Dipartimento di Informatica, Bioingegneria,
Robotica ed Ingegneria dei Sistemi

Challenges in biomedical data science: data-driven solutions to clinical questions

by

Samuele Fiorini

Theses Series **DIBRIS-TH-2018-00**

DIBRIS, Università di Genova
Via Opera Pia, 13 16145 Genova, Italy http://www.dibris.unige.it/

Università degli Studi di Genova

Dipartimento di Informatica, Bioingegneria,

Robotica ed Ingegneria dei Sistemi

Ph.D. Thesis in Computer Science and Systems Engineering
Computer Science Curriculum

Challenges in biomedical data science: data-driven solutions to clinical questions

by

Samuele Fiorini

May, 2018

Dottorato di Ricerca in Informatica ed Ingegneria dei Sistemi
Indirizzo Informatica
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università degli Studi di Genova

DIBRIS, Università degli Studi di Genova
Via Opera Pia, 13
I-16145 Genova, Italy
http://www.dibris.unige.it/

Ph.D. Thesis in Computer Science and Systems Engineering
Computer Science Curriculum
(S.S.D. INF/01)

Submitted by Samuele Fiorini
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università degli Studi di Genova
samuele.fiorini@dibris.unige.it

Date of submission: May 10, 2018

Title: Challenges in biomedical data science.

Advisor: Annalisa Barla
Dipartimento di Informatica, Bioingegneria, Robotica ed Ingegneria dei Sistemi
Università degli Studi di Genova
annalisa.barla@unige.it

Ext. Reviewers:
Federico Girosi
Centre For Health Research
Western Sydney University
f.girosi@westernsydney.edu.au

Giorgio Valentini
Dipartimento di Informatica
Università degli Studi di Milano
valentini@di.unimi.it

Abstract

Data are influencing every aspect of our lives, from our work activities, to our spare time and even to our health. In this regard, medical diagnosis and treatments are often supported by quantitative measures and observations, such as laboratory tests, medical imaging or genetic analysis. In medicine, as well as in several other scientific domains, the amount of data involved in each decision-making process has become overwhelming. The complexity of the phenomena under investigation and the scale of modern data collections has long superseded human analysis and insights potential. Therefore, a new scientific branch that simultaneously addresses statistical and computational challenges is rapidly emerging, and it is known as *data science*.

Data science is the evolving cross-disciplinary field that, borrowing concepts from several scientific areas, aims at devising data-driven decision making strategies for real-world problems. Data science differs from classical applied statistics as it generally combines mathematical background with advanced computer science and thorough domain knowledge. Following the data science philosophy, it is possible to ask the right questions to the data and to find statistically sound answers in a reasonable amount of time.

Machine learning can be seen as one of the main components of data science. The aim of machine learning is to devise algorithms that can recognize and exploit hidden patterns in some set of data in order to formulate an accurate prediction strategy that holds for current and future data as well. Thanks to machine learning it is now possible to achieve automatic solutions to complex tasks with little human supervision. As of today, machine learning is the workhorse of data science.

This thesis revolves around the application of machine learning and data science concepts to solve biomedical and clinical challenges. In particular, after a preliminary overview of the main data science and machine learning concepts and techniques, we will see the importance of exploratory data analysis and how it can be easily performed on any structured input dataset. Moreover, we will see that, with sparsity-enforcing linear models, it is possible to predict the age of an individual from a set of molecular biomarkers measured from peripheral blood. Furthermore, we will present a nonlinear temporal model that accurately predicts the disease evolution in multiple sclerosis patients from a set of inexpensive and patient-friendly measures repeated in time. Finally, we will see how with a continuous glucose monitor and a kernel machine it is possible to accurately forecast the glycemic level of type 1 and type 2 diabetic patients, in order to improve their treatment.

With the final aim of devising actionable solutions, that are ready to be applied in clinical contexts, the predictive performance of the data-driven models proposed throughout the chapters of this thesis is rigorously assessed exploiting bias-aware cross-validation schemes.

Contents

1 Introduction

The influence of data pervades our everyday lives. We are literally surrounded by data acquisition and processing devices that continuously interact with us. The watches at our wrists, the phones in our pockets, the house where we live and even the cars we drive, everything is equipped with devices that can automatically collect and process information in order to make data-driven suggestions affecting our daily routines. This is so pervasive as it often leads to effective, efficient and actionable solutions to real-world problems.

It sounds remarkable already, but there is more. Data collection and processing are not only influencing our daily actions, but also every recent discovery in almost every scientific domains, such as computer science, physics, chemistry, biomedicine and so on.

