N is the total number of points, and it does not depends on the values of correlation.[54]

With this property, Fisher transform is also important to test some hypotesis about correlations, we can run the test with the transformed variables which are normal distribuited with a known variance.

Thus, this transformation allows us to obtain a variable which is normally distribuited even when Pearson correlation coefficients follow a bivariate normal distribution or they are leaning toward the extremes. In our data, this difference in distributions is not pronounced, one example where



(a)

Figure 7.1: *Correlation and z-values matrix computed from timeseries extracted using Harvard-Oxford atlas, from patient 20243 belonging to ABIDE I dataset.* ***Note:*** *The two figures are not on the same values scale because Pearson correlation coefficients have range [-1, 1], and z-scored coefficients have range* $(-\infty, \infty)$

this difference is evident is shown in figure 7.2, but since we are not working with highly correlated or uncorrelated variables, there are many other examples where pearson correlations already follow a gaussian distribution and there is not much difference with z-score values distribution. In any case, we are dealing with a bivariate distribution, and then performing Fisher transformation is a common practice to get a dataset more normally distributed.

$$z = \frac{1}{2} ln \left(\frac{1+r}{1-r} \right) = arctanh(r) \tag{7.4}$$

At the end of this analysis, we obtain correlation matrices like the one shown as an example in fig 7.1, referred to patient 20243 from ABIDE I dataset.

7.2 Wavelet analysis

A different approach to compute a correlation coefficient is by making use of wavelet analysis.[19] [53] wavelet is a mathematical tool for analyzing time series or images, and provides a comprehensive way for investigating the bivariate relationship between timeseries both in time (or space) and

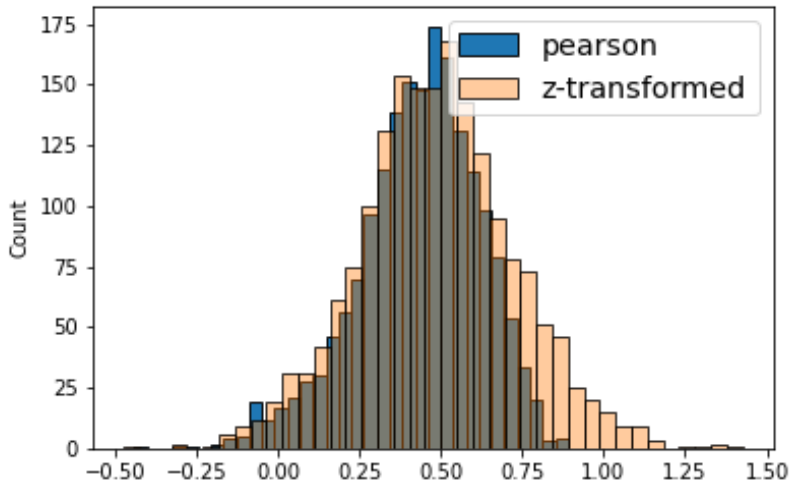


Figure 7.2: *Distribution of feature 3: Pearson correlation values in blue and Fisher transformed values in light orange*

frequency domain.

To understand wavelet transformation it could be helpful to compare it with Fourier analysis. Fourier analysis allows us to expand a periodic function f(t) into a series, ideally infinite sums of weighted sines and cosines with different frequencies

$$f(t) = a_0 + \sum_{k=1}^{\infty}(a_k cos(2\pi kt/T) + b_k sin(2\pi kt/T)) \tag{7.5}$$

In equation 7.5 terms corresponding to the k-th frequency are called harmonics and are multiples of the fundamental frequency corresponding to k = 1.

The Fourier Transform (FT) is a natural extension of the Fourier series to aperiodic functions defined over the real axis. Aperiodic functions don't allow a discrete superposition of sines and cosines terms, and for this reason they need to be represented by a continue superposition. Fourier transform takes a function from time or space domain and turns it into spatial or temporal frequency domain. It is a complex function since sines and cosines terms can be represented by a complex exponential as shown below in equation **??**.

$$\tilde{F}(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}dx \tag{7.6}$$

To deal with discrete signal, for example a sequence x_n obtained from a discrete sampling of a continuous signal x(t), is possible to make use of the Discrete Time Fourier Transform, wich allows us to write the $\tilde{F}(\omega)$ as in equation 7.7 which represent the discrete version of 7.6.

The Discrete Fourier Transform, is useful to analyze discrete periodic signal, it acts on a function defined over a finite domain and returns a sequence of samples of the discrete fourier transform we can denote as $\tilde{x_k}$

$$\tilde{F}(k) = \tilde{x}_k = \sum_{n=0}^{N-1} x_n e^{\frac{-2\pi i k n}{N}} \tag{7.7}$$

where N is the total number of points of the timeseries x(t). The frequencies at which samplings are computed are $\omega_k = 2\pi k/N$. The series we obtain from relation 7.7: $\tilde{x}_k$ is periodic and its period is equal to N.

One of the main drawback of FT though, is the lacking of spatial information, for example for a non stationary signal, is possible that the signal contains a variable component of frequency in different space or time locations.

A time-frequency analysis provides for this lacking computing both frequency and spatial information from a signal, allowing us to analyze non stationary signals and obtain informations both on time (or space) and frequency domain. Since we are working on signal defined over a time domain, from now on we would refer only to "time "and "frequency "domain, but we keep in mind that all the relation are true for spatial and spatial frequency domain as well.

The process through which we exctract informations about frequencies (F) and time location (T) is a convolution of the signal x(t) with a function w(t) called windowing function usually centered at zero which rapidly decays to zero, often symmetrically.

$$\tilde{x}(F,T) = \int_{-\infty}^{\infty} x(t) w(t-T) e^{-2\pi i F t} dt \tag{7.8}$$

The convolution term $w(t-T)e^{-2\pi iFt}$ changes according to the type of analysis we are performing. Parameters F and T are often simply called a and b, where b is a time translation parameter,and acts by simply shifting our convolution function throughout the entire timeseries x(t). It acts the the same way through all the type of analysis. What differenciate the most the different analysis

isthe parameter "a ". It is the frequency parameter and the way it is introduced in a convolution determines the developing of the transform. The two main types of time-frequency analysis are

1. Windowed Fourier Transform (WFT): $\psi_{ab}(t) = e^{it/a}\phi(t-b)$ where $\phi(t)$ is a window function of constant width and the parameter a acts as frequency modulation (freq $\sim$ 1/a): the lower is a, the greater the number of oscillation inside the window $\phi(t)$.

2. Wavelet Transform (WT): $\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$ where ψ is a function called *mother wavelet*; a is a positive real number and defines the dimension of ψ: with a>1 we obtain a dilation and a<1 corresponds to a contraction. b is any real number (positives and negavies) and defines the location of the wavelet in time.

In Wavelet Transform, the equivalent to the window function in WFT is a wavelet function ψ.

Wavelet

A wavelet, literally "small wave "is a wave function limited both in space and period: begins at zero, grows and decrease in a limited time period and returns to zero. With these properties, is a function local in the temporal domain as well as in the frequency domain.

To be defined as so, a wavelet is required to satisfy two properties: have zero mean (it must be oscillating) and unitary squared norm [38].

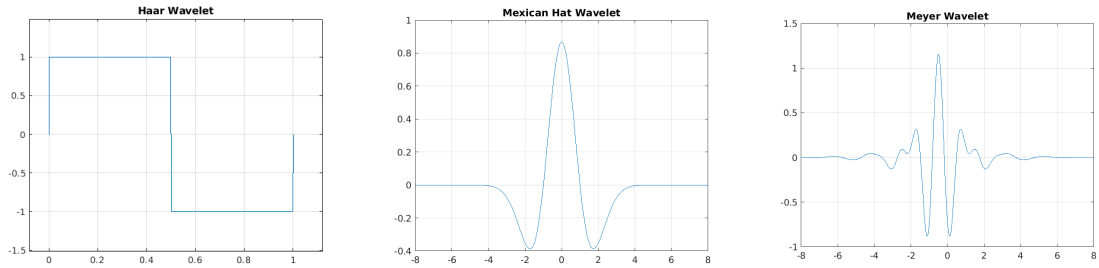
$$\int_{-\infty}^{\infty} \psi(x)dx = 0 \qquad \int_{-\infty}^{\infty} |\psi(x)|^2 dx = 1 \tag{7.9}$$

It is defined over the space L^2 of Lebesgue measurable functions that are both absolutely integrable and square integrable. In this space the two properties can be satisfied.

There are different wavelets that satisfy properties 7.9, such as Haar (fig 7.3a), Meyer (fig 7.3c) [15], Mexican hat (fig 7.3b) or Morlet (fig 7.4) [1].

What we are going to use in our analysis is Morlet wavelet, defined by equation 7.10 and shown in figure 7.4. For each wavelet, there's a trade-off between time ans frequency resolution: compressing the wavelet in a shorten time domain improves time resolution vut at the cost of frequency resolution

[1] https://it.mathworks.com/help/wavelet/gs/introduction-to-the-wavelet-families.html



(a) ***(b)*** ***(c)***

Figure 7.3: *Visual comparison between three different types of mother wavelets, Haar wavelet (fig 7.3a) is a simple square wave function, mexican hat (fig 7.3b) is a wavelet belonging to gaussian function family and represents the negative normalized second derivative of a gaussian function; and Meyer wavelet (fig 7.3c) a wavelet with applications in fields like adaptive filters*

and vice vice-versa. With this wavelet this trade-off between time and frequency resolution can be controlled by the choice of ω_0.[32]. We choose to run our analysis with a value of $\omega_0 = 6$ since it provides a good balance between the two resolution [24] and besides, this value gives the best ratio between Fourier Period and the scale parameter a during a Transform, equal to $\lambda = 1.03$. This way results in frequency domain are more interpretable since the scale parameters a used for the Transform is almost equal to the Fourier period.

$$\psi(t) = \pi^{-1/4} e^{iw_0 t} e^{-t^2/2} \tag{7.10}$$

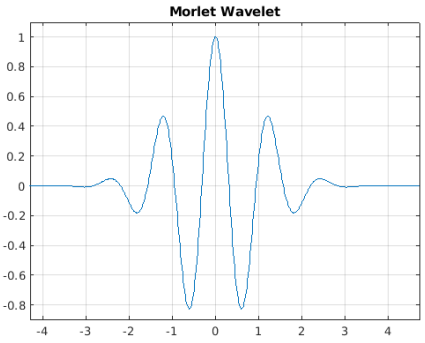


Figure 7.4: *Morlet wavelet*

Two main type of analysis can be performed using wavelets: continuous wavelet transform

(CWT) and discrete wavelet transform (DWT) and the CWT is what we employed for our analysis.

Continuous wavelet transform decomposes a time series in time-frequency domain by successively convolving the timeseries with several scaled and translated version of the mother wavelet ψ: $\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$. If the timeseries is described by a function f, assumed to be real in the equation below, the convolution of f and $\psi_{a,b}(t)$ is

$$W_{a,b}(f) = \int_{-\infty}^{+\infty} f(t) \cdot \psi^*_{a,b}(t)\, dt \tag{7.11}$$

Where the $*$ indicates the complex conjugate. This integral is performed for different values of a and b. It can be useful to visualize this integral in a dynamic way: set a value for a, a wavelet centerd in b, is slided across the signal by changing the value of b and for each of these values, a coefficient is extracted by integrating the product between the wavelet and the signal; in this way, coefficients are function of frequency (or scale) and time. This operation is repeated for different values of a and b, and allows us to assemble a matrix called *scalogram*, which is basically a plot of these coefficients in time-frequency domain.

Formally, a wavelet transform, can be described as a mapping from $L^2(\Re) \to L^2(\Re^2)$.

To computationally implement CWT, a discretized version was implemented on MATLAB tools Wavelet toolbox, but it does not be confused with the Discrete wavelet transform [2], a similar type of analysis but with some important differences. The difference between the continuous wavelet transform implemented on MATLAB and discrete wavelet transform lies in how finely stretching and shifting parameters are sampled: the CWT discretizes scale more finely than the discrete wavelet transform.

In the CWT, parameters are discretized based on a fractional power of two, by setting a $= 2^{j/\nu}$ ([3]) [30] [15] where ν, j are integers. The parameter ν is often referred to as the number of "voices per octave" because increasing the scale by an octave (namely to double the frequency) requires ν intermediate steps, for example from $f = f_0 2^{\nu/\nu}$ *to* $f = f_0 2^{2\nu/\nu}$, ν steps are required. The larger the value of ν, the finer the discretization of the scale parameter. In the DWT, the scale parameter

[2] https://it.mathworks.com/help/wavelet/gs/continuous-and-discrete-wavelet-transforms.html

[3] This way, the discretized wavelet can be written as $\frac{1}{2^{j/\nu}}\psi\left(\frac{t-b}{2^{j/\nu}}\right)$

is always discretized to integer powers of 2, 2^j, j=1,2,3,..., so that the number of voices per octave is always 1. Since it employs a more rough discretization, DTW is usually used for denoising and compression of signals and images.

In our analysis we are going to use DTW implemented in matlab with the default parameters of 12 voices per octave.

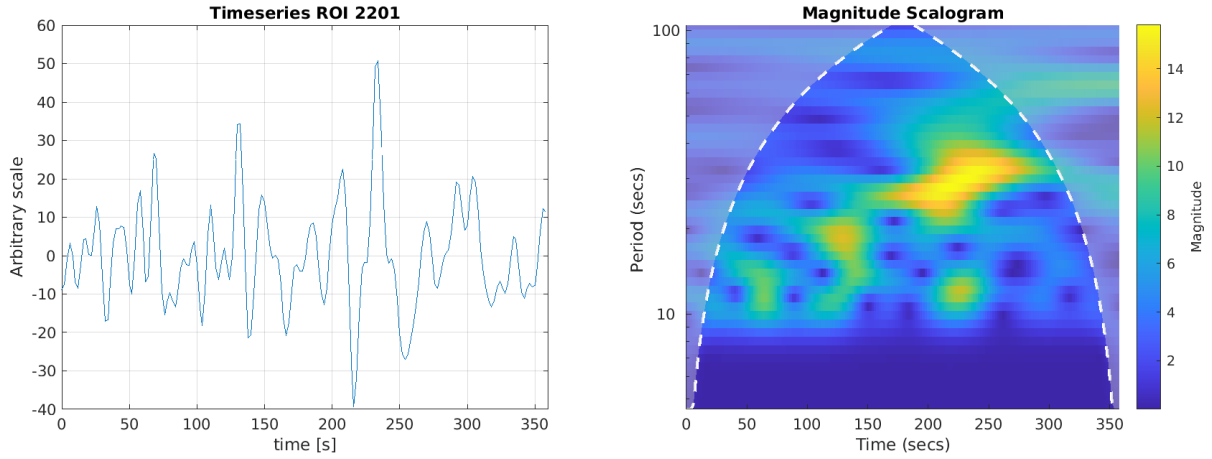
One problem that arises from working on a timeseries, which is basically a signal with a finite support, is that when a wavelet is located at the beginning of at the end of the signal, the wavelet extends itself outside the boundary of the signal, and the convolution would require nonexistent values beyond the boundary.

To overcome this problem it would be possible to accept this data information loss and truncate the values beyond boundaries; whereas an other approach could be to artificially extend data using methods such as zero padding which assumes that the signal is zero outside its original support, or symmetrization which extend the signal symmetrically outside the boundaries or smooth padding which recovers a signal by extrapolating values from the first derivative values or from the signal itself. Symmetrization method is the one employed for the subsequent analysis. The confidence value obtained on the boundaries, though, is lower than other obtained for central values of time location. Areas of the scalogram affected by these edge effects are indicated as "outside the Cone of influence (COI) ".

Figure 7.5 shows the timeserie belonging to ROI 2201: Right Angular Gyrus of patient 51056 from ABIDE I, and its corresponding Continuous Wavelet Transform. x axis corresponds to time points and on the y axis frequencies or periods (derived from the parameter a) are represented. Just as an example, we choose to display the y axis as periods. The COI is shown as a dotted white line and values outside the COI, where edge effect becomes effective are shown with a lighter faded. The color is a visual representation of the magnitude of each wavelet coefficient.

Wavelet coherence

Continuous wavelet transform can only be used to analyze one signal at a time; if our goal is to analize and compare two signals using wavelet transform, the analysis to perform is called wavelet



(a) ***(b)***

Figure 7.5: *Timeseries (fig 7.5a) of ROI 2201: Right Angular Gyrus of patient 51056 from ABIDE I, and its corresponding Continuous Wavelet Transform (fig 7.5b). x axis corresponds to time points and y axis the periods derived from the scale parameter a of the wavelet convolving function. The COI is shown as a dotted white line and values outside the COI, are shown with a lighter faded. The color represents the magnitude of each wavelet coefficient.*

coherence.

From two Continuous Wavelet Transforms is possible to compute the Cross Wavelet Transform (XWT) which allow us to examine their cross-wavelet power and relative phases.

Using XWT is then possible to compute the Wavelet Coherence.

Denoting as $W^X(a,b)$ the continuous wavelet transform of a signal X, is defined the *wavelet power spectrum* of a signal X(n) as

$$W^{XX}(a,b) = W^X(a,b)\left[W^X(a,b)\right]^* \tag{7.12}$$

where the * represent the conjugate transpose. Similarly, the *wavelet cross spectrum* of two time series X and Y is defined as

$$W^{XY}(a,b) = W^X(a,b)[W^Y(a,b)]^* \tag{7.13}$$

whose module $|W^{XY}(a,b)|$ represents the amount of joint power between the two time series, and is called *cross wavelet power*. From the cross wavelet spectrum (eq 7.13), is possible to compute the the complex argument

$$\Delta\phi(a,b) = arctan\left(\frac{Im\left[W^{XY}(a,b)\right]}{Re\left[W^{XY}(a,b)\right]}\right) \tag{7.14}$$

which represents the relative phase between X and Y, for each given value of the parameters a and b, and is defined over the interval $[-\pi,\pi]$.

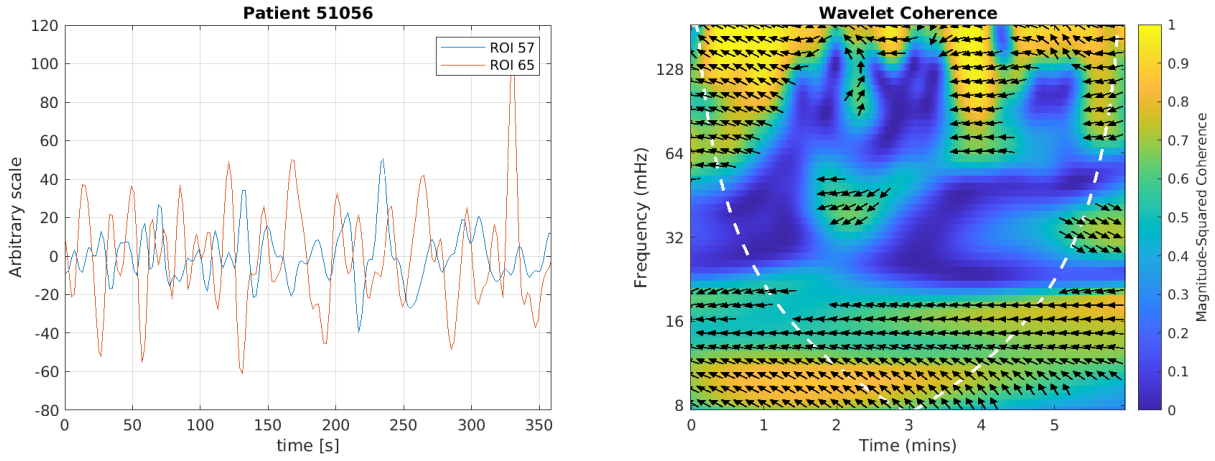
The wavelet coherence $R^2(a,b)$ is finally defined as reported in equation 7.15. This coefficient ranges in the interval (0, 1) and represents the localized correlation coefficient between X and Y in the time-frequency domain.

$$\mathbf{??}R^2(a,b) = \frac{|S(W^{XY}(a,b))|^2}{S(|W^X(a,b)|^2)S(|W^Y(a,b)|^2)} \tag{7.15}$$

In equation **??** S is a smoothing operator both in frequency and time defined as [51] [24]

$$S = S_{scale}S_{time}(W) \quad S_{time} = W \cdot c_1^{\frac{-t^2}{2a^2}} \quad S_{scale}(W) = W \cdot c_2\Pi(0.6)$$

where c_1, c_2 are normalization constants, Π is a boxcar (rectangle) function and 0.6 is an empirically determined factor for Morlet Wavelet [50].



(a) *(b)*

Figure 7.6: *Plot of two timeseries from patient 51056 in figure 7.6a and their relative wavelet Coherence scalogram. On the x axis the timepoints and on the y axis the frequencies of this time-frequency decomposition. The color represents the magnitude of the cross-power coefficients and the relative phase shift is represented as arrows pointing to the right is phase shift is zero ad to the left is is 180 degrees. Starting from 0 degrees arrows rotate counterclockwise.*

An example of wavelet coherence scalogram is shown in figure 7.6b. It represents the wavelet coherence computed from two timeseries (fig 7.6a)of patient 51056: ROI 57 corresponds to Right Angular Gyrus and ROI 65 to Right lateral occipital cortex. The color represents the magniture of the cross-power, and phase information is represented as oriented arrows, pointing towards an imaginary 360 degrees circle, where the zero phase shift is represented by an arrow pointing right and a 180 degrees, by a left arrow.

Our goal is to extract from this wavelet coherence matrix, a single value, interpretable as a correlation coefficient, just like we did with Pearson coefficients. We can accomplish so, by extracting two types of information from this scalogram: the magnitude of the correlation and relative phase between the two signal.

We can extract the magnitude information of each entry of the scalogram, but before doing so, we need to assess the level of significance of this coherence matrix over the noise, this way we can estimate the statistical significance level of our values, and only collect the significative ones.

The theoretical procedure [24] [4] is to generate, using Monte Carlo methods, a large (1000) samples of wavelet coherence matrices using red noise timeseries. [2] These red noise samples should be generated, for each time series, with the same 1-lag autoregressive coefficients (AR1 coefficient) of the two timeseries under examination. Then, for each pair of red-noise timeseries, we should compute wavelet coherence matrix. However, since AR1 coefficients have little impact on the significance level [24], we choose to generate a single sample of 1000 pairs of red noise and for each pair we computed the wavelet coherence matrix.

For each pair of noise, the corresponding wavelet coherence matrix from equation 7.14 is stacked to obtain a $A \times B \times x \times 1000$ 3D noise matrix to extract the distribution of the entries. This null distribution of wavelet coherence coefficient is used to estimate the threshold to 5% significance level for the subsequent analysis. We refer to the value corresponding to this 95-th percentile as a_{95}.

As a second step we extract the relative phase information of each entry of the matrix, and combining these two informations together, we can estimate the time of in-phase (or out of phase) coherence which can be seen as the percentage of time synchronicity (or antisynchronicity) between the two timeseries. [4] This time of in-phase coefficient is defined as:

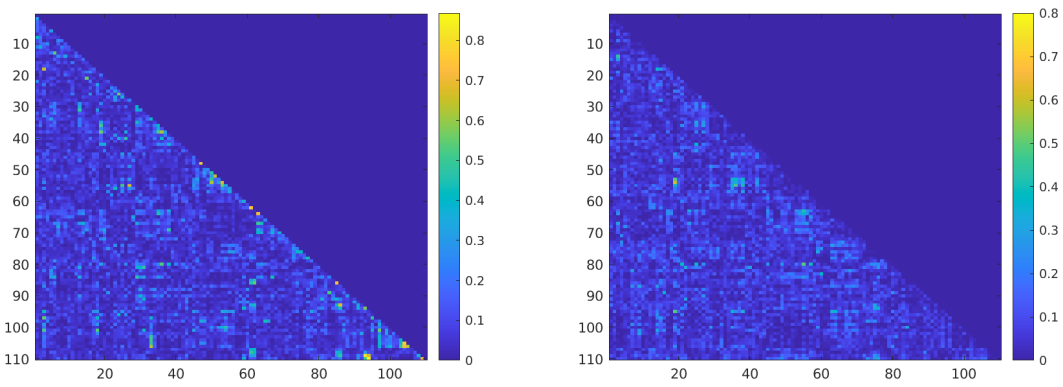
$$c_{ij} = \frac{100}{N} \sum_{a,b}^{N} I\left\{R^2_{ij}(a,b) > a_{95}\right\} \cdot I\left\{-\frac{\pi}{4} < arg(W^{XY}(a,b))_{ij} < \frac{\pi}{4}\right\} \tag{7.16}$$

Where the indices i and j refers to the two timeseries i and j; N is the total number of points inside the cone of influence (COI) $I\{...\}$ is either 0 or 1 depending on if the condition inside is satisfied; a_{95} is the threshold value above which the computed wavelet coherence coefficient is regarded as significative.