In these fields, data are so valuable, that it is rather common to achieve different discoveries from the same data collection. This lead to the development of a new paradigm that can be expressed by the motto: *collect first, ask questions later*.

However, this comes at a price. Managing and maintaining infrastructure for data-intensive applications is expensive in terms of economical and human resources employed. In the last few years, the amount of generated data rapidly became overwhelming and it superseded human analysis and insights potential.

Biomedical data are prototypical in this sense. In this field, almost every medical diagnosis is currently supported by quantitative observations. When a clinician is asked to analyze the medical records of a patient, he/she needs to deal with a large number of highly heterogeneous measures that can be hard to understand as they can be missing or incomplete and their interaction can be circumstantial or unknown.

In these circumstances, classical model, that are only driven by prior knowledge of the problem, fall short as they may not explain some relevant part of the phenomenon under investigation.

Nowadays, biomedical data are often analyzed with data-driven models. These models are designed to automatically infer structure and relationships hidden in the data and to use them to predict some target measure. For instance, models of this class can be used to assign a patient to a given class (*e.g.* case/control, or the presence of some phenotype) given a number of blood or genetic measures, or they can predict some future disease-related event from historical medical records.

In this thesis, we will see an overview of the data-driven solutions most commonly adopted to solve biomedical challenges. Theoretical notions with practical examples and real case studies will be presented. Great effort will be devoted toward the presentation of statistics and engineering concepts in a unified fashion. The description of each method will balance the trade-off between providing an intuitive grasp of its behavior and its mathematical foundation, presented in a more rigorous way.

1.1 Outline

This thesis is divided in two parts. Part I presents a thorough description of the multi-disciplinary prerequisites that are relevant for the comprehension of Part II, which describes the original contributions of the work.

Part I is organized as follows: Chapter 2 introduces the concept of *data science* and its declination toward life science and biomedical studies. In this chapter, the major challenges of the field are presented along with several examples of the most common clinical/biological questions and their translation to data analysis tasks (Section 2.1).

Chapter 3 summarizes basic notation and definitions adopted throughout the thesis (Section 1.2) and presents an overview of the statistical and technological tools that are mostly relevant for this work. In particular, this chapter defines the concept of *machine learning* from a general perspective and provides rigorous description of a selection of supervised and unsupervised learning strategies (Sections 3.1 and 3.3). This chapter also defines the concept of variable/feature selection (Section 3.2) and introduces the most relevant model selection and evaluation strategies (Section 3.4). At the end of this chapter, hints on the computational requirements and implementation strategies are presented (Section 3.5).

Part II describes the contribution of this work which consisted in the process of devising data-driven strategies to tackle a number of biological data science challenges coming from real-world clinical environments. For each task, this part shows how the previously introduced tools can be exploited in order to develop statistically sound models that are capable of providing insightful answers to different clinical questions.

Part II is organized as follows: Chapter 4 introduces ADENINE, an open-source Python framework for large-scale data exploration. The material covered in this chapter is also available as conference proceedings paper [Fiorini et al., 2017b]. Chapter 5 describes the preliminary results of an ongoing work held in collaboration with *Istituto Giannina Gaslini Children's Hospital* (Genoa, IT) on age prediction from molecular biomarkers (paper in preparation).

Chapter 6 describes a work held in collaboration with the *Italian Multiple Sclerosis Foundation* (Genoa, IT). This work aims at devising a model to predict the evolution of multiple sclerosis patients exploiting the use of patient-friendly and inexpensive measures such as patient centered outcomes. Most of the material covered in this chapter is also available as conference proceeding paper [Fiorini et al., 2015], and peer-reviewed journal articles [Brichetto et al., 2015; Fiorini et al., 2016; Brichetto et al., 2016; Fiorini et al., 2017c; Tacchino et al., 2017].

Chapter 7 describes a work held in collaboration with *Ospedale Policlinico San Martino*. In this work a machine learning time-series model is used to forecast future glucose sensor data values. This work is based on data collected by type I and type II diabetic patients. The material covered in this chapter was recently presented at an international IEEE conference, thus it is available as proceeding paper [Fiorini et al., 2017a].

Conclusions are finally drawn in Chapter 8.

Every figure and every experimental result obtained in this thesis can be easily reproduced by using the Python scripts and JUPYTER notebooks[1] available on a dedicated GitHub repository: https://github.com/samuelefiorini/phdthesis, which also keeps track of

[1] Source: http://jupyter.org (last visit 2018-01).

the LaTeX source code of the thesis.