Similarily to this coefficient is the time of counter-phase coefficient, which can be obtained by modifying from the formula 7.16 the phase condition into

$$I\left\{arg(W^{XY}(a,b))_{ij} < -\frac{3\pi}{4} \ \vee \ arg(W^{XY}(a,b))_{ij} > \frac{3\pi}{4}\right\}$$

In short with this analysis, we are just considering coefficients with an high significance level (above 95%) and with a small $\in [-\pi/4, \pi/4]$, or big phase shift ($< -\frac{3\pi}{4} \vee > \frac{3\pi}{4}$).



(a) *Coefficients of In-phase percentage* ***(b)*** *Coefficients of Off-phase percentage*

Figure 7.7: *Correlation matrix created with wavelet coefficients of in-phase and off-phase percentage. Patient 51056, ABIDE I*

Wavelet coherence maps were calculated using MATLAB's wavelet Toolbox, which employs the Morlet wavelet, for the wavelet decomposition, as mentioned before, and decomposes the frequency range using 12 subscales per octave and 9 octave.

From equation 7.16 both coefficients in-phase and anti-phase matrix coefficients were computed. The correlation coefficients matrix for each subject, were then flattened and used as input to the neural networks. An example of correlation matrix created with in-phase and out-of-phase coefficients from data of patient 51056 is shown in figure 7.7

Chapter 8

Multicenter data harmonization

Since nw, a big percentage of neuroimaging studies have been carried out with limited data acquired from a single center, to minimize varibility in data due to the different instrumentations employed. In recent years, however, there's the tendency to create shared datasets, by pooling together data acquired from different centers. A positive side of creating a large dataset putting together data from multiple centers is the possibility to work on dataset of bigger dimension, resulting in obtaining more statistically accurate analysis, and facilitates the generalization and the robustness of the model. As a drowback, however, it introduces variability in our analysis such as differences due to scanner models, acquisition parameters, generally known as scanner effects. A level out procedure becomes necessary to avoid the model to learn these differences deriving from inomogenity of data because of differences in data acquisition procedures. This procedure is generally referred as harmonization. Harmonization was tested on different datasets and has proven to be an effective way to reduce scanner effects in different kind of datasets such as genes' microarray data, diffusion tensor imaging or structural MRI. [29] [21] [31].

8.1 Harmonization - theory

The state-of-art procedure used in genomic is called ComBat (named after Combating Batch effects), used for structural MRI but aplicable to any kind of imaging data, is used to mitigate scanner effects. It is based on a previous method initially proposed in 2007, for gene expression studies to compute

batch effect corrections, later implemented by Fortin et al [20] for the harmonization of cortical MRI volumes, and finally in its current improved version, developed by Pomponio et al. [39] in 2019. In this paragraph we are going to explain how ComBat technique works and after that, we'll see how it was modified and improved in the implementation made by Pomponio et al.

ComBat technique belongs to the family of Location and Scale adjustment methods, but it represents an improved version of them: since it models site-specific scaling factors and uses empirical Bayes estimate to improve the estimation of the site parameters for small-sized datasets.[20] It aims to reduce inter-site variability while preserving biological variability such as differences due to sex, age, FIQ (Full intellective quotient), ICV (Intra Cranical Volume) and so on. The model assumes that a feature can be modeled as a linear combination of the biological variables plus the site effect, and the errors introduced with site effect can be modeled as both a multiplicative and an additive term and that it can be standardized by adjusting the mean and the variance across the batches.

Let y_{ijf} be the numeric value of the feature f for the patient i acquired with the scan (or equivalently, from the site) j, so that i = 1, 2, ..., K indexes the scanner, j = 1, 2, ..., N_i indexes the total number of subjects acquired for each scanner i, and f ranges from 1 to F being F the total number of features. Besides, let's assume that this value y, can be written as

$$y_{ijf} = \alpha_f + x_{ij}^T \beta_f + \gamma_{if} + \delta_{if} \epsilon_{ijf} \tag{8.1}$$

where:

- α_f is the mean value for the feature f,
- x_{ij} is the entry of the matrix X created with the covariates of interest such as age, sex,
- β_f is the vector of regression coefficients corresponding to X for the feature f,
- γ_{if} and δ_{if} represent the additive and multiplicative terms for site-i effects related to feature f respectively,
- ϵ_{ijf} is a error term which is assumed to follow a normal distribution with zero mean and variance σ_f^2.

The final location-and-scale-adjusted data y^*_{ijf} are given by

$$y^*_{ijf} = \frac{y_{ijf} - \hat{\alpha}_f - X \cdot \hat{\beta}_f - \hat{\gamma}_{jf}}{\hat{\delta}_{jf}} + \hat{\alpha}_f + X \cdot \hat{\beta}_{jf} \tag{8.2}$$

where $\hat{\alpha}_f, \hat{\beta}_f, \hat{\gamma}_f, \hat{\delta}_{jf}$ are estimators of the corresponding parameters, based on the model.

One of the most disadvantages of using a simple location and scale batch adjustment is that it requires a large batch size for the implementation because is not robust to outliers in small sample sizes.

ComBat uses empirical Bayes to provide a more robust adjustment for the parameters $\hat{\gamma}_{if}$ $\hat{\delta}^2_{if}$, making the model able to deal with small-sized dataset as well.

The process ComBat employs to estimate feature-dependent parameters, and to adjust data for batch effect can be summarized in 3 steps, at the end oh which we reach obtain the results shown in equation 8.2

1. **Standardization**: Data are standardized feature-wise, so that every feature has similar overall mean and variance. least square regression, is then performed to determine parameters $\hat{\alpha}_f, \hat{\beta}_f, \hat{\gamma}_{jf}$ and subsequently, $\hat{\sigma}^2_f = \frac{1}{n} \sum_{ji} \left(y_{ijf} - \hat{\alpha}_f - X\hat{\beta}_f - \hat{\gamma}_j f \right)^2$ being n the total number of patients.

   The standardized data are then calculated by equation 8.3

$$Z_{ijf} = \frac{y_{ijf} - \hat{\alpha}_f - X\hat{\beta}_f}{\hat{\sigma}_f} \tag{8.3}$$

2. **Empirical Batch parameters estimate**: We assume that standardized data follow a normal distribution $Z_{ijf} \sim N(\gamma_{jf}, \delta^2_{jf})$ and we seek for a proper estimation of parameters $\gamma_{jf}, \delta^2_{jf}$. it is also assumed that these site-effect parameters follow the normal distribution and the inverse gamma distribution respectively

$$\gamma_{jf} \sim N(\eta_j, \tau^2_j) \qquad \delta^2_{jf} \sim \text{Inverse Gamma}(\lambda_j, \theta_j) = \frac{\theta^\lambda}{\Gamma(\lambda)} (1/x)^{\lambda+1} \cdot e^{-\theta/x}$$

   And these hyperparameters $\eta_j, \tau^2_j, \lambda_j, \theta_j$ are empirically estimated from standardized data Z_{ijf} by using the method of moments. we then obtain improved estimation of parameters γ^*_{jf} and δ^*_{jf} and we use them for the third and last step, where we adjust our data using them.

3. **Adjust the data**: After calculated the site-effect parameters we are finaly able to adjust our initial data using the relation

$$y_{ijf}^{ComBat} = \frac{\hat{\sigma}_f}{\delta_{jf}^*}(Z_{ijf} - \gamma_{jf}^*) + \hat{\alpha}_f + X\hat{\beta}_f \tag{8.4}$$

which is just an equivalent way to write equation 8.2, using parameters estimated in the previous passage.

So far, this was the main idea behind ComBat technique, but we said before that the last implementation of this technique by Pomponio et al. [39] is an improved version of it. They differn in the modelling of the biologically covariates, which in the formulas above are expressed by the terms $\hat{\alpha}_f + X\hat{\beta}_f$ which is a simple linear model. Pomponio substituted this linear model with a Generalized Additive Model (GAM). In this model covariates such as sex, age, FIQ, are represented by terms $x_{ij},\ z_{ij},\ w_{ij}$ which allow a better parametric modelling to deal with non-linear trends such as the trend of cortical thickness in relation to age. This way, the terms $\hat{\alpha}_f + X\hat{\beta}_f$ in equation 8.2 are substituted by a linear combination of function F of these covariates

$$\hat{\alpha}_f + X\hat{\beta}_f \ \longrightarrow\ F_f(x_{ij}, z_{ij}, w_{if}...) = a_f + g_f(x_{ij}) + h_f(z_{ij}) + p_f(w_{ij}) + ... \tag{8.5}$$

Where functions g_f, h_f, p_f can be either linear or non-linear functions of our covariates, according to how we want to model these covariates, or in other words, according to wether we want to specify some kind of non-linear relationship between covariates and our data.

Chapter 9

Domain-adversarial NN

In this section we discuss a different and innovative approach to deal with multi-center datasets. So far we presented one important harmonization procedure that removes site-related features in an analytical way; an other possible approach is to make use of a deep neural network specially designed to perform a classification unbiased by site-related features. The construction of this network gets its main idea on the netwrok proposed by Ganin, Ustinova et al, in their article [23]. This work adresses the issue of working with two different datasets. They call them source and target dataset, and each one contains data following different distributions. Moreover, source is labeled while target is unlabeled. Data are in a total amount of N, divided into n samples in source and N-n samples in target. Their goal is to create a network able to learn relevant features for a classification task from the source dataset (since is the only one labeled), and learn as well some distinctive feature related to the two domains. This result is achivable by constructing a model able to generalize well from one domain to another.

We illustrate the main aspects of this problem using a shallow neural network with a single hidden layer consisting of D nodes. We suppose the network takes as input an m-dimensional features vector. The hidden layer can be represented as a function $G_f : \Re^m \to \Re^D$ with a weight parameters matrix W and bias vector b which for brevity's sake we can denote as θ_f. Given an input vector $x \in \Re^m$ it acts like

$$G_f(x; \mathbf{W}, \mathbf{b}) = f(W \cdot x + b) \tag{9.1}$$

Where f is some activation function that we can represent as a sigmoid function.

Similarly the output will be a function $G_y : R^D \to \Re^Y$ where Y is the total number of classes of our data. This layer can be written as

$$G_y(G_f(x; W, b); V, c) = f'(V \cdot G_f(x) + c) \tag{9.2}$$

Here, too we can denote parameters V and c with a single notation θ_y. A usual train carried on only on the (labeled) Source domain, will therefore bring to the minimization of the loss function associated with the output, dependent on parameters θ_f, θ_y

$$\min_{\theta_f, \theta_y}[\frac{1}{n}\sum_{i=1}^{n} L_y^i(Gy(G_f(x_i)), y_i)] \tag{9.3}$$

.

Training procedure and minimization of the loss function can be performed only on the source dataset consisting on n samples, since target data are unlabeled. At this point, to tackle the problem of domain independence their idea is to consider the hidden layer as an internal representation of data, and use its information to create a domain regressor.

A domain regressor can be implemented as a layer G_d completely similar to the output layer G_y, depending on parameters V, c which we will denote as θ_d. It takes as input the hidden layer G_f, and after being activated by a sigmoid function, returns an output. We indicate the loss function associated to this output as L_d, and the loss previously introducted for label classification as L_y.

The complete optimization function can be now written as

$$E(\theta_f, \theta_y, \theta_d) = \frac{1}{n}\sum_{i=1}^{n} L_y^i - \lambda(\frac{1}{n}\sum_{i=1}^{n} L_d^i + \frac{1}{N-n}\sum_{i=n+1}^{N} L_d^i) \tag{9.4}$$

Following this strategy, they perform a minimization with respect some parameters and a maximization with respect to others. More precisely they seek for a saddle point given by

$$\begin{aligned} \hat{\theta}_f, \hat{\theta}_y &= \underset{\theta_f, \theta_y}{argmin} E(\theta_f, \theta_y, \theta_d) \\ \hat{\theta}_d &= \underset{\theta_d}{argmax} E(\theta_f, \theta_y, \theta_d) \end{aligned} \tag{9.5}$$

This task was accomplished by embedding the domain regressor G_d into the neural network made by G_f and G_y. The resulting neural network will consist on two branches: one for label

classification and the other with the domain regressor. They linked these parts using a *gradient reversal layer* and exploited a classic stochastic gradient descent procedure to update weights. A gradient reversal layer allow the training of the network in an adversarial way. It is placed at the top of the domain regressor branch, as we can see from the presentation image (fig 9.1) directly taken from their article. Let us point out that so far, we discussed the structure of this network by using a shallow neural network consisting just on a single hidden layer, but this architecture can be generalised by adding additional layers for each branch of the network, as shown in figure 9.1

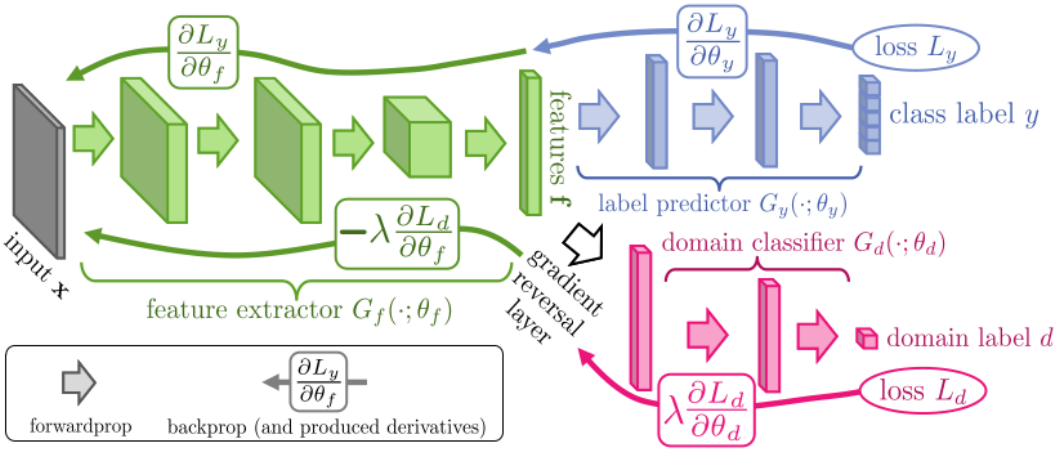


Figure 9.1: *A schematic representation of a Domain Adversarial Neural Network, from the presentation article [23]. The network mainly consists of three parts: a feature extractor, which takes inputs and creates an internal representation of data. From it the network is splitted into two branches: a label classificator G_y with associated loss L_y and a domain regressor/classifier denoted as G_d with associated loss L_d. This latter branch is linked to the feature extractor branch by making use of the gradient reversal layer.*

During the backpropagation apt to parameters update, minimizing parameters are updated in the opposite direction of the gradient as a normal parameter fine tuning explained in section 3.5 while maximizing parameters are modified going in the direction of gradient.

$$\begin{aligned}
\theta_f &= \theta_f - \lambda \frac{\partial L_y}{\partial \theta_f} \\
\theta_y &= \theta_y - \lambda \frac{\partial L_y}{\partial \theta_y} \\
\theta_d &= \theta_d + \lambda \frac{\partial L_d}{\partial \theta_d}
\end{aligned} \tag{9.6}$$

During training, classificaiton branch and domain regressor compete against each other in an adversarial way for the optimization of the parameters. For this reason this network is called Domain-Adversarial Nerual Network (DANN).

Chapter 10

SHAP

Figure 10.1

SHAP (SHapley Additive exPlanation) is a technique based on game theory that provides a method for machine learning model explanability. It is a powerful tool to get rid of the "black box "idea of a machine learning model and try to understand its deep mechanism and the reasons why a model gives a certain output, or a certain prediction related to an input sample such as a vector of feature or an image for example. As is shown in figure 10.1 just as a visual example took from the SHAP documentation [1], a common machine learning model acts like a black box that returns an output value after some non-linear unknown calculations over some input values; with an explanatory model, we are able to quantify the contribution of each feature of our input data and assess what and why are the most important features and how much they contributed to the final outcome.

[1] https://shap.readthedocs.io/en/latest/

The idea behind SHAP is to use Shapley values (see chapter 4) to exaplain every single feature's contribution in our machine learning model, treating the learning process as a cooperative game. To apply the concept of Shapley values to a machine learning model we just adapt some terms we used for game theory: the payout of a coalition becomes now the model's prediction and the players are the feature in our input data. For a single feature, its Shapley value is defined as the average marginal contribution of that feature across all possible coalitions.

There are two main classes of explainatory algorithms called local and global methods

Shap can be regarded as both a local and global explainatory method, and before it, several algorithms belonging to these categories were implemented as an attempt to create an explainatory technique for machine learning models. In general, the objective of a local explainatory model can be defined as the attempt to explain an output f(x) after a single instance x that in our case can be identified as a vector of features. Local methods differ from global methods, because the latter provide a global explanation of the model, across all the instances, they are able to attribute an importance to each feature to determine which one contributes the most to the output of the model.

One of the key point of SHAP is its flexibility, being both a local and global explainatory model, this way we are able to explain a single instance as well as an entire dataset and extract the most important features.

Being a local model, SHAP share with other algorithms some common implementation traits: given a single input vector x, to simplify the workflow, they introduce a coalition vector x' = $\{0,1\}^F$: where F is the total number of features: x' is a binary vector of the same length of x, where binary means made just by zeros and ones as its entries: zero means that in our analysis we are going to withheld the corrisponding variable from x, and the one that we are including it.

To map the x' vector to the corresponding vector containing the actual value of the features, a mapping function $h_x(x') = x$ is introduced to returns the actual features vector x, unique for each vector x. The goal of a local explainatory technique is to find a simplified explainatory model g(x') to work with, because is simpler to work with a binary vector x' and a simlified model than with the actual vector x and the model itself. We want that this simplified model, when applied on the vector x', is able to approximate our output f(x): $g(z') \approx f(h_x(z'))$ if z' $\approx$ x'. The first important

condition on this function g is that it has to be a linear function

$$g(z') = \phi_0 + \sum_{i=0}^{F} \phi_i z'_i \tag{10.1}$$

And several methods before SHAP were created that satisfy this condition such as LIME [43] or DeepLIFT [47] but they lacked additional properties that SHAP has:

1. **Local accuracy**:

$$g(x') = \phi_0 + \sum_{i}^{F} \phi_i x'_i = f(x) \quad \text{when} \quad x = h_x(x') \tag{10.2}$$

   and not just an approximation $f(x) \approx g(x')$. Here coefficients ϕ_i represent the impact of the associated feature to the model's output, and ϕ_0 represents the coefficient corresponding to a vector x' with all entries equal to zeros.

2. **Missingness**:

$$\text{if } x'_i = 0 \implies \text{ then } \phi_i = 0 \tag{10.3}$$

   Practically if a feature is equal to zero, it has no attributed impact on the model outcome

3. **Consistency**: if we have two different models g' and g. Denoting with z'/i the setting where $z'_i = 0$

$$\text{if } g'(z') - g'(z'/i) \geq g(z') - g(z'/i) \forall z' \in \{0,1\}^F \implies \text{ then } \phi_i(f', x) \geq \phi_i(f, x) \tag{10.4}$$

   This property states that if the contribution of a feature z'_i increases, regardless all the other features in a model, its associated importance value increases or at least remain the same, but does not decrease.

In 1975 it was demostrated [56] that the only coefficients satisfying all of the above properties are Shapley values, and as a consequence methods which employ different coefficients violate one of these properties, usually local accuracy and/or consistency. For this reason SHAP employs Shapley values and takes advantage of pre-existing algorithms (some of which were cited before: LIME and DeepLIFT) that used different coefficients. It readapts and enhances them providing an unified approach to assess feature importance for different models without any violation of the axioms above.

To compute the Shapley value for a feature we have to evaluate the model on all the possible subsets S we can create with our data S$\subseteq F \setminus \{i\}$. This approach, in a problem with an high number of features would result in a huge computational work. To bypass this problem, we can choose not to calculate the exact shapley value but just an approximation of it, depending on the model we are working on. Some of them such as Tree SHAP, are able to compute the exact shap coefficients and is developed to obtain fast performances on trees or ensambles of trees. Others just compute an aproximate shap value. Algorithms we are going to focus on are KernelSHAP which is a model agnostic explanation method, and DeepSHAP which is optimized to work faster on deep models by making use of the knowlegde of the structure of a neural network. Each one of these algorithms is built upon previous algorithms like LIME or DeepLIFT. In the following section we are going to explain some important feature from LIME that were employed in KernelSHAP, and from DeepLIFT employed for DeepSHAP.

10.1 LIME and KernelSHAP

LIME (Local Interpretable Model-agnostic Explanations) is a local, model-agnostic, interpretability model, made to explain the prediction of any classifier in a faithful, but only local way. This means that it can accurately explain a single prediction but it is not the most suitable to generalize to many of them [43].

Since Kernel SHAP is built upon LIME algorithm, we briefly discuss how LIME works to obtain ... it pick up an important strategy from it, to avoid the computation of all the permutations.

Starting from a single instance $x \in \Re^n$ LIME creates a perturbed dataset X and evaluate the model over this dataset. Perturbed dataset is created by making use of several binary vectors containing just zeros and ones. In fact, it randomly creates different binary vectors of the same lenght of x, but randomly set 0 and 1 as its entries. From this binary dataset it is possible to come back to the space of features value, recalling that "1 "means that we are including the corresponding feature from x, while "0 "is associated to the missingness of the corresponding feature. LIME replaces missing features in different ways according to the type of data it is dealing (Images, text

data, tabular data..). In the case of tabular data, LIME repleaces missing values by randomly sampling a value from the distribution of that feature from the training data [18]. KernelSHAP

KernelSHAP works in a similar way, but introduces a background dataset from which missing values are randomly picked and replaced in the perturbed vector.

From this point LIME works by using the synthetic dataset just created, and the model is evaluated on each vector. These outputs are subsequently used to accomplish a minimization task to finally find the importance coefficients. Minimization is performed in a regression, but before it, each output value is weighted according to the distance of the synthetic vector with the original vector x by using coefficients π_x defined as

$$\pi_x = \frac{p-1}{\binom{p}{|z'|} \cdot |z'|(p-|z'|)} \tag{10.5}$$

Where we indicated as p the total number of features in the original feature vector x anc with $|z'|$ the number of element in that subset, and cosequently $(p-|z'|)$ is the number of features not included in the subset.

To perform a fit, the loss function to minimize is in the form $L(f,g,\pi_x) = \sum_{z\in Z}[f(h(z)) - g(z')]^2\ \pi_x$ where g(z') is the simplified function expressed in equation 10.1, given by a combination of coefficients ϕ_i that are the parameters of the fit.

LIME is regarded as local method because it explains the prediction of a black box model by making use of a local model: in this case a regression, which is an interpretable model computed in the neighborhood of the instance we want to explain. KernelSHAP implementation was strongly influenced by the LIME algorithm but,thanks to the introduction of Shapley values, it also comes with theoretical guarantees about consistency and local accuracy, from game theory.

As mentioned above, KernelSHAP is a model-agnostic algorithm, in the sense that can easily used with every kind of model, but SHAP includes some model-specifics algorithms that make use of a previous general knowledge of a model's structure to optimize the explainer's performances and speed up the process. DeepSHAP, is one of these model-specific algorithms and it is what we are going to use in our analysis.

10.2 DeepLIFT and DeepSHAP

DeepSHAP alghorithm works with some similar implementation of KernelSHAP, making use of a background dataset to simulate missing values, but it is also optimized to perform better on deep models by taking advantage of the idea behind the DeepLIFT algorithm. It can be considered as an enhanced version of DeepLIFT, thanks to the introduction of the shapley values, rather than the "DeepLIFT multipliers "employed in DeepLIFT

The strategy implemented by DeepLIFT to explain features, is to compute the difference between the output of a deep learning model with our "true "data and a reference value computed as the output of our model with some reference data. The choice of the reference data depends on what kind of data we are working on: images, or genomic data for example. For genomic the most common way to produce reference output in DeepLIFT is to shuffle some training data, evaluate the model on them and average across all the scores. With the implementation of DeepSHAP, this reference value is chosen from the backgound dataset and represents an uninformative value for a feature.

As we mentioned before is also shaped to perform on deep learning algorithm computing its feature importance values called DeepLIFT multiplier during the process of backpropagation. DeepLIFT attributes to each feature x_i a value $C_{\Delta x_i,\Delta y}$ that represent the effect of that input being set to a reference value rather than its actual value.

In a few words the idea behind DeepLIFT (and DeepSHAP) can be summarised as: let t represent the output of some inner neurons and let x1, x2, ... , x_n be some preceding neurons necessary to compute t, if we label t_0 the reference output, we can compute the value $\Delta t = t - t_0$ and use them to define the DeepLIFT values $C_{\Delta x_i \Delta t}$ associated with Δx_i which is the difference between the true feature and the reference value for that feature. DeepLIFT coefficients are choose to follow the property called *summation to delta* property

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t$$

For a given input neuron x with difference from reference Δx and a target neuron t, is defined

the *multiplier*

$$m_{\Delta x \Delta t} = \frac{C_{\Delta x \Delta t}}{\Delta x}$$

which represents the contribution of Δx to Δt. Once computed the multipliers of each neuron in a layer it is possible to compute the multiplier of any target neuron during the backpropagation. For a more detailed explanation of how DeepLIFT works we refer to its presentation article [47]

DeepSHAP exploits the same principles as DeepLIFT, and combines shap values for smaller components of a network to compute values for the whole network. It accomplish this by recursively passing Deep LIFT multipliers now defined in terms of SHAP values during backpropagation. Since notation can become quite annoying, we explain the main idea of how this procedure works. Suppose we have a single input vector x consisting only on two features: $x = (x_1, x_2)$ and a single background vector $b = (b_{x_1}, b_{x_2})$ we follow the notation on article [10] but we simplify for a better understanding. We denote as f_{x_i} the actual value of the feature x_i and as b_{x_i} the value of the entry from the backgound vector corresponding to x_i, and we also denote as h_i the output of a neuron with the input is x, and with b_{h_i} the output of the neuron when the input is b. Giving a look at figure 10.2 we can write formulas for forward propagation.

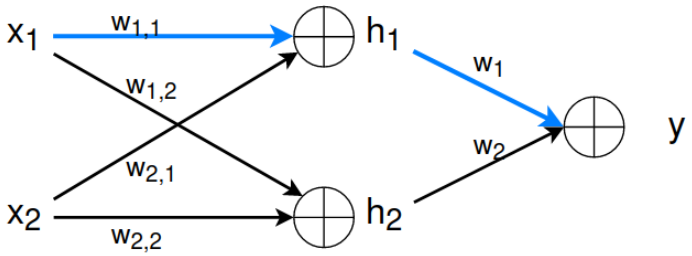


Figure 10.2: *Representation of a linear network and important quantities involved during forward propagation*

$$h_1 = w_{1,1}x_1 + w_{2,1}x_2 \qquad y = w_1 h_1 + w_2 h_2$$

Giving a look at the model, we can retrieve Shap values for each node by summation of all the contribution from each path, where the path contribution to a shapley value for x_i is defined as the

product of the weights along the path and the difference between x_i and the background value b_{x_i}. In practice, if we focus on the colored blue line we have that the contribution from this path to $\phi(h_1)$ is given by

$$\phi(h_1) = w_1(f_{h_1} - b_{h_1})$$

with this information, we can finally compute the contribution to the Shapley value of x_1 coming from this path

$$\phi(x_1) = (f_{x_1} - b_{x_1})w_{1,1}w_1 = (f_{x_1} - b_{x_1})w_{1,1}\frac{\phi(h_1)}{f_{h_1} - b_{h_1}} \tag{10.6}$$

adding all contribution from all the possible paths (just two in the figure above, but much more in a real neural network), it is possible to compute the final Shapley value for a feature x_i in terms of the attributions of all the intermediary nodes of the network.

10.3 Shap values as feature importance

To express the significance of a feature, we can rely on the relation between feature importance in a qualitative way and shap value: feature with high absolute shapley value are important. The absolute importance value for a feature f is calculated as the mean of the magnitude of all the shapley value for that feature, so the mean is performed across all the n samples we used to calculate these values

$$I_f = \frac{1}{n}\sum_{i=1}^{n} \|\phi_i^{(f)}\| \tag{10.7}$$

Shap is implemented as a free Python package with MIT license, developed and mantained by Scott Lundberg [2]. This package includes different classes such as KernelExplainer, which is the class name of KernelSHAP mentioned in section 10.1, TreeExplainer tuned to perform rapidly on tree and forest-like models, linearExplainer which deals with linear model and is able to compute the exact shapley values and not just an aproximation, and the class we are using: DeepExplainer, suitable to explain deep learning models and to compute an approximate value of shapley values.

[2] https://github.com/slundberg/shap

Chapter 11

Analysis workflow

So far we have been discussing what type of data we used in our work without specifying what kind of analysis we carried out with those data. We mainly performed classification using two different machine learning classifier: a deep neural network and a common Random Forest classifier. Before running any classification procedure we tackled the problem of data harmonization. To this end in chapter 12 we explain how we implemented the procedure of data harmonization and some results obtained by applying it on our data obtained from the analysis discussed in chapter 7 which we report for convenience:

- Data obtained by Fisher-transforming Pearson correlation coefficients extracted for each pair of timeseries of each patient
- Data obtained from wavelet analysis by computing time of in-phase or out-of-phase percentage coefficients for each pair of timeseries of each patient

We report results to have a visual feedback of how harmonization modify data and how effective is this procedure in eliminating site-related information from our data.

In chapter 14 we present results obtained from classification of data using different machine learning algorithms. We seek to find the best strategy to obtain the best separability between Controls and ASD data. Classification was carried out following different pipelines: we compared performances between a Random Forest classifier, a deep neural network and a Domain-adversarial

neural network. We also discuss the effects of data harmonization on a classification procedure and we discuss the best strategy to implement harmonization.

Analysis pipelines

For each classification analysis, we implemented a K-fold cross validation scheme to report our results. We determined and reported model performances as the mean AUC value and the standard deviation of the scores for each k-fold Since we are not working with a huge amount of data (< 1600 samples), we choose to perform a 5-fold CV. We made this choice because with a 10 k-fold CV, the error on the mean AUC score increases, since we have less data in the test dataset, and hence more variability. For this reason we choose to hold a greater number of data in the test (at the cost of reducing the number of data available for train) to reduce variability among them and reduce the standard deviation of the model's score. K-fold CV was implemented using the stratifiedKFold class provided by sklearn library for Python.

A flowchart of the analysis mentioned above is shown in figure 11.1 In details the four types of analysis we carried out are:

1. Classification of ASD/controls using raw (not harmonized) data

2. Classification using the whole harmonized dataset, creating the harmonization model on all the control subjects and applying the model on all ASD subjects, and then split this dataset into a train and test subsets for each k-fold. The flowchart to schematise this implementation is shown in figure 11.3.

3. Classification with harmonization implemented inside the K-fold cross-validation, creating the model on controls and applying it to ASD: following this procedure data were splitted into a train and a test dataset, therefore using train data we computed the harmonization model only on control patients and then apply the model to ASD patient and both Control and ASD test data. The whole flowchart is shown in figure 11.2

4. Classification using the domain adversarial neural network.

Harmonizing data inside a k-fold is the best way to keep train and test data apart and avoid

data leakage between train and test. It is a common source of error to introduce some kind of data leakage between the train and the test dataset, when we harmonize on the entire dataset, the harmonization model permorms a fit using all the data, so each data is modified according to information obtained from all the other data, so in data belonging to the train dataset there's already information about the test dataset, and this leads to a misleading increase of model's performances. Some articles perform a Leave One Site Out Cross validation. In our case we can't do that if we want to mantain train and test independent because if we compute the harmonization model only on the train dataset, we wouldn't have information to apply it to the test dataset since its data belong to a site not included in the model.

Dataset selection

With reference to the "Patients attributes selection "in figure 11.1 we discuss now what selection criterias we adopted. All the analysis mentioned above were performed using different selection criterias and thresholds on the dataset which brought us to different datasets combination. First of all, we excluded from the analysis data from sites NYU_2 and KUL_3 because as noticed in chapter 5 they did not provide control data but only ASD patients, and without control data, we can't perform harmonization on these sites because to create a harmonization model, we perform a regression on control data and then apply the model to ASD data of that site. The first and most obvious selection criteria is to use the whole dataset from ABIDE 1 and 2, consisting of only male patients aged between 5 and 40 years old, without any further constraint. So we have our first complete dataset consisting on 1470 patient data, from ABIDE I + ABIDE II datasets. This dataset can be divided to analyze classification performances separately on ABIDE I and ABIDE II, mantaining the same constraints on ages and sex. This choice was made because several articles dealing with controls/ASD classification using the ABIDE dataset, are fullfilled only on ABIDE I, or, more precisely, on ABIDE I preprocessed (see section 6.2). Thus for an immediate comparison we choose to run classification on the two disjoint dataset as well.

To seek for a more homogeneous dataset, an other constraint can be set on the eye status at scan. The objective is to select only patients who kept open eyes throughout the entire scan session. This, as mentioned in chapter 1.3 helps obtaining a more homogeneous dataset, removing also data

potentially altered by sleep. This constraint was applied on the dataset consisting on data from ABIDE I and ABIDE II pooled together cited earlier, and on ABIDE I and ABIDE II separately as well. In short, considering these constraints, we obtain 6 different subdatasets, with different number of patients each, summarised in table 11.1

Dataset	Constraints	Tot	Controls	ASD
AB I+II	eye = all, sex = M, age = (5, 40)	1470	737	733
AB I	eye = all	841	426	415
AB II	eye = all	629	311	318
AB I + II	eye = open	1026	514	512
AB I	eye=open	568	281	287
AB II	eye = open	458	233	225

Table 11.1: *Total number of data and control/ASD amount for each subset and thresholds. To avoid repetitions in all the subdatasets, are implicit constraints regarding sex (only males) and ages (between 5 and 40).*

Dimensionality reduction of data

An important issue related to data we are working on concerns their dimensionality. We are dealing with data lying in an high dimensional space but we lack an apropriate number of data to make acccurate predictions on them: this problem is usually referred as the "curse of dimensionality "In our work, dimensionality is a relevant issue since we are dealing with 5995 features and less then 1500 patients, and for this reason we run our analysis in a situation of permantent overfitting. One important method to tackle this problem is dimensionality reduction performing a Principal Components Analysis (PCA) as explained in section **??**

When applying PCA to a dataset we seek to explain as much variance as possible, for this reason we start our pca analysis extracting a number of principal components that explain more then 90% of variance, then we gradually reduced this number halving if. The maximum number of principal components is limited by the number of data we can use. If we have a dataset of $n\_samples$ samples, each with $n\_features$ features, the maximum number of PC is given by $N\_pc = min\{n\_samples, n\_features\}$. This is because in a $n - feature$-dimensional space, our points lie

in a $n\_samples$-dimensional hyperplane and all the variance can be explained inside this subspace. For a better understanding of this concept, we can imagine a dataset made of only two data, each containing 3 features. If we represent these data, we construct a 3D space where each orthogonal axes corresponds to a feature, and we insert these two points, each corresponding to a data point. If we want to explain the maximum variance we need to find a new axis, the first principal component, but through two points passes just one straight line so we can at most limit our analysis to the plane passing through that line.

In our case $n\_samples$ is always $<$ $n\_features$, so the number of principal components will be limited by the number of data in the train dataset. We use just the train dataset to calculate principal components because this is the correct way to proceed to avoid data leakage between train and test dataset.

To avoid repeating all the analysis mentioned earlier with the additional implementation of PCA, we choose to implement it only on some sub-dataset and using those coefficients that give the best classification performances.

From the initial input data consisting of 5995 features, we performed classification using different values of principal components. The search for principal components is accomplished only on the train dataset and then the same transformation is applied on the test dataset. In accordance of what we stated about harmonization when we asked whether is more correct to perform it inside or outside the k-fold splitting of train/test, calculating principal components just on the training set, allows to keep train and test set apart, and avoid data leakage of test into train.

Results are shown in chapter 14.3

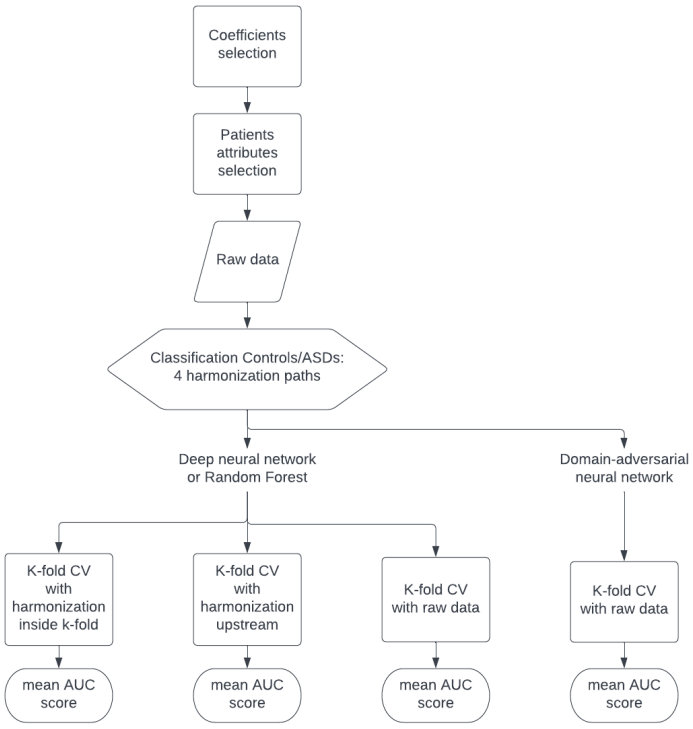


Figure 11.1: *Flowchart of the analysis carried out in this work:*

1. *Coefficient selection: select whether to use Pearson-based correlation coefficients, or wavelet-based correlation coefficients*

2. *Patients attributes selection: select the desired attributes of patients in order to make cuts on dataset, for example use the whole ABIDE I + II dataset or just ABIDE I, limit analysis to a certain age range, sex, or eye status open or close.*

3. *After making cuts, and obtained raw data, classification can be performed. Classification of Controls/ASD can be fullfilled by a deep neural network, a Random Forest classifier, or the Domain-Adversarial neural network.*

4. *Different harmonization procedures are compared: 1) harmonization implemented inside the k-fold CV, 2)harmonization of the entire dataset before k-fold CV, 3) no harmonization 4) harmonization implemented with the Domain adversarial neural network.*

5. *For each of procedures, the final classification score is reported as the mean and standard deviation across all the folds of k-fold CV.*

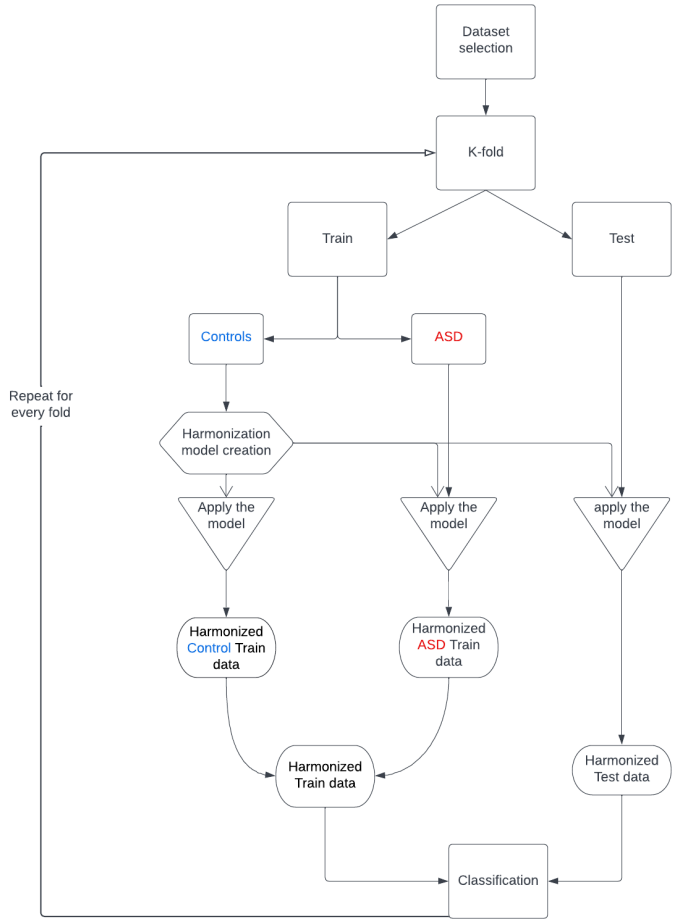


Figure 11.2: *Flowchart for the implementation of harmonization inside a k-fold CV: for each fold, data are splitted into a train and a test dataset. Harmonization model is created on control data of the train dataset, and applied to ASD data and on the entire test dataset. Harmonized control and ASD train data are merged to obtain a harmonized train dataset. Classification is performed on the new harmonized dataset: training is accomplished on the train dataset and the model is evaluated on test dataset. This entire process is repeated for each fold with different train/test data.*

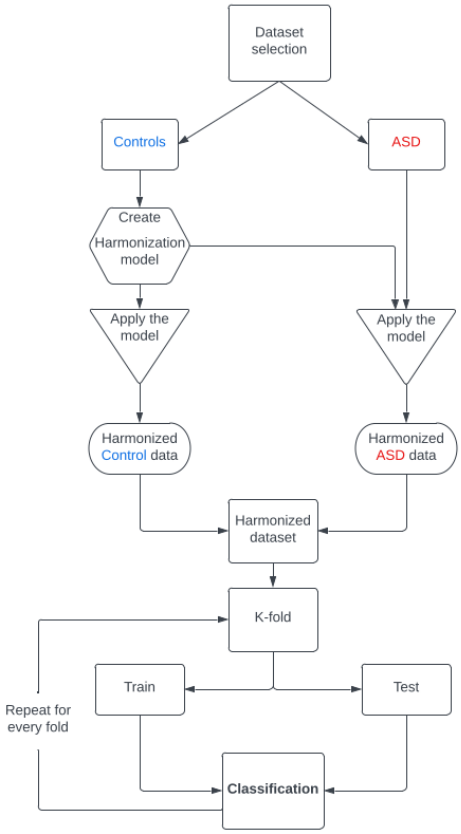


Figure 11.3: *Flowchart of the upstream implementation of harmonization, before the k-fold split: from the entire dataset controls are collected, and the harmonization model is built with only these controls data. The model is applied to the controls and ASD, and the harmonized dataset is created. With this harmonized dataset, a k-fold CV is performed: the dataset is split into train and test subsets and on these data, and classification is carried through.*

Chapter 12

Harmonization - results

As described in section 8.1, ComBat-based harmonization procedure simultaneously models and estimates biological and non biological terms, from data and algebraically removes the estimated addivite and multiplicative site-effect terms.

Harmonization procedure was tested on the datased created by using patients from ABIDE I and II after the cuts on sex and age discussed in section 11. From these patients we tested harmonization using Pearson-based correlation coefficients and wavelet coefficients of in phase time percentage. We run this analysis to have a quantitative and a visual representation of how features are modified after being processed with harmonization pipeline.

To test harmonization procedure and evaluate its performances in making feature uniform in relation to sites, we used NeuroHarmonize [1] package for Pyhton, developed by Pomponio et al. [39] which implemented and added some features to the previous NeuroComBat Python package [2].

To use Neuroharmonize package we need to input the feature data to harmonize and the corresponding covariate information with site, age, sex, and other biological quantities we desire to take into account. For our data these information were provided by a csv file on the ABIDE website. In this file, for each patient, are collected several medical and biologiclal information useful to run a great number of different analysis. A regression model is thus created with parameters estimated with the procedure explained in section 8.1. This model can be applied to the same input data

[1] https://github.com/rpomponio/neuroHarmonize

[2] https://github.com/Jfortin1/ComBatHarmonization

we used to generate it, or to new, unseen data acquired from the same sites we used to create the model.

We tested harmonization procedure on 1470 male patients coming from ABIDE I and ABIDE II of which 737 controls and 733 cases, choosing as covariate to preserve the age, and the FIQ (Full intellective quotient) to mantain important possible biolgical trends in the data and avoid overcorrection.

The central idea is to create the model on control subjects only, to operate without the influence of informations related to ASD patients whose feature might follow a different distribution compared to control. Once created the model on control subject, it is applied to both control and ASD subjects, following this procedure site-related information are eliminated from our data. To test the effectiveness of this harmonization procedure, we used a Random Forest Regression to test the site classification performances before and after the harmonization procedure. In this analysis, we split the model into a train and test subsets: using only control subjects from the train dataset, we created the harmonization model as explained before.

To actually harmonize data, the model is applied to all the data: Controls and cases of train dataset and Control and Cases of the Test. We remind that this approach, allows train and test dataset to remain independent, because harmonization model is only created on train subjects and once learnt the parameters, is applied to the test dataset. Thus, data we use to train the random forest, are kept separated from the test set, and there's no information leakage of the test data into the training data.

Two random forest classifiers were trained. One using raw (non-harmonized) data, and the second using harmonized data. Their task was to make a binary classification of data: site vs site, for each pair of sites.

Note: As mentioned in the previous paragraph, two sites namely `NYU_2` and `KUL_3` provided only ASD patients, and, since we are estimating the mean and the variance for each feature across all sites, when we train a model only on control cases, and we apply it on patients belonging to new sites, we don't have information to scale and modify those data so the model would output NaN values. This is the reason why in our analysis we excluded these sites since we

can't perform a proper harmonization of them.

Figures 12.1 and 12.2 show the results of classification site vs site using the two coefficients described above: Pearson and wavelet correlations. Sites along the x and y axes are ordered by incresging average age of patients and scores are reported in terms of AUC,

Specifically, fig 12.1b and 12.2b show the AUC score of the model trained with the raw, not harmonized dataset, while in figures 12.1a and 12.2a show the results obtained with harmonized data.

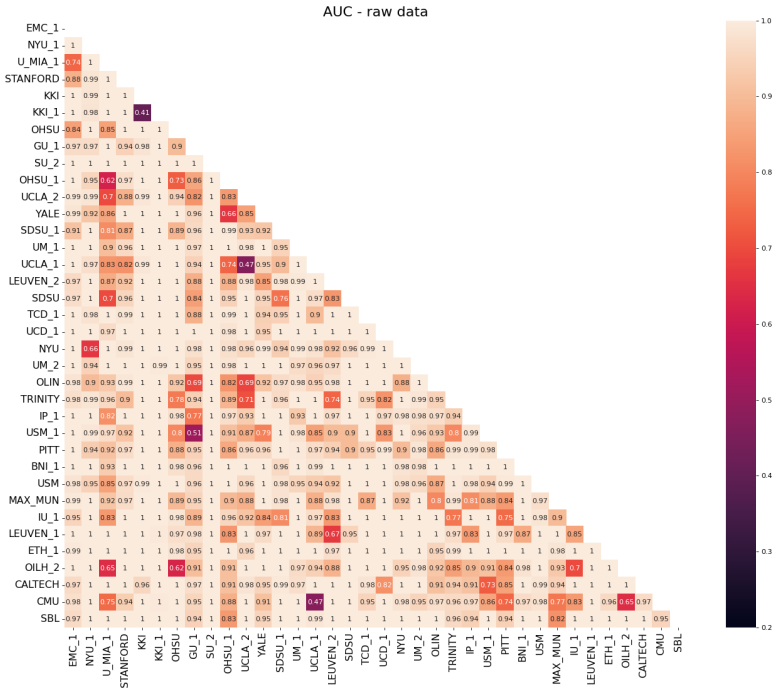
To have a visual representation of how much each feature is modified in respect of its sourced site, in fig 12.3 are shown two features among the 5995: *feature 324* and *feature 2800* as a function of sites. This figures show features computed with Pearson coefficients while in figure 12.4 are shown the same two features with wavelet coefficients. Just for a clearer idea of what is shown, feature 324 represents the correlation between right and left inferior frontal gyrus, and feature 2800 that between left precuneous cortex and the left inferior frontal gyrus. There is not a particular reason for choosing these two feature, they are just randomly drawn from the 5995 possible combinations. Their values are plotted as a box along the y axis and sites are indicated along the x-axis, it appears clear how the mean value of each feature for each site is shifted and features are stretched or shrinked to make them follow a more uniform distribution.