1.2 Basic notation and definitions

In this thesis, the following notation is adopted.

For unsupervised problems, datasets $\mathcal{D}$ are described as collection of samples $X \in \mathbb{R}^{n \times d}$. Whereas, for supervised problems, datasets are described as input-output pairs, $X \in \mathbb{R}^{n \times d}$ and $Y \in \mathbb{R}^{n \times k}$, respectively. The i-th row of X is a d-dimensional data point $\boldsymbol{x}_i$ belonging to the input space $\mathcal{X} \subseteq \mathbb{R}^d$. The corresponding outputs $\boldsymbol{y}_i$ belong to the output space $\mathcal{Y}$.

The nature of the output space defines the problem as *binary classification* if $\mathcal{Y} = \{a, b\}$ (with $a \neq b$), *multiclass classification* if $\mathcal{Y} = \{\alpha, \beta, \ldots, \omega\}$ (with $\alpha \neq \beta \neq \cdots \neq \omega$), *regression* if $\mathcal{Y} \subseteq \mathbb{R}$ and *vector-valued* or *multi-task regression* if $\mathcal{Y} \subseteq \mathbb{R}^k$. For binary classification problems common choices for the label encoding are $a = 1, b = -1$ or $a = 0, b = 1$. For multiclass classification problems classes are usually encoded as natural numbers, *i.e.* $\alpha, \beta, \ldots, \omega \in \mathbb{N}$.

Predictive models are functions $f : \mathcal{X} \to \mathcal{Y}$. The number of relevant variables is d^*. In feature selection tasks, the number of selected features is $\tilde{d}$.

A kernel function acting on the elements of the input space is defined as $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$, where $\phi(\boldsymbol{x})$ is a *feature map* from $\mathbb{R}^d \to \mathbb{R}^{d'}$. In general, feature learning algorithms project the data into a p-dimensional space.

Whenever possible, real-valued variables are indicated with lowercase letters (*e.g.* a), unidimensional vectors with lowercase bold letters (*e.g.* $\boldsymbol{a}$) and matrices, or tensors, with capital letters (*e.g.* A). When the value of some variable/parameter is the result of a data-driven estimation, such variable will be highlighted with a hat (*e.g.* $\hat{a}$). When used in the context of a data matrix, a subscript index will be used to identify a sample (row) whereas a superscript index will refer to a given feature (column). So, for instance, given a data matrix $X \in \mathbb{R}^{n \times d}$ the j-th feature of the i-th sample is x_i^j, with $0 \leq i \leq n$ and $0 \leq j \leq d$.

Part I

2 Data science background

This chapter introduces the general concept of data science and moves the focus of the thesis toward biomedical problems. Relevant examples on how clinical questions can be translated in data science problems are also presented.

Data science is an evolving cross-disciplinary field comprising the techniques used to collect and analyze arbitrarily large quantities of measures. The final aim of a data science application is to devise data-driven and statistically sound decision making strategies. This new field is of extreme interest for both industry and academia. In fact, following the data science philosophy, it is possible to obtain highly efficient solutions and new knowledge at the same time and in the same framework.

The term data science is nowadays becoming ubiquitous; although, deeply understanding what really is and, most importantly, why should we care about it, can be difficult. This may be due to the confusing fuss that generally surrounds trending terms in the Internet era. Several experts, with different backgrounds and different goals, are simultaneously presenting their opinions on the topic and often their points of view disagree.

Perhaps, focusing on the skills required to be a data scientist may shed some light on the topic. To this regard, in September 2010, Drew Conway published on his popular blog the so-called *Data Science Venn Diagram*, see Figure 2.1 [1]. Let us comment this diagram, separately focusing on each of the three main sets.

The red set, ironically named *Hacking Skills*, represents all the computer science background that a data scientist need to do his/her job. As shown in the diagram, this skill is what separates data science from traditional research. This does not mean that traditional researchers are unaware of any computer science skill, but rather that achieving good data science solutions can be, sometimes, a messy task that requires to get hands dirty with several lines of source code.

According to their scope, data collections can have various forms. They may be structured or unstructured, raw or preprocessed, distributed or centralized, and so on. The branch of data science that revolves around developing and maintaining data collections is typically called *data engineering*. Although of great interest, this aspect will not be analyzed in depth in this thesis.

The green set represents the mathematical and statistical background that is needed by data scientists to develop predictive models. Interestingly, Machine Learning (ML) is the intersection between this set with the red one. We will diffusely discuss about ML in Chapter 3, for now we will just think about it as a set of mathematical models and algorithms that, once implemented and deployed on computing facilities, are capable of automatically accomplish complex tasks.