If we split this plot into control and ASD patients to see the effect of harmonization separately on these subsets, 12.5 shows feature 324 computed using Pearson coefficients and figure 12.5b with wavelet coefficients.

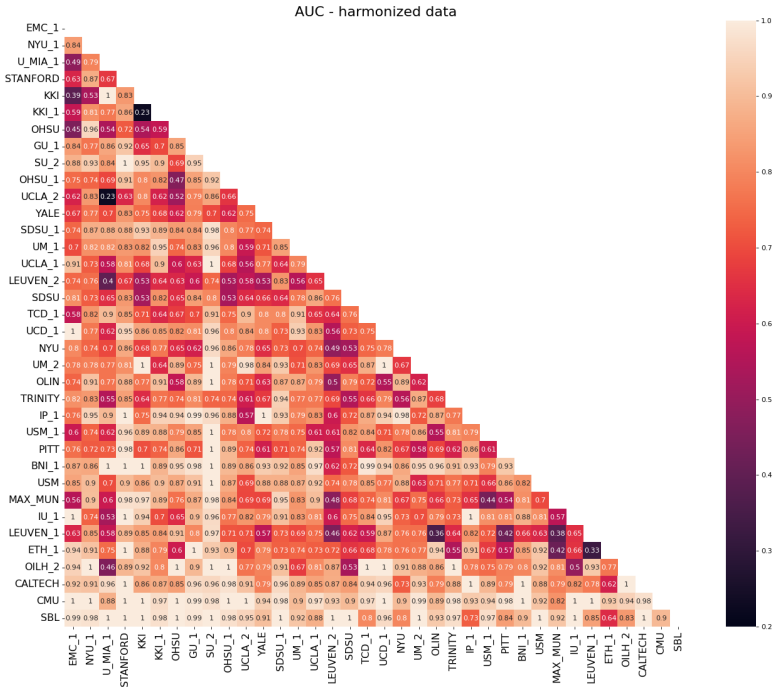
12.1 Comments on harmonization

Figure 12.2 shows a binary classification procedure with a random forest, for site discrimination of wavelet-based coefficients. It appears clear that with wavelet coefficients there is not the same outcome as with Pearson coefficients (figure 12.1), in fact with wavelets sites still remain distinguishable after harmonization.

Nevertheless, if we plot the mean value of a feature computed with wavelet-based correlation across sites (figure **??**) we note that there is a certain trend related to sites, but it is possible either

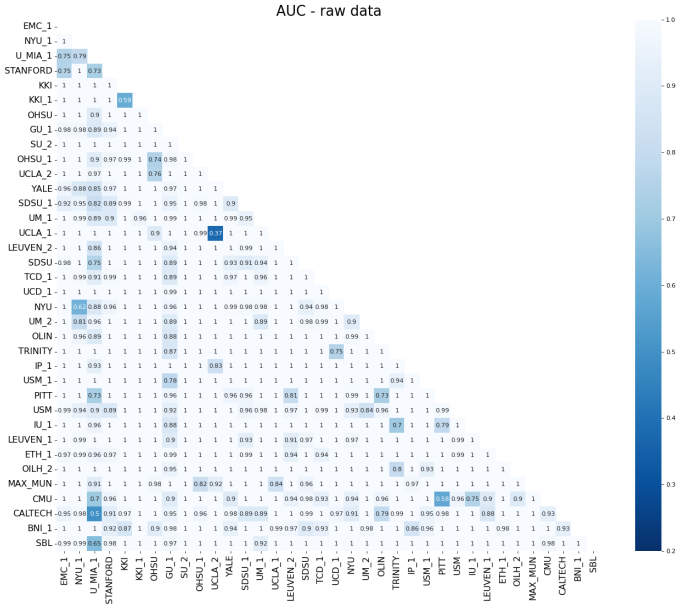


(a) *Heatmap with AUC score of a binary classification site vs site with raw data*

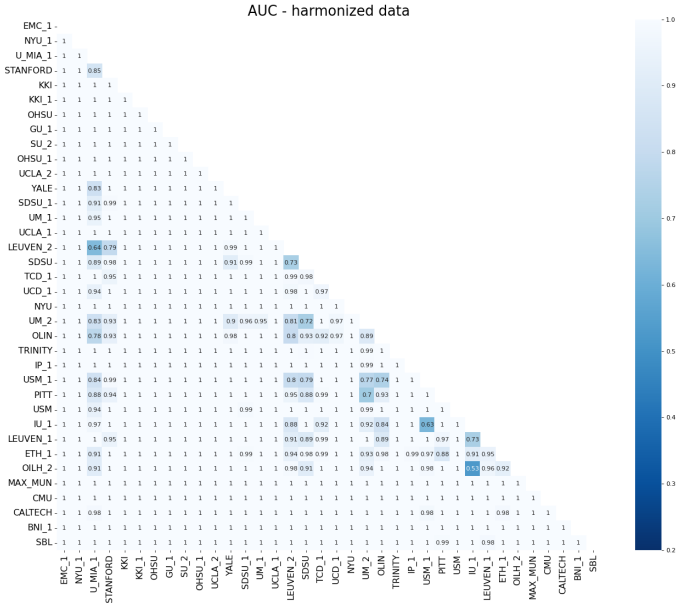


(b) *Heatmap with AUC score of a binary classification site vs site with harmonized data*

Figure 12.1: *Comparison of two heatmaps with AUC score of binary classification site vs site with* ***Pearson*** *based correlation coefficients of raw (fig 12.1a) and harmonized data (fig 12.1b) classified*

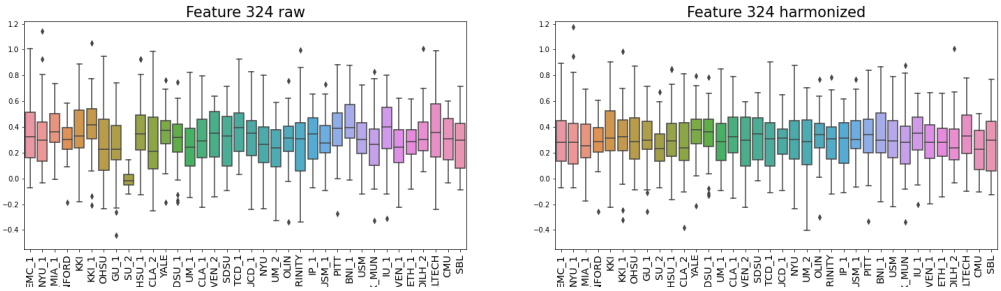


(a) *Heatmap with AUC score of a binary classification site vs site with raw data*

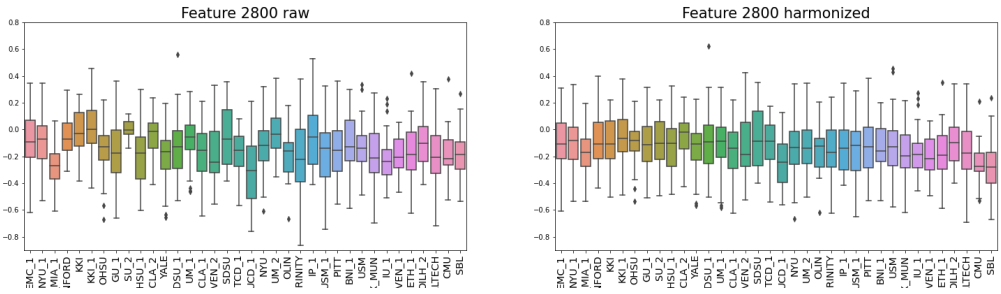


(b) *Heatmap with AUC score of a binary classification site vs site with harmonized data*

Figure 12.2: *Comparison of two heatmaps with AUC score of binary classification site vs site of* ***wavelet****-based correlation coefficients of raw (fig 12.2a) and harmonized data (fig 12.2b) classified using a Random Forest Classifier*

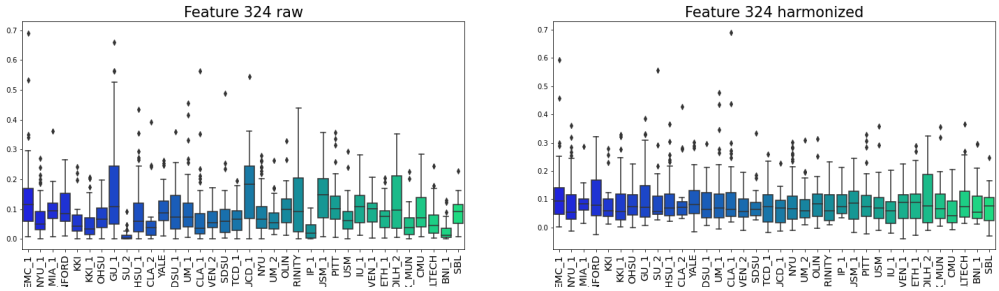


(a)

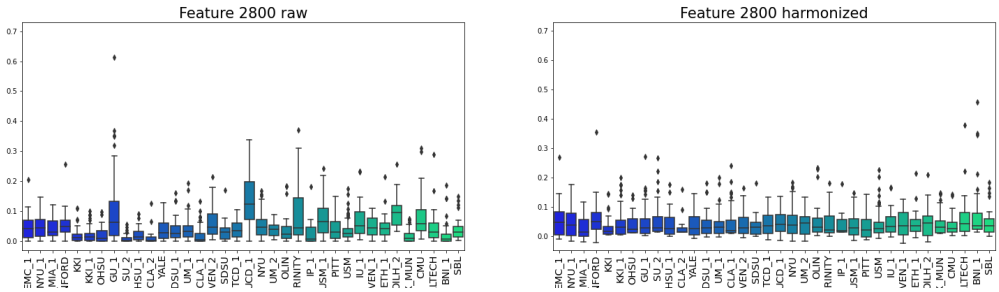


(b)

Figure 12.3: *Pearson-based connectivity coefficients per site for features 324 (fig 12.3a) and 2800 (fig 12.3b) before and after harmonization*

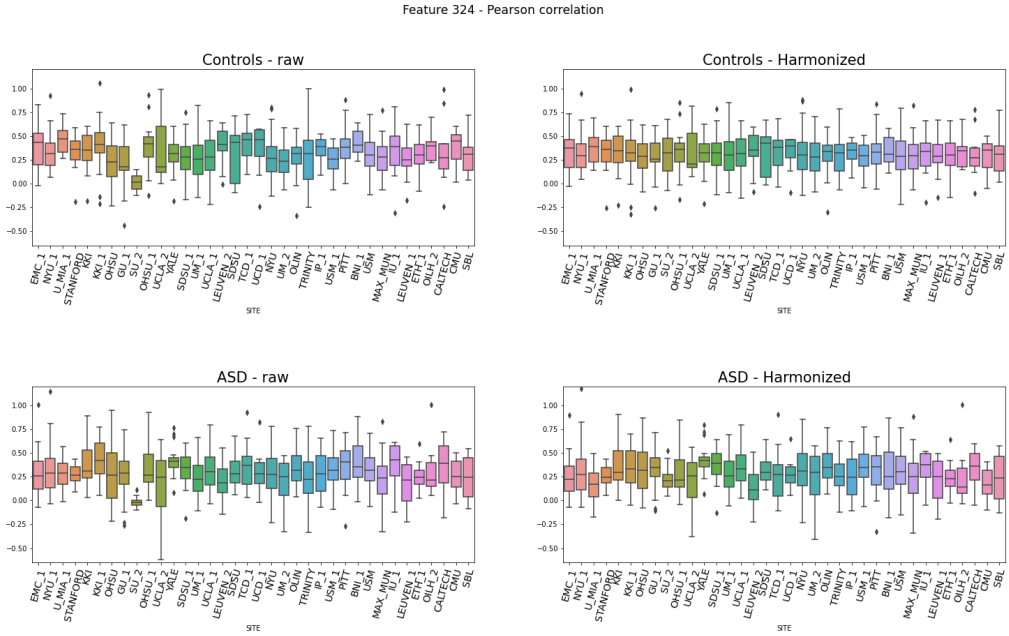


(a)

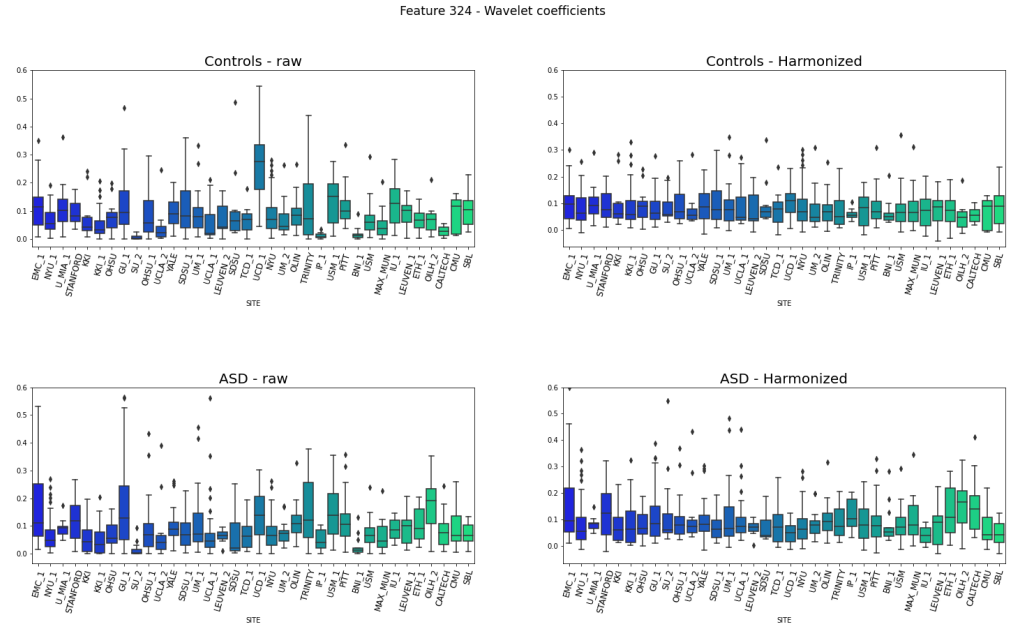


(b)

Figure 12.4: *wavelet-based connectivity coefficients of in-phase percentage, for features 324 (fig 12.4a) and 2800 (fig 12.4b) before and after harmonization*



(a) *Feature 324 of Pearson correlation coefficients*



(b) *Feature 324 of wavelet-based correlation coefficients*

Figure 12.5: *Pearson-correlation and wavelet coefficients for features 324 before and after harmonization with a separated plot for controls and ASD*

that we obtain this trend just for this choice of feature or that even if the mean values of these features underwent a level out procedure, they still remain recognisable because of the values they assume which is not in the same range for all the sites and this can help the classificator recognising sites differences.

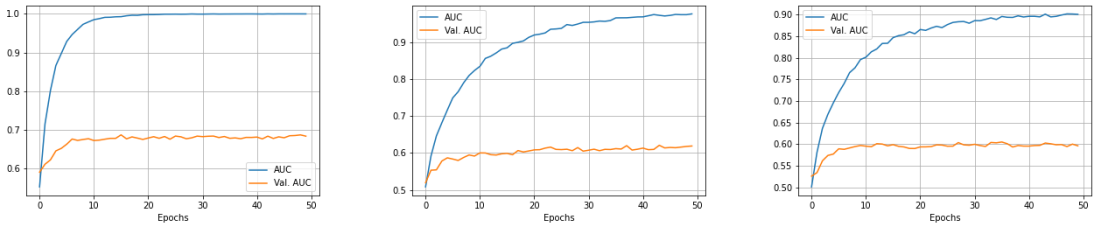
Both for wavelet-based coefficients in figure **??** and for Pearson-based coefficients in figure 12.3 we can notice a certain messy trend of features of raw data in respect to site (plots on the left side), which is not due to the presence of an unbalanced Controls/ASD number of patients per site, because from figure 12.5 we observe as well, a messy trend both for Controls and ASD. We can therefore conclude that in our data there's some kind of bias linked to the acquisition site. After harmonization procedure both Pearson-based coefficients and wavelet-based coefficients assume a more regular trend (plots on the right side of figures **??** and 12.3) This is a visual confirm that harmonization effectively removes site-related, but if for Pearson coefficients this is enough to remove inter-site variability, for wavelet coefficients

Chapter 13

Deep neural models

Deep neural network

In this work we performed a classification task, using a shallow neural network built using Keras library for Python. Since we are working in an unavoidable overfitting condition, we tried to build a network with as less feature as possible, but still preserving good classification performances. We then started the research of our newtork with a network comprising only 3 layers and we searched the minimum number of neurons to employ in each layer. We started with a trivial network made by just 3 neuron in the first layer, 2 nodes in the second and at last a single-output layer. Even if this was just an attempt to put a lower limit to the number of neurons, we noted that even with this configuration, the network tends to overfit our data. This is clear if we take a look at the learning curves in figure 13.1 which shows the training and validation AUC curves for two models: the first one of them (13.1c) even if with the simplified structure just mentioned, shows the classical trend of an overfitting state, however, it performes a bit worse than the other two more complex models. We continued adding neurons to each layer and tried different configurations. With a configuration of 8-8-1 performances were slightly higher, so we set the second layer to 8 neurons and tried changing the number of the neurons in the first layer, performances seemed to increase as the number of neurons increased until they reached a stable value. Different attempts and the related classification scores are reported in table A.3 in appendix. We decided to pick the configuration comprising the minimum number of neurons, beyond which performances were stable,



(a) *AUC learning curve corresponding to a model with a structure 64-8-1 nodes trained on the entire not-harmonized dataset*

(b) *AUC learning curve corresponding to a model with a structure 8-8-1 nodes trained on the entire not-harmonized dataset*

(c) *AUC learning curve corresponding to a model with a structure 3-2-1 nodes trained on the entire not-harmonized dataset*

Figure 13.1

and the addition of more neurons to create a more complex network was unjustified.

In our work we employed the neural network schematized in figure 13.2. This network consists of 2 hidden layers made up of 264, and 8 neurons respectively, both activated by a ReLU function and separated by a Batch-Normalization and a Dropout layer with a dropout chance of 30%. After the 8-node layer we added a second Batch-Normalization layer, before the output layer. At the end we put a single output layer with a single neuron for the classification output. This last layer is activated by a sigmoid function, which outputs a real number between 0 and 1.

To create the network we set the subsequent hyperparameters after a grid search on learning rate and number of epochs, setting a validation set of 25% of train data, and studying the evolution of the learning curves, to find the minimal parameters configuration which allows high classification scores. We trained the model using binary crossentropy loss function and optimizing parameters using Adam optimizer with a learning rate of 10^{-4}. Adam optimizer was chosen over Stochastic Gradient Descent optimizer since with Adam we empirically achieved similar classification performances with a lower error than with SGD.

Domain-adversarial neural network

As mentioned in chapter 11 an other kind of analysis was performed using a Domain adversarial neural network. This domain-adversarial or site-adversarial network is a model able to earn from

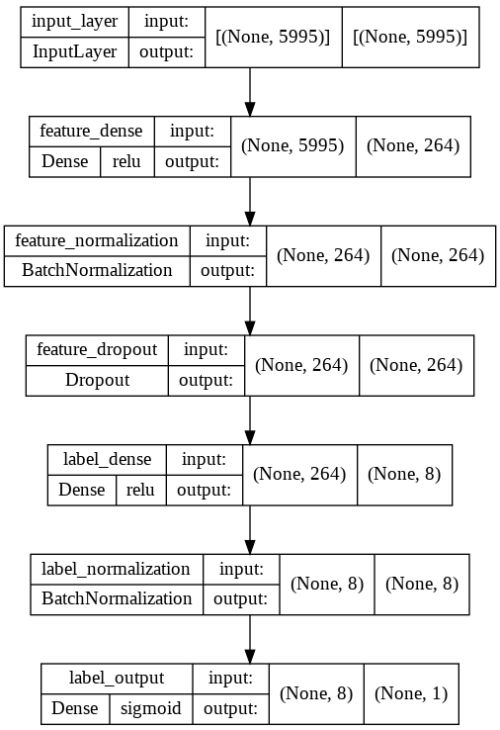


Figure 13.2: *Schematic structure of the deep neural network employed in this work for Control/ASD classification, with a scheme 264-8-1. Each box contains the layer's name, layer's task (Input, Dense, BatchNormalization etc..), activation function, input shape (number of neurons) and output shape.*

data both class and site information, and avoid some sort of bias due to site, when performing classification. The main idea behind this network comes from a revisited and customized version of the model described in chapterchap:adversarial, but before building this network we tried a different aproach.

The first attempt to implement a model able to predict category label without any influence of site's information is a model consisting on two different branches, one for the output of control/ASD prediction, and one for the output of the predicted site. With these two informations, a loss is created by combining two different loss functions, one for the class and the other for site: the idea is similrar to that proposed in the article **??**. This new loss is in the form

$$L = L_1 - \lambda L_2 \tag{13.1}$$

Where L_1 is in our case the loss for the binary classification Control/ASD that we want to minimize

(binary crossentropy), while L_2 is the loss for site classification: categorical crossentropy we aim to maximize in order to avoid learning any information related to sites. λ is a parameter empirically set, to control the contribution of the sites loss L_2 to the overall loss. This way, the minimum of the total loss is reached with parameters that minimize the category classification loss and maximize the site classification loss. Performances of this kind of model are strictly linked to the value of the parameter λ, in a sense that if a certain value gave an optimal performance on a certain dataset, on an other dataset, for example one created with some more constraints on patients, the value for which one had the best scores was slightly different. And also, since when we first employed this model it gave slightly worse performances than simple deep model, we looked for an other strategy and carried out the analysis with this second type of model. Thus we reached to the domain-adversarial neural network. Even though this network is based on the same principles of the neural network with combination of two losses (minimize the error on classification of ASD/controls, and maximizing the one on site classification), it accomplish its task in a different way. Our task is in fact to avoid the model to learn some site-related pattern from data and we accomplished so with a model whose inner weights are updated moving towards a minimization of the gradient for classification, and in the opposite direction for site classification, as explained in section 9.

To costruct our domain-adversarial neural network, we followed the same structure of the deep neural network implemented before, and we added the domain-adversarial branch. We can describe the structure of this model as consisting on three parts: a feature extractor branch takes input data and creates an inner representation of them within its structure; at this point the network is forked into two branches: a label classificator for control/ASD cassification and the second branch for site classification. At the top of the site-classification branch, the gradient reversal layer was placed.

In details the three components that make up this model are: the first feature extraction branch comprising the input layer, the 264-nodes and the 8-node layer. Between the last two, we put a Batch-normalization and a 0.3 dropout layer, as we did in the first DNN. From this point on the network is splitted into two branches: the categorical classifier, similarily to the DNN consists of a last layer with a single neuron activated by a sigmoid function, for the output of control/ASD classification. The other branch, the site classifier consists of a gradient reversal layer as the first

layer followed by a layer with 16 neurons, a Batch-Normalization layer and the last layer: a multi-output layer with N nodes, being N the total number of sites input data belong to. The effect of the gradient reversal layer can be regularized with a factor to weight the contribution of the site loss on the feature extractor parameters. We empirically set this value on 0.3. The structure of this network is illustrated in figure 13.3

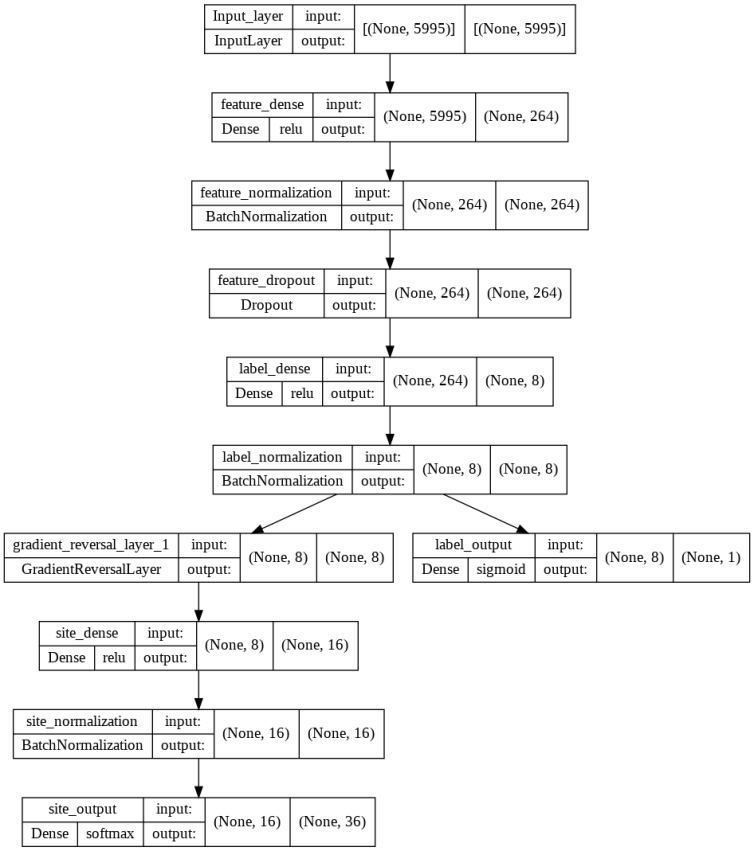


Figure 13.3: *Structure of the domain-adversarial neural network with two outputs: a single-neuron output for binary classification controls/ASD and a multi-class classification for site classification. Each box contains the layer's name, layer's task (Input, Dense, BatchNormalization etc..), activation function, input shape (number of neurons) and output shape.*

The final layer of the site-classification branch is activated by a softmax function, and the loss function employed is the categorical crossentropy, commonly employed for multi-class classification tasks as explained in 3.4.

Note: To work with this loss we need to encode each site numerically before before use its label

into our model because functions like these can't deal with string variables such as site names. For this reason we have to associate each name to a number: site_a = 0, site_b = 1 and so on. If we stop to this point and use integer numbers $\{0, ..., s\}$ as labels (where s corresponds to the integer associated to the last sites : N_total_sites - 1), our classification model would end up considering sites with higher number as "larger "than others. This should be avoided in our case since all the sites are "equally important ". To avoid this a common strategy is to transform these variables according to a technique called One-hot-encoding. This common approach converts each value to a vector of len s, containing all zeros but in the entry corresponding to that site, where it puts 1. With this transformation, each site label becomes a binary vector, with just a single 1 places in correspondence to that specific site.

We tested both these different adversarial models (the model with loss combination and this with gradient reversal) and we noticed that they had more or less similar performances, but we choose to carry out the following analysis with this second model, because, even if, like the model with loss compositions, in the gradient reversal layer, a parameter λ controls the weights updating during the backpropagation, results obtained with this model are more stable in respect to the the value of this parameter. Moreover, there are other studies [**?**] that compare the performances obtained with different implementation of adversarial network, and, even if they work with different data (text data) from ours, they find the DANN the best performing implementation; since this network appears to be more promising than the simple loss combination networks, we choose to employ this network in the following analysis.

Chapter 14

Results of classification

In the following section we report results of classification controls/ASD obtained by following different strategies of harmonization as explained in chapter 11 using the neural network and the adversarial neural network presented in chapter 13.

From now on, in all the tables we denote as "Harmon k-fold "the pipeline where harmonization is implemented inside the k-fold and harmonization model is created only on control train data. "Harmon. upstream "denote the pipeline where harmonization is implemented upstream, before the k-fold partition of dataset into train and test; with this pipeline, harmonization model is created using data of all controls. "No harmon. "denote the pipeline of classification of raw data, while "Adversarial "denotes the pipeline where we use the domain-adversarial neural network giving as input raw data.

And when listing datasets, we denote as ABIDE I+II / ABIDE I / ABIDE II datasets with cuts on covariates such as sex and ages as discussed in chapter 11 and when we select only patients with open eye we expressly specify it.

14.1 Results with Pearson coefficients

The first classification analysis was run on Pearson-related correlation coefficients. To recall the different analysis pipeline we can take a look at flowchart 11.1, and in this analysis "Coefficients selection "translates as we are selecting coefficients based on Pearson correlation.

Results of classification are reported in terms of mean AUC across all the 5 folds. We listed in table **??** results obtained with our DNN model followind different harmonization implementation and with the domain-adversarial network.

In table 14.2 and we compared some of these results with a Random Forest classifier to assess whether is convenient to use a deep model instead of a conventional machine learning algorithm such as Random Forest. Comparison with Random Forest were performed on the whole dataset ABIDE I+II, and on ABIDE I with open eye, since on this dataset we obtained the best classification performances. Results of DNN and RF concern the three pipelines of: hamonization inside K-fold, harmonization upstream, and raw data (non-harmonized).

Dataset	Harmon K-fold	Harmon. upstream	No harmon.	Adversarial
AB I+II	71±1	74±2	73±3	70±3
AB I	71±3	74±2	72±3	71±4
AB 2	63 ±5	68± 6	64± 4	66 ± 4
AB I + II, eye open	72±3	75±4	72±3	71±3
AB I, eye open	73± 1	76±2	72±1	72±4
AB II, eye open	66±3	70±6	69±6	68±6

Table 14.1: *AUC score obtained with the deep neural network, using Pearson-related coefficients and following different harmonization procedures.*

AB I + II		
	DNN	RF
K-fold	71±1	66+2
Upstream	74±2	72+1
Non-harmon	73±3	65+1

AB I eye = open		
	DNN	RF
K-fold	73± 1	70+3
Upstream	76±2	71+2
Non-harmon	72±1	68+3

Table 14.2: *Classification score of a DNN and a RF classifier, using Pearcon correlation coefficients, for dataset: ABIDE I + II and ABIDE I with open eyes*

14.2 Results with wavelet coefficients

The same analysis performed with Pearson-related coefficients were carried out with wavelet-related coefficients. If we refer once again to the flowchart 11.1, now with "Coefficients selection "we denote four different dataset created using wavelet coefficients, whose classification results are reported in the respective tables.

1. In phase wavelet coefficients, with results reported in table 14.5
2. In phase and counter phase coefficient stacked together in a single array long twice the in-phase coefficients array. Results reported on table 14.3
3. Counter phase wavelet coefficients, with results reported in appendix in table A.1
4. Coefficients resulting from the subtraction of the two: In phase - Counter phase: denoted as w_in - w_out, reported in appendix, in table A.2 Ha senso questa analisi? Come si giustifica?

We choose to put some tables in appendix to avoid to weight down this section with tables. We choose to put in appendix results less relevant both from a physical point of view and for classification scores.

Dataset	Harmon K-fold	Harmon. upstream	No harmon.	Adversarial
AB I±II	65± 2	74± 2	66± 2	67 ±2
AB I	67± 3	73 ±2	68± 3	68± 2
AB 2	62 ±4	68± 4	63± 4	62± 4
AB I ± II - eye open	65 ±4	71 ±3	66 ±3	66± 3
AB I open eye	69± 4	74± 6	67± 3	67 ±4
AB II eye open	66 ±2	69 ±3	67± 2	68 ±2

Table 14.3: *AUC score obtained with the deep neural network, using wavelet coefficients w_in and w_out stacked together, following different harmonization procedures.*

As we did with Pearson-related coefficients, we compared results obtained using a DNN with results obtained from a Random Forest. We run this comparison only with coefficients that gave the best classification performances between the four coefficients (or combination of coefficients) listed at the beginning of this section. Results are compared in table 14.4

ABIDE I + II		
	DNN	RF
K-fold	65± 2	60+2
Upstream	74± 2	74+3
Non-harmon	66± 2	59+3

AB I eye = open		
	DNN	RF
K-fold	69± 4	62+5
Upstream	74± 6	69+3
Non-harmon	67± 3	61+5

Table 14.4: *Classification scores comparison between the DNN model and a Random Forest classifier, on wavelet coherence coefficients W_in and W_out, for dataset: ABIDE I + II and ABIDE I with open eyes*

Dataset	Harmon K-fold	Harmon. upstream	No harmon.	Adversarial
AB I±II	65± 2	71± 2	62± 2	64± 1
AB I	66± 3	73 ±3	65± 3	66± 2
AB 2	62 ±3	65 ±3	61 ±4	60 ±2
AB I ± II - eye open	64± 4	70 ±3	66± 3	65 ±3
AB I open eye	64 ±5	68 ±5	64 ±3	64 ±4
AB II eye open	66 ±2	68± 3	66± 4	65 ±5

Table 14.5: *Classification scores: AUC, only w_in coefficients*

14.3 Results with PCA

As mentioned in section 11 we choose to run PCA analysis using coefficients that gave the best classification results in the previous analysis namely Pearson correlation coefficients. Furthermore, to not weight down this table, we limit our dataset choice to the whole ABIDE I + II dataset and to ABIDE I only open eye, because on it we obtained a bit higher classification results.

We started our analysis choosing the proper number of PC to explain $> 90\%$ of variance and then gradually reduced them as reported in table 14.6

Explained variance		
N PC	ABIDE I + II [%]	AB I eye open [%]
800	93	–
400	78	98
200	62	77
100	48	58
50	36	41
20	24	26

Table 14.6: *Explained variance [%] for different numbers of principal components, and for the two dataset used in the analysis*

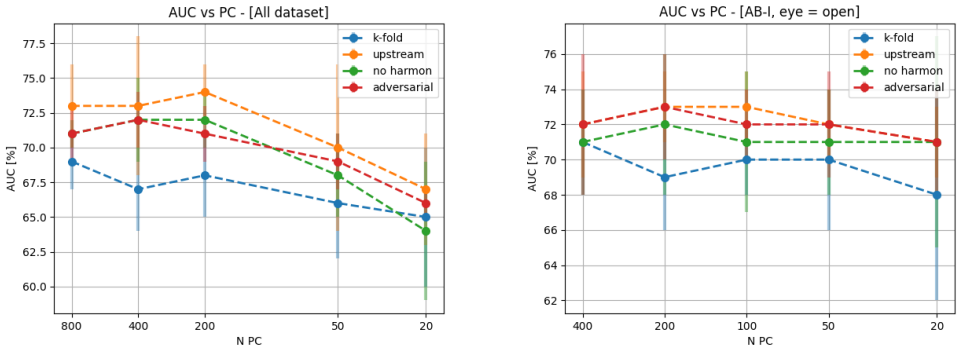
We followed the same pipelines as we did for Pearson and wavelet coefficients, following the four different harmonization procedures. Results obtained from this analysis are reported in table 14.7 and shown in figure 14.1

14.4 Discussion on classification results

Comparison between Pearson and wavelets From results reported on section 14.1 and 14.2 we can observe that average results obtained with Pearson correlation coefficients are systematically higher than wavelet-based correlation coefficients. We can therefore assume that working with wavelet-based correlation coefficient obtained through processes described in section **??** does not

PC	Dataset	Harmon K-fold	Harmon. upstream	No harmon.	Adversarial
800	AB I+II	69±2	73±3	71±1	71±2
400	AB I+II	67±3	73±5	72±3	73±3
	AB I eye open	71±3	72±3	71±3	72±4
200	AB I+II	68±3	74±2	72±2	71±2
	AB I eye open	69±3	73±2	72±4	73±3
100	AB I+II	67 ±2	72±2	69±1	70±2
	AB I - eye open	70±2	73±2	71±4	72±2
50	AB I+II	66±4	70±6	68±3	69±2
	AB I - eye open	70±4	72±3	69±3	72±3
20	AB I+II	65±5	67±4	64±5	66±1
	AB I - eye open	68±6	71±2	71±6	71±3

Table 14.7: *AUC score obtained with the deep neural networks, using a decreasing number of principal components, following different harmonization procedures*



(a) *Classification on the entire dataset ABIDE I - ABIDE I+II*

(b) *Classification on dataset ABIDE I - only open eyes*

Figure 14.1: *AUC scores obtained from control/ASD classification of data using decreasing numbers of principal components for Pearson-based coefficients. Results obtained with different analysis are represented with different colors: AUC scores with data harmonized inside k.fold are marked* ***blue****, AUC score with data harmonized upstream* ***orange****, AUC scores with raw data (not harmonized)* ***green*** *and AUC scores classification with the adversarial network* ***red****.*

bring any advantages to this kind of analysis. A possible reason for this outcome could be the loss of information during the extraction of these coefficients because of the introduction of redundancy and randomness into this analysis. Starting from two timeseries wavelet transform compute a 2D matrix for each timeseries and uses them to calculate the cross-power spectrum as explained in the dedicated section. Doing so we create a redundant representation of timeseries data moving from one dimension to two-dimensional data. We furthermore compare the cross-power spectrum obtained from the wavelet transform of each timeseries, with a red-noise cross-power spectrum to assess the significativity of our data. In the end we shrink again our dimensions to reduce to a single data point representing a correlation.

Since differences of traits between controls and ASD are not very strong, this process can cause the loss of this weak information, which does not happen with a more direct and linear correlation calculus like with Pearson coefficients.

This lower scores is found both on wavelet coefficients of in-phase and counter-phase time percentage, and not even the combination of this two in a double-sized array, or with a coefficient made by the difference between these two coefficients brings any benefits, as an evidence that this procedure is not the best suitable to our classification goals.

Effects of a smaller and more regular dataset

Let's focus for a moment just on table 14.1 (even if these considerations also apply to wavelet results) to discuss the benefits of reducing the dataset size making cuts on covariates to run analysis on a more homogeneous dataset. We can state that the eye status of patients during the scan has a non-negligible impact on the outcome of classification since limiting the dataset only on subjects with open eyes systematically improves classification scores of more than 1 % for every dataset (ABIDE I+II, ABIDE I only, ABIDE II only). As introduced in chapter 1.1 there can be differences in brain functional areas between patients with open eyes and patients with eyes closed. This is related to activation of different cortex and gyri areas or because during the scan, some patients may be fallen asleep, and this heavily modify the functional connectivity matrix. For this reason, removing these source of variability could bring to a cleaner distinction of relevant patterns between controls data and ASD.

We can also notice that results on the dataset made by ABIDE I + II are mostly driven to the data belonging to ABIDE I than those belonging to ABIDE II, and this is true either if we restrict our analysis on open eyes patients only or we do not. In fact, putting together the two dataset does not lead to a great improvement of scores than limiting just on ABIDE I. It is possible that this trend is due to two main factors: the greater number of ABIDE I patient with respect to those in ABIDE II left after all the cuts, and the presence in ABIDE I of a great site: NYU containing a great percentage of patient which are also balanced between controls and ASD. This could monopolise the results since a great percentage of train and test data would belong to this center leading to a lower variability due to site-related features as demonstrated in studies like [48]. Ma allora perchè questo trend lo vediamo anche sui dati armonizzati?

Comparison between a deep neural network and a random forest classifier hfill

As a general rule of thumb is alway a good practice to compare results obtained with a complex model, to results obtained with a simpler machine learning classifier like Random Forest. Instead of immediately opting for a complex model, simpler methods should be tried to establish a baseline and allow a meaningful comparison. The principle of Occam's razor, when applied to machine learning demands that if two model perform quite the same, the simpler of the two should be picked [17]. This is true especially for our data since we start from a situation of overfitting and there is no need to choose a complex model a priori if this choice is not supported by data. This is the same principle that lead us during the choice of the best structure and the best number of layers and neurons of the deep neural network.

Comparing the results obtained by the deep model with those obtained with a Random Forest, (table 14.2 and 14.4) it is possible to deduce that a deep model is more adapt to find characteristic patterns between controls and ASD than a simpler Ranfom Forest classifier. Scores obtained with a deep model are systematically much higher than those obtained with a random forest classifier. This trend is true, with the exception of data classified with the upstream harmonization pipeline, but as explained in the paragraph below, this procedure leads to biased data and as a consequence results obtained with this pipeline are not reliable. We are still going to use Random Forest classifier in some of the remaining anslysis just as a comparison, even though we assert that a deep model is

more suitable to understand differences between the two classes of interest.

Effect of harmonization An other common trend that stands out from all the analysis is the systematic improvement of AUC score, when we use data harmonized with the pipeline of upstream harmonization (explained in section 11 and outlined on flowchart in figure 11.3. This improvement disappears when we implement harmonization inside the k-fold cross validation procedure (sketched in flowchart 11.2). These two implementation differ on the order by which harmonization is implemented: with upstream harmonization we harmonize the whole dataset and after we split it into train and test to run the cross validation procedure; conversely, with harmonization implemented inside the cross validation procedure, we run harmonization only after the division of the dataset into a train and test subsets. We can state that these high results are an evidence that the upstream harmonization is the wrong way to proceed, since classification results are affected by a bias due to data leakage which occur if we implement harmonization upstream. It is also possible that some good results on other similar studies, obtained with harmonization implemented in this same way ([27]) may be due to a similar data leakage as we have in this analysis. As a general definition, we have data leakage when information outside the training set is used to create and train models. This additional information allows the model to know something additional about the data that otherwise it would have not known. This leads to an improvement of model's prediction and an overestimation of performances. In this case we have data leakage when we harmonize the entire dataset before splitting it into train and test, because, according to this way to proceed, when we use train data, they already contain informations about test data. In fact, harmonized train and test data they have been created using covariates, features and other informations belonging to the whole dataset. We can then assert that the right way to implement harmonization is inside the k-fold cross validation, otherwise, if we don't run a cross validation, it is important to implement it after the splitting of dataset into a train and a test goups. This way, we can create the harmonization model only on control subjects belonging to train dataset, and we train the model with this dataset so there are not external information in it during training. We can then harmonize data belonging to test applying the harmonization model, with parameters created only on the train dataset, on test data. For this reason, from now on, when we refer to "harmonized data ", we are

referring to the procedure according to which data are harmonized inside the k-fold CV.

Comparing results of harmonized data with results on raw data we don't notice any particular improvements. This is likely due to the weak information contained within our data. Distinction between healthy and ASD subjects based only on functional connectivity data is not a straigthforward task, and it is possible that these results are driven by all the information available. Removing site-related information from our data does not bring relevant improvements because this is all the information we can retrieve from our data, and with this procedure we are not able to bring out new information from our data. However harmonization procedure can become a usefull tool to remove some noise from data and obtain a cleaner assessment of what feature are the most discriminative between controls and ASD. This theme is entirely addressed in chapter 15 talking about feature importance.

Results with adversarial Looking at tables 14.1 we notice that the adversarial model perform slightly worse working on Pearson-based coefficients while for wavelet coefficients (tables 14.5 and 14.3) we obtain on average the same values between DNN and adversarial. A reason for this is the possible confusion in weight updates introduced by thhe adversarial network that sometimes may generate noise between weights belonging to the feature extraction branch. So far, in different papers, [23], [**?**] or [**?**] (where, however adversarial is introduced just by combining losses), adversarial networks are employed with only two domains, which can be considered as the equivalent of our sites. In our data we deal with 36 sites when we consider ABIDE I+II, or, best case scenario, about 15 sites when we reduce to ABIDE I or II with just open eyes. This way even if the site-predictor branch is encouraged to learn site-distinctive traits, it is not always able to distinguish between 36 sites, so this can cause confusion during backpropagation on weights of the feature extractor branch. This noise introduced by this process may be the cause of this slight worsening of classification performances. For this reason it is possible that when dealing with more than two domains (sites) this implementation of an adversarial network is not the most suitable way to proceed to remove site-related patterns from data.

Effect of dimensionality reduction The same alertness we paid for the implementation of harmonization was applied when extracting PCA. As mentioned in the overview chapter 11, it is possible to make the mistake of computing princial components on the whole dataset, before its separation into train and test sets. For this reason we implemented PCA extracting PCs from the training dataset and applying the transformation to the both of train and test dataset.

At a first glance we can notice that reducing the number of PCs leads to a gradual reduction of classification performances. This trend is more evident in the whole datataset, while in the ABIDE I only open eyes dataset, we obtain a flatter performances curve. A reduction from 5995 to 800 or 400 feature is a substanctial reduction since we are lowering our dimensionality by an order of magnitude still preserving an high classification AUC comparable to that obtained without PCA. In this analysis as well, data leakage due to the prior harmonization of dataset plays an important role as it leads to a rise in classification performances for any choice of number of PCs.

An other aspect that leap out comparing results with PCA with results without it, is a light improvement of performances of the adversarial learning. An explaination to this can be that a reduced source of error (deriving from this dimensionality reduction) brings to a reduced variability in site-related features making these confounding feature more recognizable. This could lead to a smoother weight updates that doesn't affect classification weights as much as it does with raw data.

What arises from this analysis, is that PCA represent an important strategy to tackle the problem of dimensionality. It is plausible tbat among all the 5995 features, a lots of them are uninformative for Control/ASD classification and they just increase dimensions of data without any benefit, or worse, just adding noise. This can be linked to a biological reason: we can in fact imagine that there are some brain areas that just don't significantly change their activity between healthy and ASD people. Following this explaination, all the coefficients representing a link between them are just similar between the two groups and don't bring any benefit in this discrimination. For this reason, reducing the source of noise due to this data-points leads to the creation of an equally informative dataset, but with a reduced source of error.

Chapter 15

SHAP - Implementation and results

In this chapter we are going to give a look at how we implemented a feature explainatory model using the DeepSHAP algorithm described in chapter 10, and we compared its results with a more conventional random forest feature importance analysis.

Here we present a brief summary of all the analysis we performed using feature importance informations.

We choose to run this analysis only on Pearson correlation coefficients since they gave the best classification results which means achieve the best separability between controls and ASD. We used the whole dataset consisting of ABIDE I + ABIDE II patients with the same characteristics as we choose for classification: only males and with an age range of 5-40. In addition we put in appendix some results obtained on the dataset ABIDE I only open eye.

Metto una lista numerate? Devo essere più preciso in questa overview? In this section we are looking for what are the most important features that guide the prediction made by a model. These features will be representative of some altered functional connections between brain areas. From the study of these altered connection we should be able to identify biological areas relevant for the distinction between an healty and with ASD subjects.

What we are looking for in this section is whether there are some features whose contribution was greater than others during the distinction between controls and ASD data, and if these feature are recurrent, regardless the type of analysis we are carrying out, or the machine learning classifier

we use.

As a first, itroductory analysis, in section 15.1 we start analyzing the most important feature that lead the classification of the DNNs models, using SHAP. We extract the most important features for each of the pipeline discussed in section 11, our goal is to check if the presence of harmonization, implemented by different strategies, significatively alter the contribution of features in the output of the model. In section 15.1 we repeat the same analysis earlier done with DNNs, with a random forest classifier. For this machine learning model, feature extraction is implemented by the use of the feature_extraction methods provided by sklearn. In these analysis our goal is to understand if a different machine learning algorithm gives results relying on different features than a deep network model. We also check the impact of harmonization just as we did with the deep models These, however are just a first, preliminary inspections and for this reason we limit our focus to the first 20 important feature for each pipeline. We limit choose this number because we can easily visualize them and have a visual comparison to find differences between the diffent analysis.

A more meaningful analysis, is executed afterwards. We select just 5 pipelines among all the one considered earlier, because we regard at these 5 as the most significative analysis. We will explain more in details in section **??** what these analysis are and why we excluded the others and we will present results on common important features between these 5 analysis. From them we examine the biological meaning of these feature and the brain areas they involve, to create an histogram of the most involved areas in discrimination of healthy and ASD subjects.