Finally, the blue set, named *Substantive Expertise*, represents the key difference between a data scientist and a classic statistician/data analyst. The main characteristics of data scientists is that they know how to ask the *right* questions to the data. In other words, their domain knowledge allows them to understand which kind of information may be hidden in the data and which data-driven conclusions can be made. Data science produces statistically sound results, in which

[1] Source: https://goo.gl/m4gwmJ (last visit 2018-01).

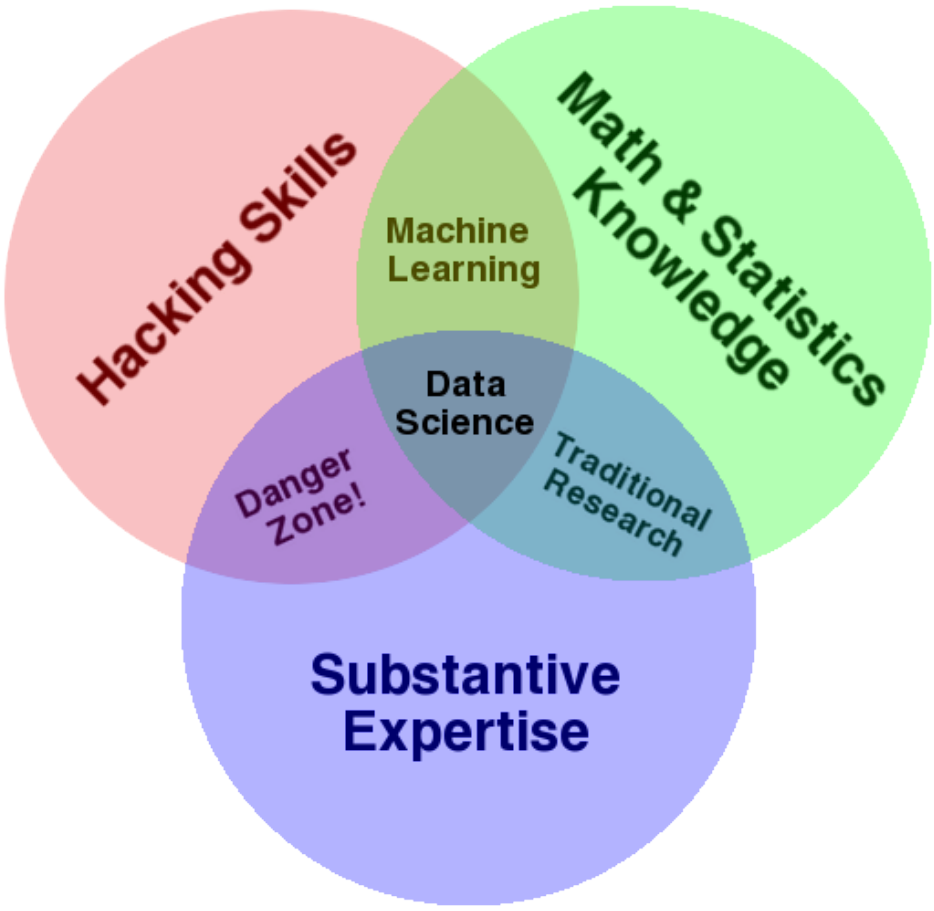


Figure 2.1: Drew Conway's Data Science Venn Diagram.

potentiality and limits of the drawn conclusion are always highlighted. Data scientists shall never *fell in love* with their data, but they should always keep a high level of objectiveness in their analysis. Otherwise, as brilliantly pointed-out by the diagram, we should fear to fall in the intersection between blue and red sets, the *Danger Zone!*

As of today, data are leading recent discoveries in almost every scientific field. In the next sections of this thesis we will focus on the application of data science techniques to biological and biomedical domains.

2.1 Turning biological questions into data science problems

One of the main characteristics of data scientists is being able to devise statistically sound procedures to provide data-driven answers to practical questions. This is very much the case in applied life science studies, where the biological question usually drives the data collection and therefore the data science challenge to be solved.

Although, in order to achieve meaningful results, thorough protocols must be followed, as widely described in Chapter 3. Here, we see an overview of the most recurrent biological questions and their direct translation into data science tasks.