To instantiate the desired class with SHAP: DeepExplainer in our case, we require the trained deep model and a fraction of the training data to use as background. We can choose as many background data as we want keeping in mind that the bigger is this background subset N_{bkg}, the more accurate is the computing of shapley values, but the more computationally expansive this process will be. In particular using a big amount of background data, would occupy a vast amount of RAM memory and eventually run out of it; However, since the error we make on shap values linked to this choice is $\sim 1/\sqrt{N_{bkg}}$, a background dataset made of 100 samples is already a good compomise to have a relatively small error. We choose though, a background size of 500 samples

for the entire dataset and a size of 100 samples when working on ABIDE I only open eyes, since we had a reduced number of train data to collect background data from.

Once the explainer is created on our model, we need to input as many test dataset as we want to explain, and for each one of them, the explainer will output an array containing a shap coefficient for all the n_features of each test data. Once inputed a batch of test data, we obtain a matrix of the same size of the test dataset we used, shaped n_samples x n_features, containing all the shapley values for each feature of each test data. We choose to input all the test dataset for each fold to obtain a more statistically accurate estimate of feature importance.

Since we are presenting our results following a k-fold cross validation scheme, we implemented this procedure inside the k-fold CV. To do so and collect a final and general result, we computed the shap values for all the test samples of each k-fold and we stacked them together. At the end of the CV, we have a matrix of shap values, and we can proceed to visualize the results.

Different types of SHAP plots

We can visualize, for each test data, the contribution of each feature on pushing or pulling the predicted output from the baseline output of our model, but to produce this kind of plot we need to save the test datasets as well. An example is shown in figure 15.1. This type of plot, called *waterfall plot*, shows the result of each feature on a single test data. We have the baseline value of the model in the lower right corner and on the top left the output of the model with this test data. For each feature, positive shapley values, representing positive contribution to our output are colored red and negative ones are blue. Even if this is just an example taken from a single test data, it representative of all the analysis where there is not a feature whose contribution is way greater than the others. In fact each feature makes a small contribution to the final outcome, but the sum of them push the model towards an output.

If we want look at the overall result on all the test data, it would be inconvenient to create a plot like 15.1 for each data. For this reason we have to look at an other plot called *force plot*

The force plot of feature's importance can involve either the module of shap values or shap values themselves (positive or negative), an example of the latter is figure 15.2a: this plot shows

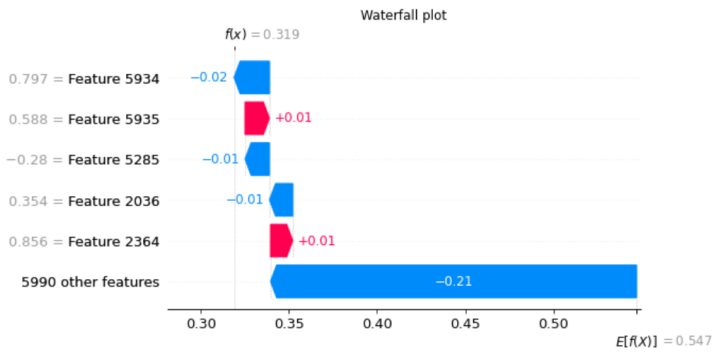


Figure 15.1: *Example of a waterfall plot obtained with a single test data. For each row is displayed the feature's name and value and its contribution to the final output: negative contribution are colored blue and positive ones red. It is also marked the reference output in the bottom right corner, and the output of this instance on the top left.*

for each instance (for each sample from the test data) [1] the contribution of one particular feature on the output computed with that test data. In other words, for each test data, we have an array of shapley values: one for each feature, and these shap values for each features are represented by a dot, placed to the right or to the left of the thin black line corresponding shapley values with a value zero (a zero value indicates that for a certain test data, that corresponding feature did not contributed to the output). This position in respect to the black vertical line, indicates the sign of that contribution: on the right we found positive coefficients that gave a positive contribution pushing the output towards values greater than the reference output, and on the left are placed those that had a negative contribution. Just to be clear, positive and negative adjectives are to be intended in a mathematical sense, as greater or smaller than zero, and pushing the output towards greater values (closer to 1 than to 0), means that the input data is being classified as autism. This process of placing shap values for each feature in a plot, is repeated for each test data to obtain a scatter plot of shapley values for each feature, where features are ordered by importance. Each spot is also colored red or blue. Color indicates the magnitude of that feature calculated in respect of the mean value of that feature across all the test dataset. With this additional information we are able to understand if a great or a small value of a feature pushes the model towards one output

[1] https://shap.readthedocs.io/en/latest/

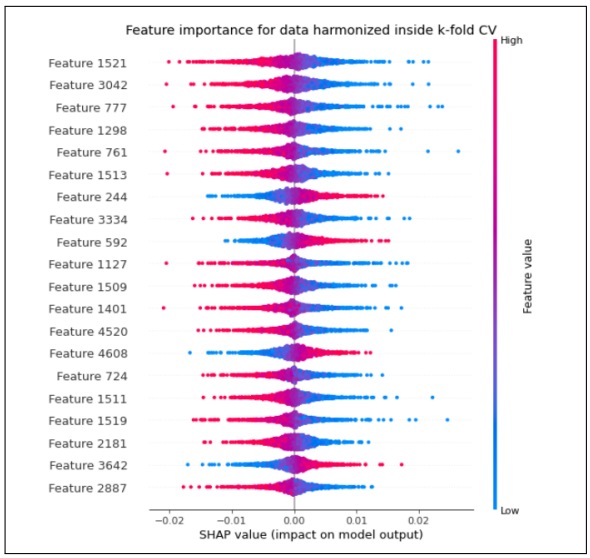
or the opposite (0 or 1). As a practicle example to understand this concept of colors, with our data, if we ask "is a strong or a weak correlation between two areas, that contributed the most to the prediction of my model towards high values (and then more close to a label 1 which means autism)?".

However, for our data, this kind of plot, is not the best suitable for readibility because of the large amount of features and the small contribution of each feature to the final outcome, in respect to other feature. This results in a smoothed scatter plot for each feature, containing many but unnecessary details which make this kind of plot not the most clear to assess how much a feature is important for that model.

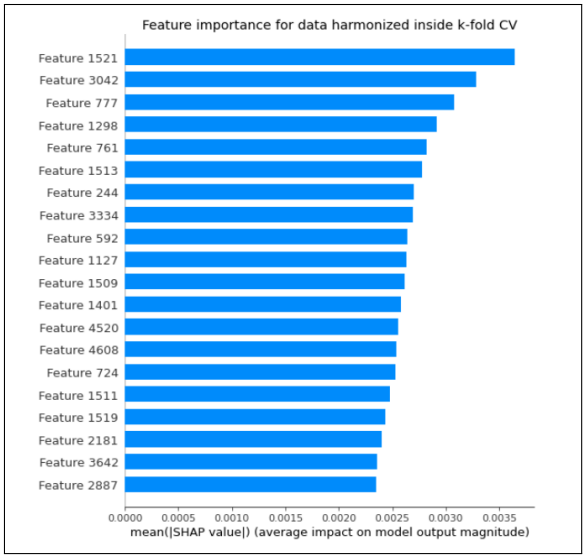
We can choose to visualize the total amount of "contribution "of each feature, by considering the absolute value of the shap values of figure 15.2a to obtain figure 15.2b representing a bar plot of all the first 20 important feature from the most to the least important. Each bar represents a feature importance, computed from equation 10.7.

For a better readability we prefer reporting the results like this, in terms of absolute shapley value, where we just consider the module of the shap values, regardless its positive or negative contribution to the output, just to understand the absolute contribution of a single feature for a given analysis pipeline.

From this plot we have a more immediate understanding of what are the most important features, and, as we noticed from the example in figure 15.1, there is not a standing out feature, or group of feature that have a contribution way greater than the others, but in fact feature's importance has a quite smoothed descending trend. An example to visualize this trend is shown in figures 15.2b representing the first 20 most important features extracted from DNN trained on data harmonized inside k-fold CV.



(a) *Plot of dotted shapley values: for each feature each instance (each test sample) is represented by a dot, the position indicates a positive or negative contribution to the output, while the color represents the magnitude of the feature, compared with the average value of that feature across all the test dataset.*



(b) *Bar plot of shapley values: for each feature is represented the mean absolute value of all the shap values computed for each instance*

Figure 15.2: *Two different but closely related force plots of shap values regarding the first 20 most important features obtained from data where harmonization procedure is implemented inside the k-fold CV scheme*

15.1 Plots of the most relevant features from DNNs and random forest

Feature importance with DNN

If we plot the first twenty important feature for DNNs, obtained from each pipeline described in section 11, we can notice that some features seem to be persistent through all the four pipelines; we find that: feature 592 and 1521 appear, even if in different order of importance, in all our

four analysis; (We replicated the plot in figure 15.2b putting it again in figure A.1a for a more immediate visual comparison of all the twenty important features for each analysis). If we consider only features common between harmonization in k-fold, harmonization upstream and raw data, two more features appear common: 1513 and 3334. We run this latter analysis excluding the adversarial network to allow comparison with results of the following section 15.1 extracted from random forest.

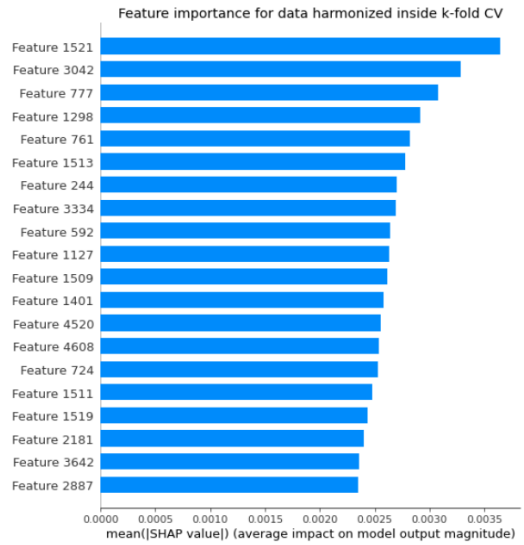
Since each feature is representative of a correlation between two brain areas, we list what are these ares involed:

- Feature 592: correlation between Right Middle Temporal Gyrus (anterior division) and Left Superior Temporal Gyrus (anterior division)
- Feature 1521: correlation between Left Angular Gyrus and Right Middle Temporal Gyrus (posterior division)
- Feature 1513: correlation between Left Angular Gyrus and Right Temporal Pole
- Feature 3334: correlation between Right Parahippocampal Gyrus (posterior division) and Right Accumbens

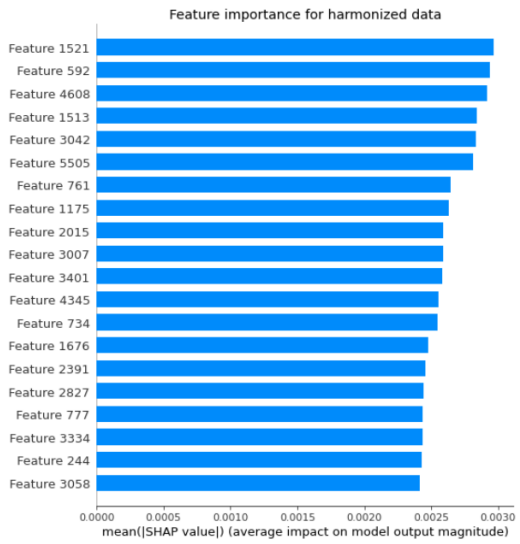
Feature importance with Random Forest

In this section we repeat the same plot we obtained with a dnn in figure A.1, with a different machine learning algorithm: a random forest classifier. We plot the first 20 most important features obtained from the three main analysis of harmonization in k-fold, harmonization upstream, and raw data. Features are extracted an plots are created using the function provided by Scikit-Learn library. Results are shown in figure 15.4. From the three plots we notice that 7 different features are common to the three analysis: feature 710, 1127, 1703, 748, 2735, 1400 and 2811. As we did for DNNs we create a list with information about the brain areas involved in these correlations.

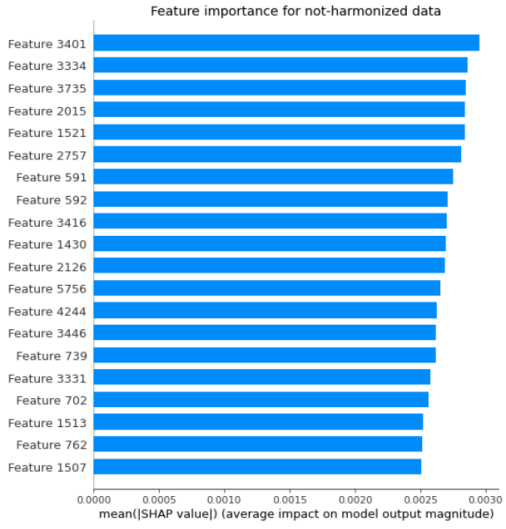
- Feature 710: correlation between Right Middle Temporal Gyrus (temporooccipital part) and Right Thalamus
- Feature 1127: correlation between Left Postcentral Gyrus and Right Postcentral Gyrus



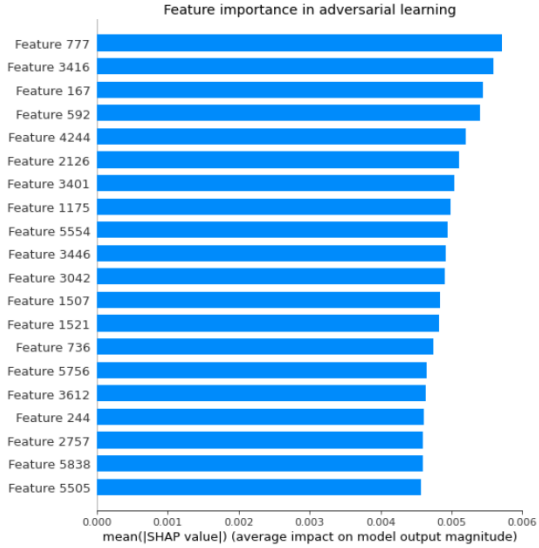
(a) *Bar plot of shap values and related important features extracted with harmonized data, with harmonization procedure implemented inside* ***k-fold*** *CV*



(b) *Bar plot of shap values and related important features extracted with harmonized data, with harmonization procedure implemented* ***upstream****, before the k-fold*

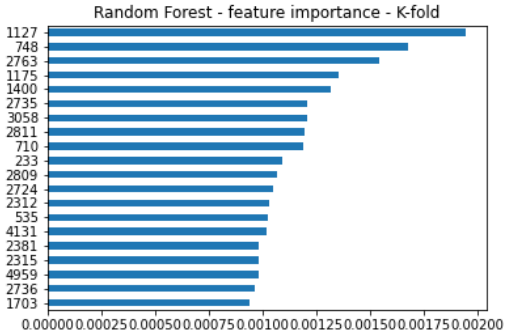


(c) *Bar plot of shap values and related important features extracted from* ***raw****, not-harmonized data*

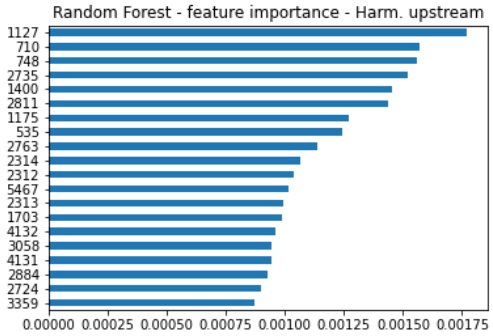


(d) *Bar plot of shap values and related important features extracted from raw data using the* ***Adversarial*** *neural network,*

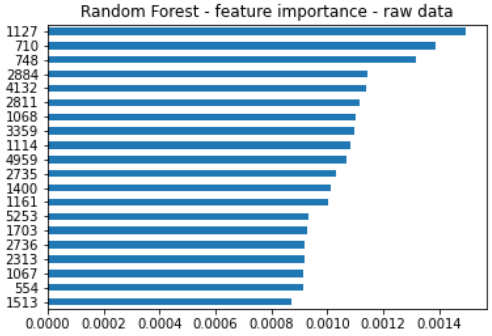
Figure 15.3: *First twenty most important features plotted with a bar plot of mean absolute shap values. Extracted from each of the four analysis we run with DNNs and Adversarial network*



(a) *kfold*



(b) *upstream*



(c) *Non-harmonized*

Figure 15.4: *Feature importance obtained with a Random Forest classifier: results on the entire dataset.*

- Feature 1703: correlation between Right Lateral Occipital Cortex (inferior division) and Right Supramarginal Gyrus (anterior division)
- Feature 748: correlation between Left Middle Temporal Gyrus (temporooccipital part) and Right Thalamus
- Feature 2735: correlation between Right Precuneous Cortex and Right Middle Temporal Gyrus (anterior division)
- Feature 1400: correlation between Left Supramarginal Gyrus (posterior division) and Right Inferior Frontal Gyrus (pars triangularis)
- Feature 2811: correlation between Left Precuneous Cortex and Right Middle Temporal Gyrus (posterior division)

Comments on feature importance

Comparing results obtained using different classifier: a DNN and a random forest classifier, we notice from plots A.1 and 15.4 that there are no common features between the first 20. This is most likely due to the difference in the inner algorithm that guide throughout the decisional procecss. A DNN and a random forest have very different modes of operation. This can lead the two of them to find different patterns within data whose results do not (or just partially) overlap. We are prone to consider the DNNs most reliable than the random forest because of the higher classification perfocmance achieved with that models, but as we can notice from plots, random forest feature importance is able to determine some feature that stand out over the others. The same trend is not visible with feature extracted from the DNNs except for the pipeline of harmonization inside k-fold where it seems that some feature have a greater contribution than others and the descending in feature importance is less smooth. A possible reason for this trend, in accordance of what we said during the discussion of classification results, is that the harmonization procedure, even if does not bring up new information, helps in data cleaning, and remove some noise due to site-distinctive patterns. Harmonization thus helps to obtain a better and a more defined assessment of what the most important features are, in discrimination of controls/ASD.

Even if there are not overlapping features between analysis with DNN and anslysis with random forest, if we take a look at the lsits of brain areas involved in correlation, we can notice that certain brain areas such as the temporal gyri are in common between these two models. This can suggest that even if the feature itself does not determine in the same way the output ot these two models, maybe some brain area represented by features, are recurrent in this kind of discrimination. In this regard, the following analysis aims to study this aspect and determine what are the most important brain areas that allow a discrimination between healthy and ASD patients.

15.2 Cross analysis of important features

So far, we discussed results linked to images just as a rapid and visual comparison of feature importance between DNN and random forest. Now we go deeper to fulfill a more meaningful analysis of what are important feature and what they represent.

Since as explained in section 14.4 we decided to exclude from our analysis the procedure of upstream harmonization because it creates a bias in results, from now on, we focus only on the other pipelines and when we refers to harmonized data, we are referring to data with harmonization procedure implemented inside kfold.

To assess what are the most relevant recurrent feature between these different types of analysis, we choose to limit our analysis to the 1% important features among 5995; to this end we collected the first 60 important features and we limited our analysis to only 5 classification procedures listed below. We limit to these pipelines since they are the most correct and significant, and we can extract non-biased results.

1. Classification of harmonized data using the DNN
2. Classification of harmonized data using the Random Forest classifier
3. Classification of raw data using the DNN
4. Classification of raw data using the Random Forest classifier
5. Classification using raw data with the adversarial neural network

Firstly we checked if and how many common features we find among the first 60 important features between these five classification. But turns out that there are no common features between all these 5 pipelines taking into account only the first 60.

Now, we carry out analysis on subgroups of the previous classification methods and we report the number of common important features among the first 60 most relevant as a number and as percentage. This kind of analysis is useful to assess whether the harmonization procedure significatively changes important features. It is also aimed to quantify the common features between a DNN and a random forest classifier.

- Focusing just on **DNN** we compared pipelines 1. and 3. finding that 22 features (37 %) out of 60 are common between harmonized and raw data.
- Focusing just on **Random Forest** classifier, we compared pipelines 2. and 4. finding that 33 features out of 60 are common (55%) between harmonized and raw data.
- Comparing results on **harmonized data** obtained with **DNN** (pipeline 2.) and **Random Forest** (1.) we obtain 13 common features (22 %)
- Comparing results on **raw data** obtained with **DNN** (pipeline 3.) and **Random Forest** (4.) we obtain 7 common features (12%)
- Comparing results of the **Adversarial network** (5.) with the **harmonized data** with DNN (1.) we obtain 21 common features (35%)
- Comparing results of the **Adversarial network** (5.) with DNN classification of **raw data** (3.) we obtain 28 common features (47%)

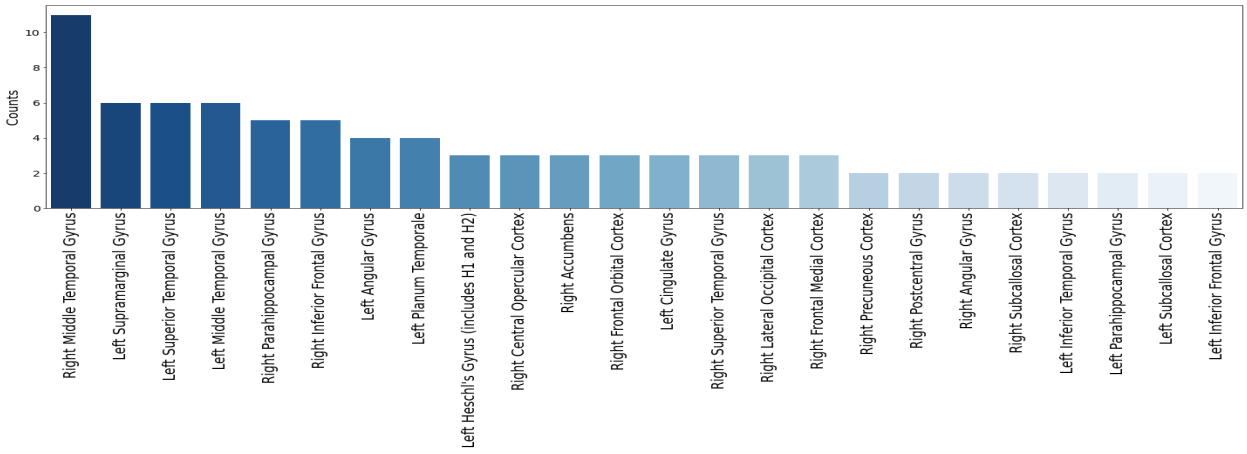
15.3 Brain areas related to important features

As mentioned before, each feature is representative of a correlation between two brain areas, and, since for brain parcellization we used the Harvard-Oxford atlas with 110 ROIs, each brain area is involved in 109 features. In this section we answer to the question we asked before, whether or not some brain areas are recurrent in healthy/ASD discrimination, and what are these areas.

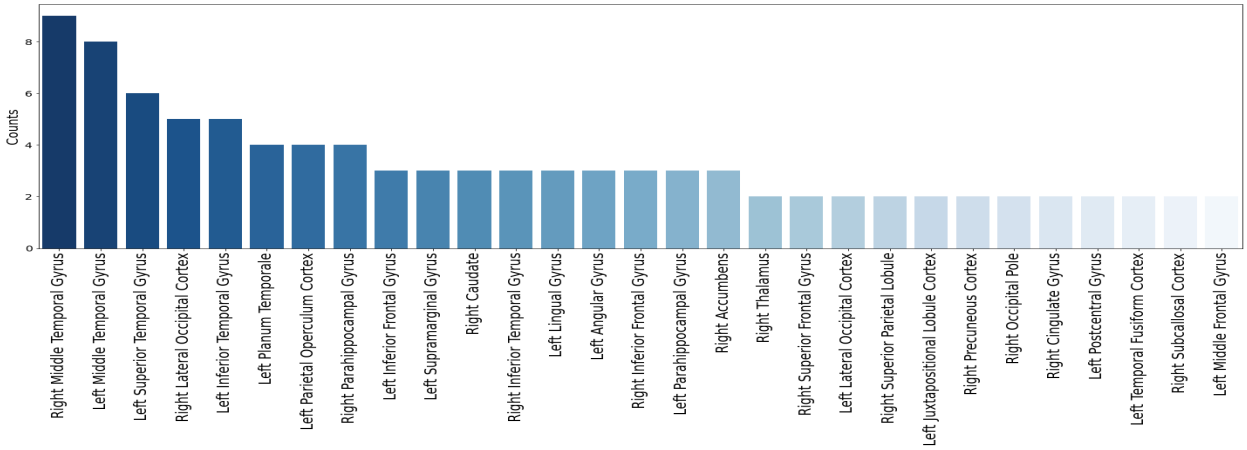
For this purpose, we plotted an histogram to represent the number of occurrences for each brain areas among the first 60 features for each classification procedure, and subsequently we created the histogram of the overall important brain areas pooling all of these results into a single histogram .

Figure 15.5 shows for each pipeline, what are the brain areas recurrent among the 60 most important feature. In figure 15.6 is shown the overall histogram comprising all the results.

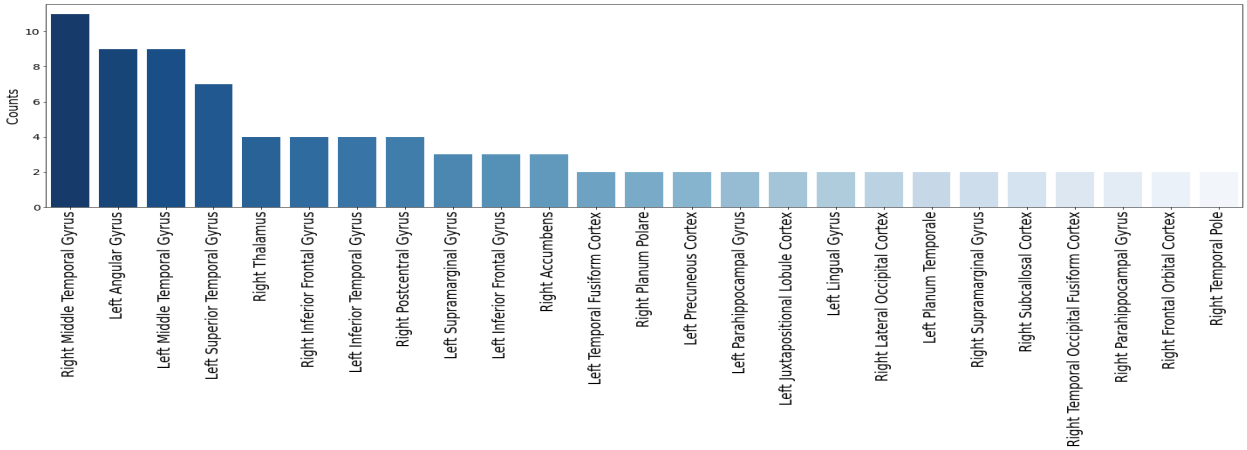
Since it is immediate to notice that there are recurrent brain areas across all the analysis, we are willing to visualize (fig 15.6) the overall histogram putting together results obtained from each



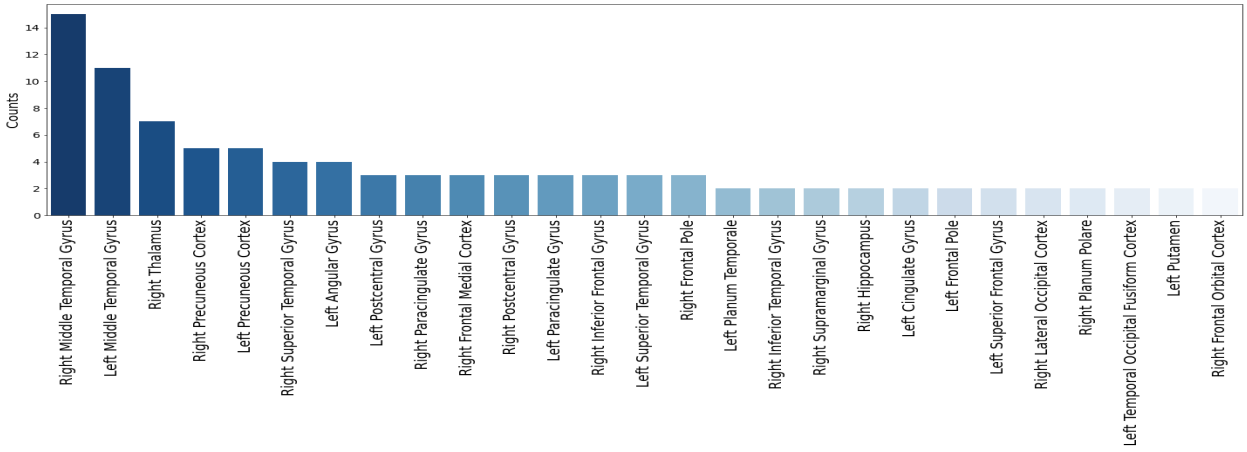
(a) *Important areas from Raw data classified with DNN*



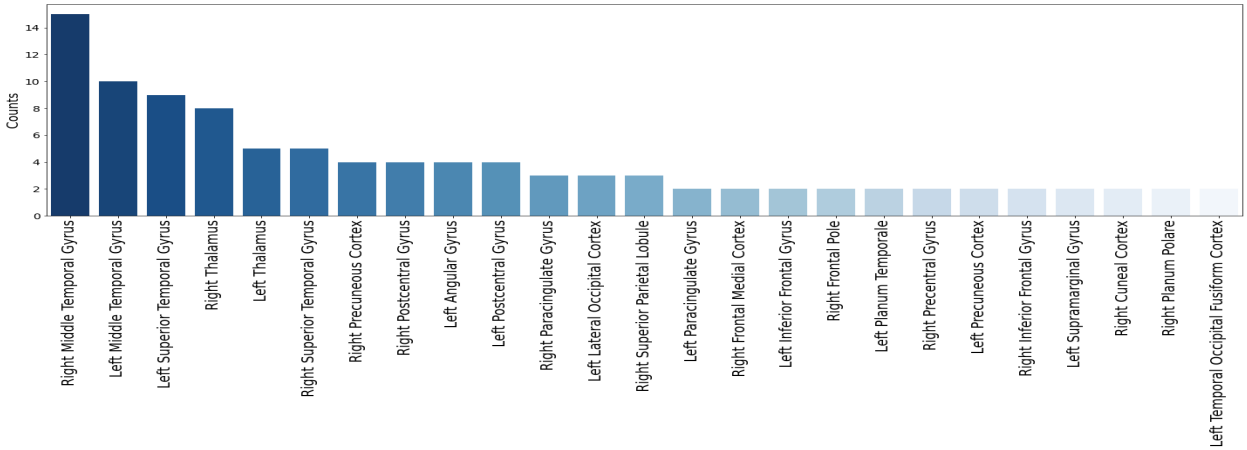
(b) *Important areas from Raw data classified with Adversarial network*



(c) *Important areas from harmonized data classified with DNN*



(d) *Important areas from harmonized data classified with Random Forest*



(e) *Important areas from Raw data classified with Random Forest*

Figure 15.5: *Histograms of important brain areas extracted from the first 60 important features of different classification procedures*

analysis. To this end, as mentioned above, we computed in figure 15.6 the overall histogram of the most recurrent brain areas between these common feature.

15.4 Discussion on common features and important brain areas

Manca discussione su zone collegate ad autismo

Common important features With this analysis to check the common features between different analysis, we can quantify the amount of common features to have a better comparison on how much different harmonization strategies or classificator have an impact on the assessment of feature importance. Looking at the results listed in section 15.2 we can assert that a DNN and a random forest definitely have different way to use features to make prediction and to assess their importance. We find that just a low percentage of features are common between a DNN and a random forest both with raw and harmonized data. This has a non-negligible impact when choosing the best classificator for any classification task. We should keep in mind that there is not the best classifier in an absolute sense, but each classifier is able to find different patternt among data and we should pick the best only in relation to a specific dataset.

We can also state that harmonization procedure changes, especially in a DNN the first important features, as we pointed out when discussing the plots of feature importance, harmonization is able to remove noise from data and reveals in a cleaner way what feature are really important for the network, without the contribution of site-related noise. It is possible that the number of feature in a random forest between harmonized and non harmonized data is bigger because random forest is a simpler algorithm. In fact a DNN is able to learn more complicated pattern, and in doing so, it is more sensible to noise and to the discovery of more subtile pattern that can drive to this difference in feature ranking. This finer searching for patterns, though, is what lead a DNN to achieve better classification performances with respect to a random forest. Thus, harmonization changes the most important features, but we believe that this chance happen in a positive way, since feature are less affected by noise and results are more reliable.

This change resulting from a better definition of feature due to harmonization, also appears when

comparing results with the adversarial network. In fact we observe that The number of common feature between adversarial network and raw data is similar to the number between harmonized and raw data determined using the DNN. The resaon for this lies in the mechanism of the adversarial network: it results in a reduction of some site-related noise thanks to the adversarial branch, but at the same time, the introduction of some confusion due to the flawed definition of the correct site.

Even if between raw data and adversarial network we should have obtained a result similar to ras data and harmonized data ($\approx 37\%$), we obtain a greater similarity of feature importance ($\approx 47\%$). This means that data are modified and hamronized according to some inner processes, but in a really different way than the analytical harmonization.

Important areas

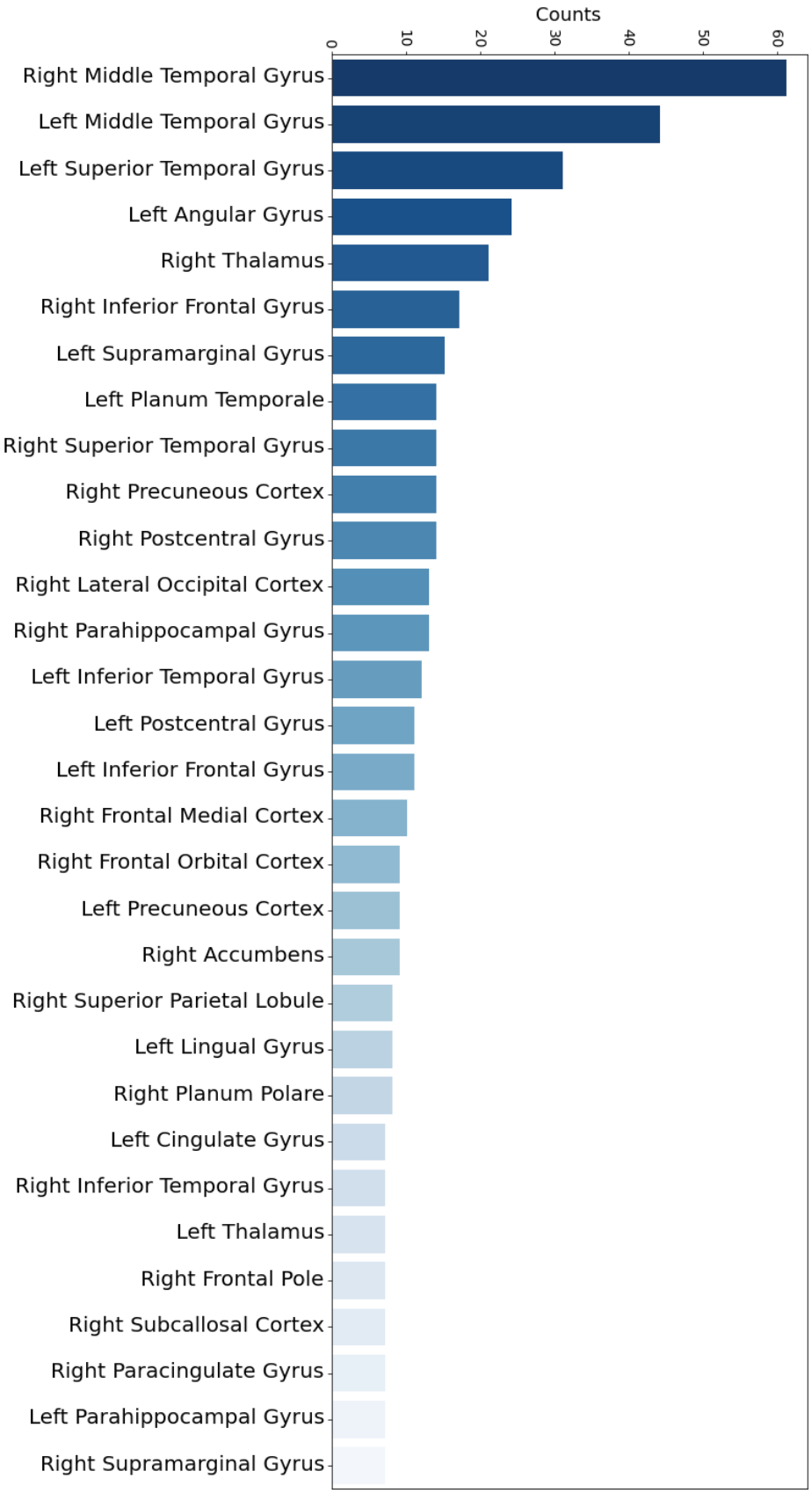


Figure 15.6: *Histogram of the most important brain areas putting together results obtained from all the five analysis listed above.*

Appendix A

Other classification results

Dataset	Harmon K-fold	Harmon. upstream	No harmon.	Adversarial
AB I±II	60±1	71±3	63±2	60±1
AB I	61±2	69±3	62±2	61±2
AB 2	56±3	69±4	60±4	59±4
AB I ± II - eye open	63±2	68±3	64±2	63±3
AB I open eye	63±5	74±3	71±4	72±2
AB II eye open	63±5	70±3	63±3	62±3
AB I ± II AGE=all, eye=all	65±3	71±2	66±2	63±2

Table A.1: *Classification scores using only out-of-phase wavelet coefficients*

Dataset	Harmon K-fold	Harmon. upstream	No harmon.	Adversarial
AB I±II	67± 3	70± 3	66 ±3	66 ±3
AB I	70± 3	72 ±2	68 ±3	68± 3
AB 2	61± 4	65± 4	64 ±5	65 ±4
AB I ± II - eye open	70± 1	73± 2	71 ±1	68 ±2
AB I open eye	71± 4	74 ±3	71 ±3	68 ±2
AB II eye open	62 ±4	66 ±3	68 ±3	64± 5

Table A.2: *Classifiaction scores using (W_in - W_out) wavelet coefficients coefficients*

Structure	k-fold	upstream	no harmonization
3-2-1	60±2	63±3	60±4
8-8-1	64±1	65±4	64±5
64-8-1	69±1	72±2	68±1
64-32-8-1	68±1	72±2	69±1
128-8-1	70±2	71±3	69±2
128-64-1	70±1	72±3	69±2
264-8-1	70±2	73±3	70±2
512-8-1	71±2	73±2	70±2
1024-8-1	70±1	74±2	71±2
1024-32-1	70±2	74±2	71±3

Table A.3: *Different model structures and related performances for the three main analysis we carried out during this entire work. Colored the structure we choose to employ through all our analysis*

A.1 Feature importance results on ABIDE I open eye dataset

The same tests we did to assess feature importance on ABIDE I + II dataset, we carried out on ABIDE I with only patient with open eye. Here in figure **??** we report the results obtained from shap, related to the first twenty most relevant features. We can notice at a first glance that there are less feature common to all the four analysis: There are no common features among them that we can find across all these four analysis, if we limit our study on the first twenty. We have to search among the first thirty features to find something in common, which is still a good procedure since we are dealing with 5995 features and the first 30 are just the 0.5% of them.

We find that, among the first 30 features, only feature 762 is common to the four analysis, while if we exclude the adversarial network we add feature 1417 and 748

- Feature 762: correlation between Left Middle Temporal Gyrus (temporooccipital part) and Left Middle Frontal Gyrus
- Feature 1417: correlation between Left Supramarginal Gyrus (posterior division) and Left Middle Temporal Gyrus (temporooccipital part)
- Feature 748: correlation between Left Middle Temporal Gyrus (temporooccipital part) and Right Thalamus (nan)

With a random forest we obtain more coherence between important features: searching among the first 20 features, we find that are common to the 3 pipelines: 2690, 1127, 2313, 2314, 748, 2712, 2735, 2582, 1400

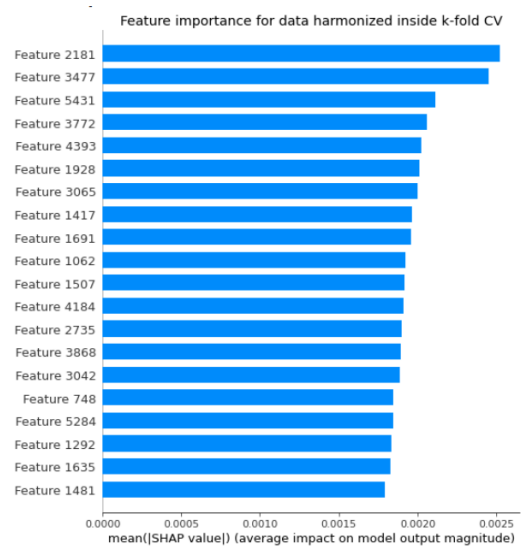
- Feature 2690: correlation between Left Cingulate Gyrus (posterior division) and Right Frontal Medial Cortex (nan)
- Feature 1127: correlation between Left Postcentral Gyrus (nan) and Right Postcentral Gyrus (nan)
- Feature 2313: correlation between Right Paracingulate Gyrus (nan) and Left Middle Temporal Gyrus (anterior division)

- Feature 2314: correlation between Right Paracingulate Gyrus (nan) and Right Middle Temporal Gyrus (posterior division)
- Feature 748: correlation between Left Middle Temporal Gyrus (temporooccipital part) and Right Thalamus (nan)
- Feature 2712: correlation between Right Precuneous Cortex (nan) and Right Hippocampus (nan)
- Feature 2735: correlation between Right Precuneous Cortex (nan) and Right Middle Temporal Gyrus (anterior division)
- Feature 2582: correlation between Right Cingulate Gyrus (posterior division) and Right Precentral Gyrus (nan)
- Feature 1400: correlation between Left Supramarginal Gyrus (posterior division) and Right Inferior Frontal Gyrus (pars triangularis)

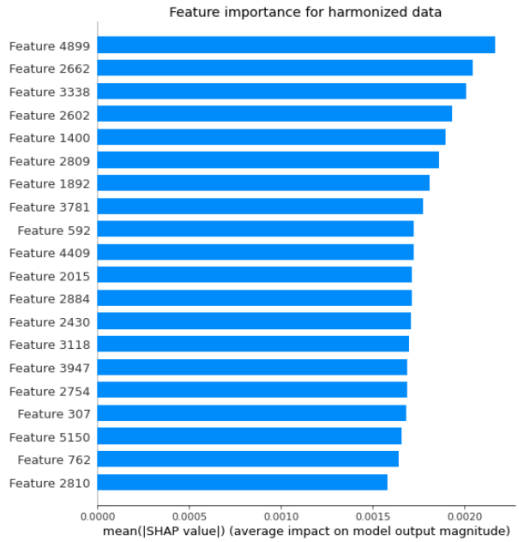
The same cross analysis we did on the entire dataset:

- Focusing on random forest classifier, we compared important features between 2 and 4 and found that 34 features out of 60 are common (57%).
- Focusing on DNN we compared pipelines 1 and 3 finding that 19 features (32 %) out of 60 are common.
- Comparing the same analysis pipeline with DNN and Random Forest we find that with harmonized data: procedure 2 and 1 there are 11 common features (18 %)
- Comparing results on raw data obtained with DNN 3 and Random Forest 4 we obtain 13 common features (22%)
- Comparing the adversarial results 5 with the harmonized data with DNN 1 we obtain 19 common features (32%)

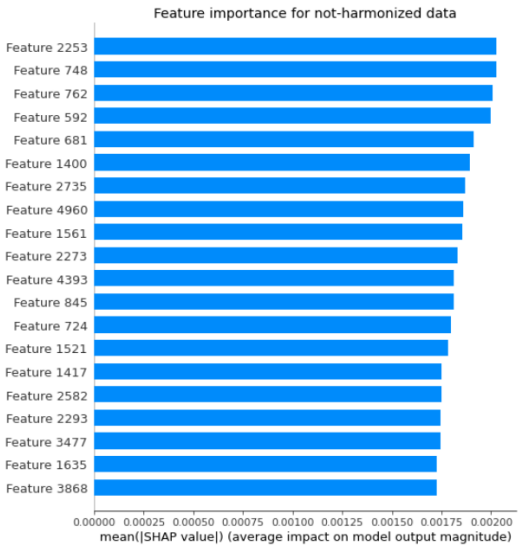
- Comparing the adversarial results 5 with DNN classification of raw data 3 we obtain 23 common features (38%)



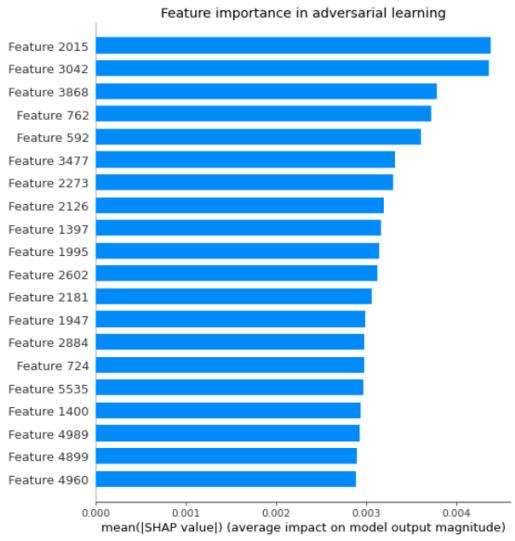
(a) *Bar plot of shap values and related important features extracted with harmonized data, with harmonization procedure implemented inside* ***k-fold*** *CV*



(b) *Bar plot of shap values and related important features extracted with harmonized data, with harmonization procedure implemented* ***up-stream****, before the k-fold*

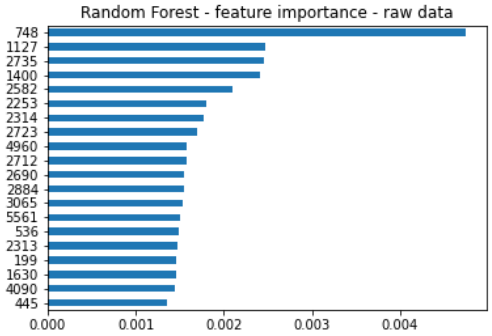


(c) *Bar plot of shap values and related important features extracted from* ***raw****, not-harmonized data*

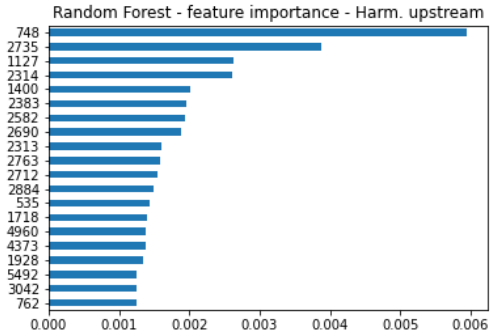


(d) *Bar plot of shap values and related important features extracted from raw data using the* ***Adversarial*** *neural network,*

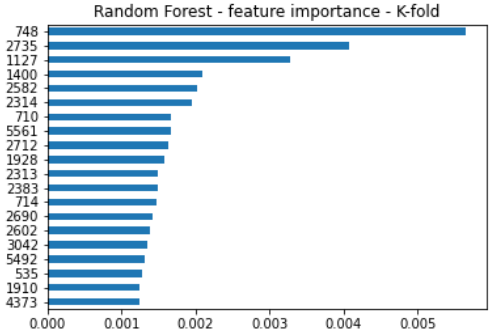
Figure A.1: *Results obtained on ABIDE I only open eye dataset. First twenty most important features plotted with a bar plot of mean absolute shap values. Extracted from each of the four analysis we run with DNNs and Adversarial network.*



(a) Non-harmonized

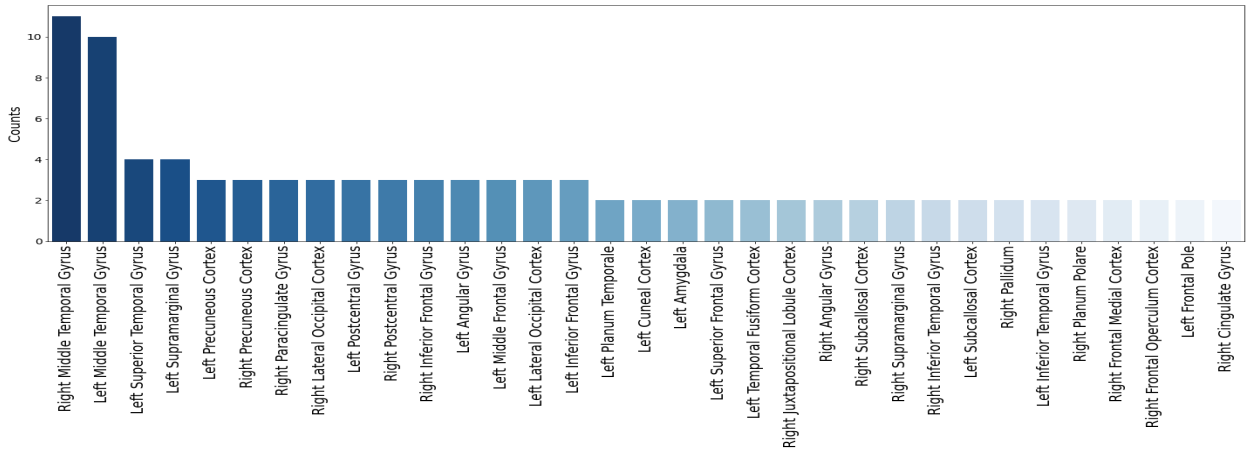


(b) upstream

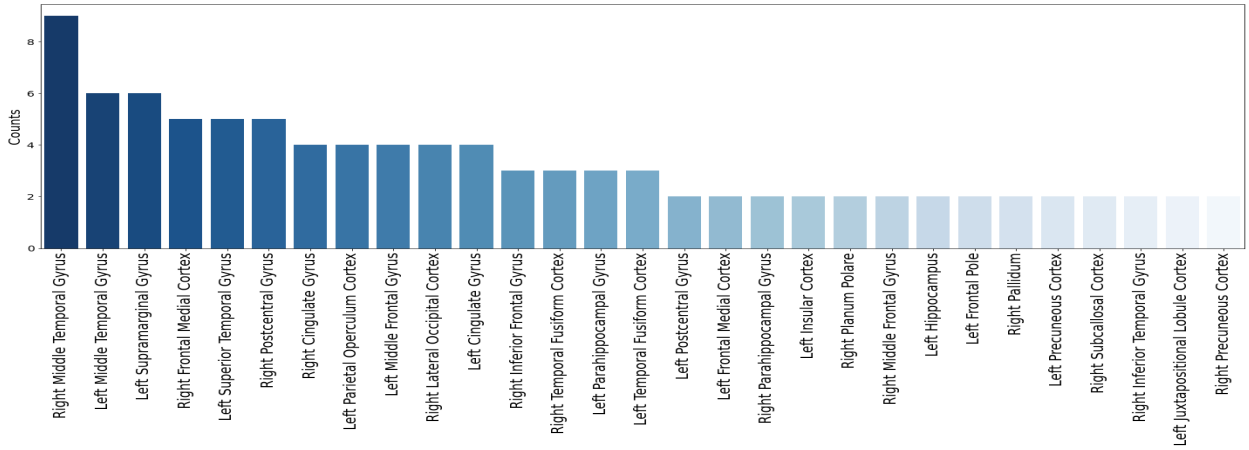


(c) kfold

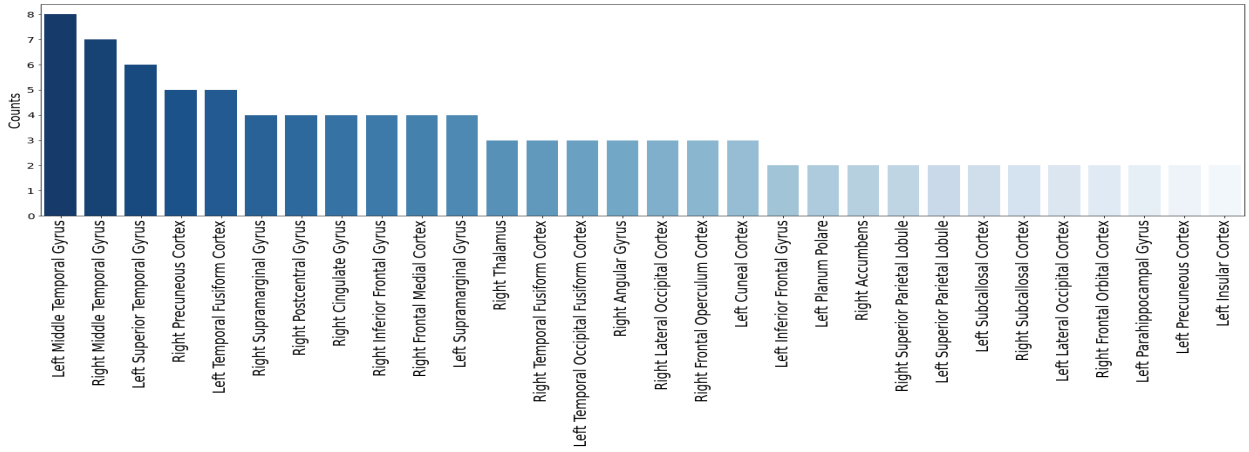
Figure A.2: *Feature importance obtained from a Random Forest classifier: results on ABIDE I only open eyes.*



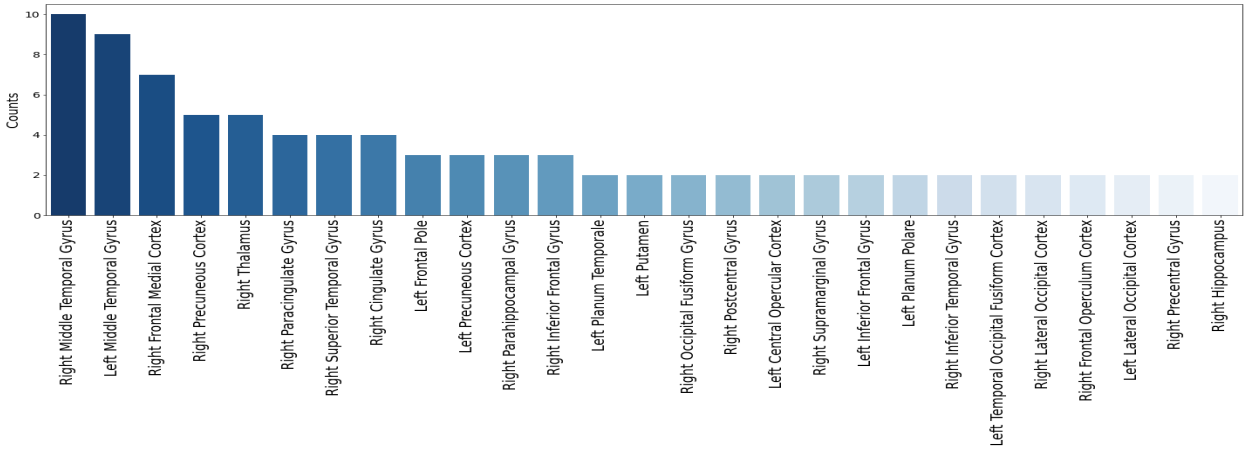
(a) *Important areas from Raw data classified with DNN*



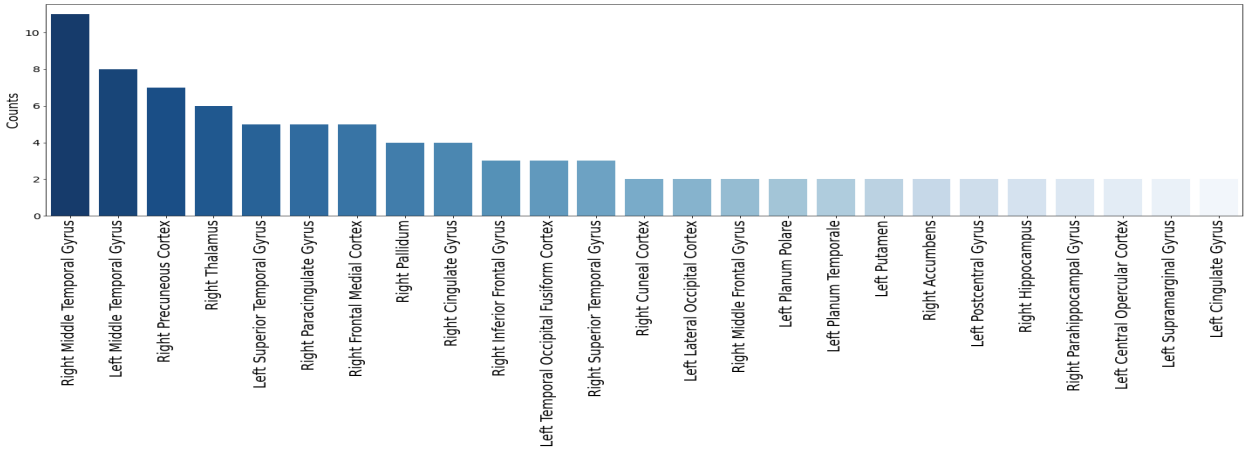
(b) *Important areas from Raw data classified with Adversarial network*



(c) *Important areas from harmonized data classified with DNN*



(d) *Important areas from harmonized data classified with Random Forest*



(e) *Important areas from Raw data classified with Random Forest*

Figure A.3: *Histograms of important brain areas extracted from the first 60 important features of different classification procedures*

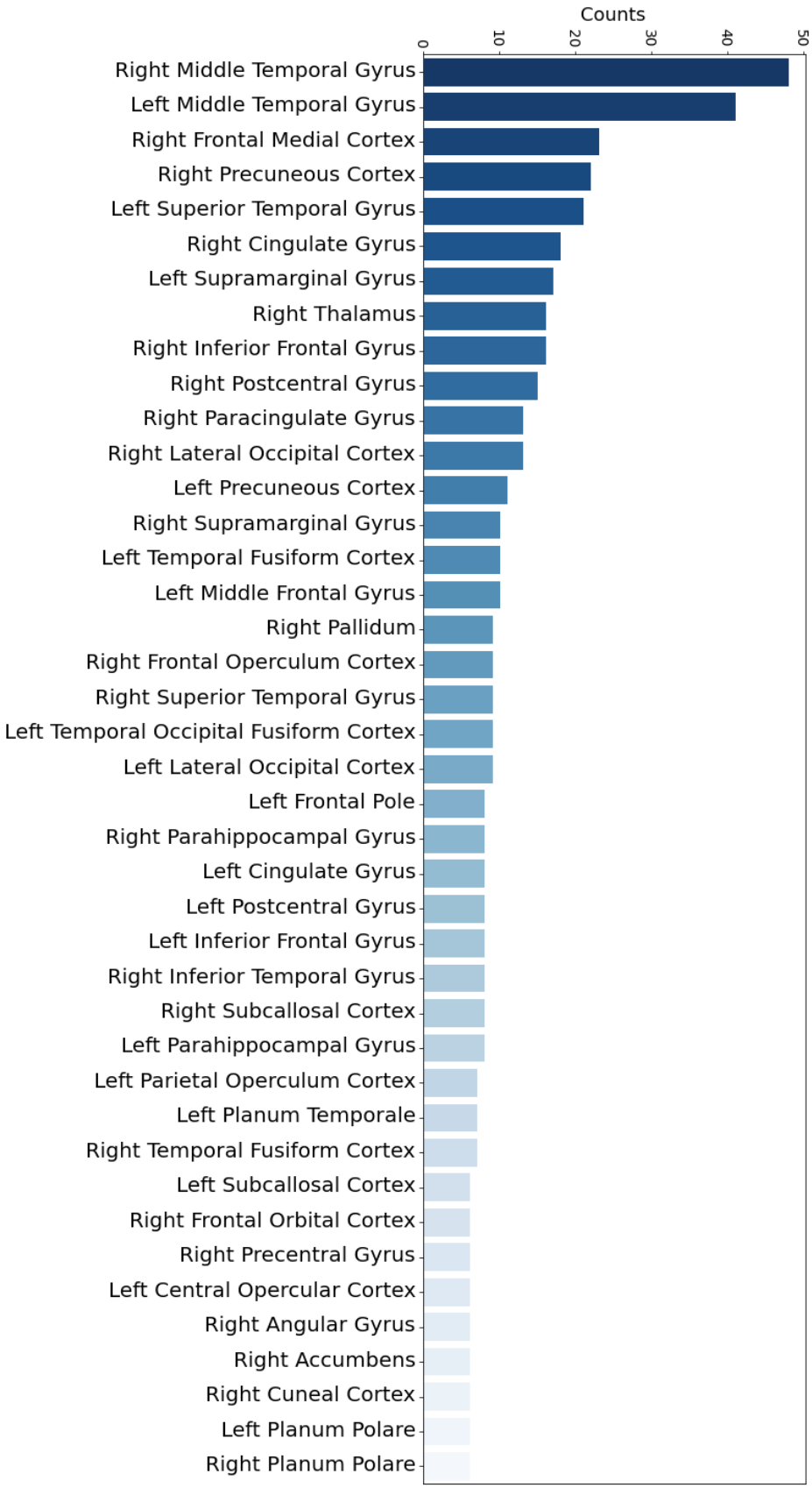


Figure A.4: *Histogram of the most important brain areas putting together results obtained from all the five analysis listed above.*

Bibliography

[1] Encarnación Algaba, Vito Fragnelli, and Joaquín Sánchez-Soriano. Handbook of the shapley value.

[2] Hartmann Atm. Lecture 11: White and red noise.

[3] Luca Baldini, Niccolò Di Lalla, Massimiliano Razzano, and Carmelo Sgrò. Introduzione all'analisi dei dati per il laboratorio di fisica.

[4] Antoine Bernas, Albert P. Aldenkamp, and Svitlana Zinger. Wavelet coherence-based classifier: A resting-state functional mri study on neurodynamics in adolescents with high-functioning autism. *Computer Methods and Programs in Biomedicine*, 154:143–151, 2 2018.

[5] Janine Bijsterbosch, Stephen Smith, and Christian Beckmann. Introduction to resting state fmri functional connectivity oxford neuroimaging primers.

[6] Matthew Brett. The mni brain and the talairach atlas, 2002.

[7] Matthew Brett, Ingrid S. Johnsrude, and Adrian M. Owen. The problem of functional localization in the human brain. *Nature Reviews Neuroscience*, 3:243–249, 3 2002.

[8] Robert W. Brown, E. Mark Haacke, Yu-Chung N. Cheng, Michael R. Thompson, and Ramesh Venkatesan. *Magnetic Resonance Imaging*. John Wiley and Sons Ltd, 4 2014.

[9] Craddock Cameron, Benhajali Yassine, Chu Carlton, Chouinard Francois, Evans Alan, Jakab András, Khundrakpam Budhachandra, Lewis John, Li Qingyang, Milham Michael, Yan Chaogan, and Bellec Pierre. The neuro bureau preprocessing initiative: open sharing of preprocessed neuroimaging data and derivatives. *Frontiers in Neuroinformatics*, 7, 2013.

[10] Hugh Chen, Scott Lundberg, and Su-In Lee. Explaining models by propagating shapley values of local components, 2019.

[11] C. Edward Coffey, Joseph F. Lucke, Judith A. Saxton, Graham Ratcliff, Lori Jo Unitas, Brenda Billig, and R. Nick Bryan. Sex differences in brain aging. *Archives of Neurology*, 55:169, 2 1998.

[12] Víctor Costumero, Elisenda Bueichekú, Jesús Adrián-Ventura, and César Ávila. Opening or closing eyes at rest modulates the functional connectivity of v1 with default and salience networks. *Scientific Reports*, 10:9137, 12 2020.

[13] Ann Le Couteur, Gyles Haden, Donna Hammal, and Helen McConachie. Diagnosing autism spectrum disorders in pre-school children using two standardised assessment instruments: The adi-r and the ados. *Journal of Autism and Developmental Disorders*, 38:362–372, 2 2008.

[14] R. Cameron Craddock, G.Andrew James, Paul E. Holtzheimer, Xiaoping P. Hu, and Helen S. Mayberg. A whole brain fmri atlas generated via spatially constrained spectral clustering. *Human Brain Mapping*, 33:1914–1928, 8 2012.

[15] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1 1992.

[16] Rahul S. Desikan, Florent Ségonne, Bruce Fischl, Brian T. Quinn, Bradford C. Dickerson, Deborah Blacker, Randy L. Buckner, Anders M. Dale, R. Paul Maguire, Bradley T. Hyman, Marilyn S. Albert, and Ronald J. Killiany. An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest. *NeuroImage*, 31:968–980, 7 2006.

[17] Pedro Domingos. The role of occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3:409–425, 1999.

[18] Pol Ferrando. Understanding how lime explains predictions, 2018.

[19] Isidoro Ferrante. *Elaborazione dei segnali per la fisica*. Pisa university press, 2015.

[20] Jean-Philippe Fortin, Nicholas Cullen, Yvette I. Sheline, Warren D. Taylor, Irem Aselcioglu, Philip A. Cook, Phil Adams, Crystal Cooper, Maurizio Fava, Patrick J. McGrath, Melvin McInnis, Mary L. Phillips, Madhukar H. Trivedi, Myrna M. Weissman, and Russell T. Shinohara. Harmonization of cortical thickness measurements across scanners and sites. *NeuroImage*, 167:104–120, 2 2018.

[21] Jean-Philippe Fortin, Drew Parker, Birkan Tunç, Takanori Watanabe, Mark A. Elliott, Kosha Ruparel, David R. Roalf, Theodore D. Satterthwaite, Ruben C. Gur, Raquel E. Gur, Robert T. Schultz, Ragini Verma, and Russell T. Shinohara. Harmonization of multi-site diffusion tensor imaging data. *NeuroImage*, 161:149–170, 11 2017.

[22] C. M. Freitag. The genetics of autistic disorders and its clinical relevance: A review of the literature, 1 2007.

[23] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. 2015.

[24] A Grinsted, J C Moore, and S Jevrejeva. Application of the cross wavelet transform and wavelet coherence to geophysical time series nonlinear processes in geophysics application of the cross wavelet transform and wavelet coherence to geophysical time series, 2004.

[25] SAMUEL B. GUZE. Diagnostic and statistical manual of mental disorders, 4th ed. (dsm-iv). *American Journal of Psychiatry*, 152:1228–1228, 8 1995.

[26] Scott A. Huettel, Allen W. Song, and Gregory McCarthy. *Functional Magnetic Resonance Imaging*. 2009.

[27] Madhura Ingalhalikar, Sumeet Shinde, Arnav Karmarkar, Archith Rajan, D. Rangaprakash, and Gopikrishna Deshpande. Functional connectivity-based prediction of autism on site harmonized abide dataset. *IEEE Transactions on Biomedical Engineering*, 68:3628–3637, 12 2021.

[28] Mark Jenkinson and Michael Chappell. Introduction to neuroimaging analysis, 2018.

[29] W. Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical bayes methods. *Biostatistics*, 8:118–127, 1 2007.

[30] Paul C. Liu. Wavelet spectrum analysis and ocean wind waves, 1994.

[31] Angela Lombardi, Nicola Amoroso, Domenico Diacono, Alfonso Monaco, Sabina Tangaro, and Roberto Bellotti. Extensive evaluation of morphological statistical harmonization for brain age prediction. *Brain Sciences*, 10:364, 6 2020.

[32] Karsten Müller, Gabriele Lohmann, Jane Neumann, Maren Grigutsch, Toralf Mildner, and D. Yves Von Cramon. Investigating the wavelet coherence phase of the bold signal. *Journal of Magnetic Resonance Imaging*, 20:145–152, 7 2004.

[33] Sarah Guido Andreas C. Müller. *Introduction to Machine Learning with Python.* O'Reilly Media, Inc.

[34] John F. Nash. Equilibrium points in ¡i¿n¡/i¿ -person games. *Proceedings of the National Academy of Sciences*, 36:48–49, 1 1950.

[35] World Health Organization. The icd-10 classification of mental and behavioural disorders : diagnostic criteria for research, 1993.

[36] Sally Ozonoff, Beth L. Goodlin-Jones, and Marjorie Solomon. Evidence-based assessment of autism spectrum disorders in children and adolescents, 2005.

[37] F Pedregosa, G Varoquaux, A Gramfort, B Michel V., Thirion, O Grisel, M Blondel, R Prettenhofer P., Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[38] Donald B. Percival and Andrew T. Walden. *Wavelet Methods for Time SeriesAnalysis.* Cambridge University Press, 2000.

[39] Raymond Pomponio, Guray Erus, Mohamad Habes, Jimit Doshi, Dhivya Srinivasan, Elizabeth Mamourian, Vishnu Bashyam, Ilya M. Nasrallah, Theodore D. Satterthwaite, Yong Fan,

Lenore J. Launer, Colin L. Masters, Paul Maruff, Chuanjun Zhuo, Henry Völzke, Sterling C. Johnson, Jurgen Fripp, Nikolaos Koutsouleris, Daniel H. Wolf, Raquel Gur, Ruben Gur, John Morris, Marilyn S. Albert, Hans J. Grabe, Susan M. Resnick, R. Nick Bryan, David A. Wolk, Russell T. Shinohara, Haochang Shou, and Christos Davatzikos. Harmonization of large mri datasets for the analysis of brain imaging patterns throughout the lifespan. *NeuroImage*, 208, 3 2020.

[40] Isabelle Rapin and Roberto F. Tuchman. Autism: Definition, neurobiology, screening, diagnosis. *Pediatric Clinics of North America*, 55:1129–1146, 10 2008.

[41] Vahid Mirjalili Sebastian Raschka. *Python Machine Learning*. Packt Publishing, 2019.

[42] Helen V. Ratajczak. Theoretical aspects of autism: Causes—a review. *Journal of Immunotoxicology*, 8:68–79, 3 2011.

[43] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. volume 13-17-August-2016, pages 1135–1144. Association for Computing Machinery, 8 2016.

[44] Kaitlin Riddle, Carissa J. Cascio, and Neil D. Woodward. Brain structure in autism: a voxel-based morphometry analysis of the autism brain imaging database exchange (abide). *Brain Imaging and Behavior*, 11:541–551, 4 2017.

[45] Diana L Robins, Deborah Fein, and Marianne Barton. Modified checklist for autism in toddlers, revised, with follow-up (m-chat-r/f) tm, 2009.

[46] Don Ross. Game theory, 2021.

[47] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. 4 2017.

[48] Giovanna Spera, Alessandra Retico, Paolo Bosco, Elisa Ferrari, Letizia Palumbo, Piernicola Oliva, Filippo Muratori, and Sara Calderoni. Evaluation of altered functional connections in male children with autism spectrum disorders on multiple-site data optimized with machine learning. *Frontiers in Psychiatry*, 10, 9 2019.

[49] Kaustubh Supekar, Lucina Q. Uddin, Amirah Khouzam, Jennifer Phillips, William D. Gaillard, Lauren E. Kenworthy, Benjamin E. Yerys, Chandan J. Vaidya, and Vinod Menon. Brain hyperconnectivity in children with autism and its links to social deficits. *Cell Reports*, 5:738–747, 11 2013.

[50] Christopher Torrence and Gilbert P. Compo. A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society*, 79:61–78, 1 1998.

[51] Christopher Torrence and Peter J. Webster. Interdecadal changes in the enso–monsoon system. *Journal of Climate*, 12:2679–2690, 8 1999.

[52] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, and M. Joliot. Automated anatomical labeling of activations in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain. *NeuroImage*, 15:273–289, 1 2002.

[53] J.C. van den Berg. *Wavelets in Physics*. Cambridge University Press, 8 1999.

[54] Rick Wicklin. Fisher's transformation of the correlation coefficient, 2017.

[55] Xin Yang, Paul T., and Ning Zhang. A deep neural network study of the abide repository on autism spectrum classification. *International Journal of Advanced Computer Science and Applications*, 11, 2020.

[56] H. P. Young. Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14:65–72, 6 1985.

[57] Chao Zhang, Nathan D. Cahill, Mohammad R. Arbabshirani, Tonya White, Stefi A. Baum, and Andrew M. Michael. Sex and age effects of functional connectivity in early adulthood. *Brain Connectivity*, 6:700–713, 11 2016.