How to predict phenotypes from observed data? Starting from a collection of input

measures that are likely to be related with some known target phenotype (*e.g.* eyes/hair color, limbs shape/size, internal anomaly, *etc*.), the final goal can be to develop a model that represents the relationship between input and output. Several researches fall in this class, for instance in molecular (*e.g.* lab tests, gene expression, proteomics, sequencing) [Angermueller et al., 2016; Okser et al., 2014; Abraham et al., 2013] or radiomics/imaging studies (*e.g.* MRI, PET/SPECT, microscopy) [Min et al., 2016; Helmstaedter et al., 2013]. Biological questions of this class are usually tackled by *supervised learning* models. In particular, when the observed clinical outcome is expressed as a one-dimensional continuous value, as in survival analysis, a *single-output regression* problem is posed. Moreover, if the outcome is vector-valued, as in the case of multiple genetic trait prediction [He et al., 2016], the problem can be cast in a *multiple-output regression* framework [Baldassarre et al., 2012; Argyriou et al., 2008]. Biological studies involving categorical outcomes translate into *classification* problems. In particular, if the clinical outcome assumes only two values, as in the *case-control* scenario, the classification problem is said to be *binary*, whilst, if multiple classes are observed, the classification task becomes *multiclass* [Yuan et al., 2016; Ancona et al., 2005] (detailed discussion on this topic is provided in Section 3.1).

Which variables are the most significant? A complementary question revolves around the interpretability of the predictive model. In particular, if dealing with high-dimensional biological data, the main goal can be to identify a relevant subset of meaningful variables for the observed phenomenon. This problem can be cast into a variable/feature selection problem [Guyon et al., 2002]. In particular, a predictive model is said to be *sparse* when it only contains a small number of non-zero parameters, with respect to the number of features that can be measured on the objects this model represents [Hastie et al., 2015; Meier et al., 2008]. This is closely related to feature selection: if these parameters are weights on the features of the model, then only the features with non-zero weights actually enter the model and can be considered selected (more details on this topic are presented in Section 3.2).

How to stratify the data? Collecting measures from several samples, the final goal here is to divide them in homogeneous groups, according to some *similarity* criterion. In data science, this is usually referred to as *clustering* [Hastie et al., 2009]. This happens quite often, for instance when the final goal is to identify the number of groups in some population, or when we look for a single sample which is prototypical of some situation. More details on clustering are presented in Section 3.3.1.

How to represent the samples? In order to formulate a model of some natural phenomenon, it is necessary to design and follow a suitable data collection protocol. A natural question that may arise can be whether the raw collected measures are intrinsically representative of the target phenomenon or if some transformation must be applied in order to achieve a data representation that can be successfully exploited by a predictive model. For instance, it may be plausible to assume that the data lie in a low-dimensional embedding or, conversely, that they can be better represented by a richer polynomial or Gaussian expansion. A common solution, in this case, is to take advantage of *feature engineering* techniques to obtain hand crafted features. However, this process can be very time-consuming and it may require the help of domain experts. The process of automatically identifying suitable representations from the data itself is usually referred to as *(un)supervised feature learning* [Angermueller et al., 2016; Mamoshina et al., 2016] (more details on this topic are provided in Section 3.3.2).

Are there recurring patterns in the data? Analyzing data coming from complex domains, one may be interested in understanding whether complex observations can be represented by some combination of simpler events. This typically translates into *adaptive sparse coding* or *dictionary learning* problems [Masecchia et al., 2015; Alexandrov et al., 2013; Nowak et al., 2011].

How to deal with missing values? Applied life science studies must often deal with the issue of missing data. For instance, peaks can be missed in mass-spectrometry [Jung et al., 2014] or gene expression levels can be impossible to measure due to insufficient array resolution or image corruption [Stekhoven and Bühlmann, 2011; Troyanskaya et al., 2001]. Common strategies, such as discarding the samples with missing entries, or replacing the holes with mean, median or most represented value, fall short when the missing value rate is high or the number of collected samples is relatively small. This problem usually translates into a *matrix completion* task [Candès and Recht, 2009].

3 Machine learning state of the art

This chapter defines the concept of machine learning in its two major declinations: supervised and unsupervised. It continues providing a comprehensive overview of algorithms, models and techniques relevant for the biomedical data science applications described in Part II. At the end of this chapter, an overview on the computational requirements and the most recent machine learning technologies is given.

The term *Machine Learning* (ML) first appeared in the late '50s in the field of computer science and it is now becoming a buzzword used in several contexts spanning from particle physics and astronomy to medicine and social sciences [Service, 2017]. With a simple search on Google Trends[1] it is possible to roughly quantify the pervasiveness of this term on the Internet in the last few years. From Figure 3.1 we can see that the interest toward both the terms *machine learning* and *data science* is growing, with the first consistently superior to the second.

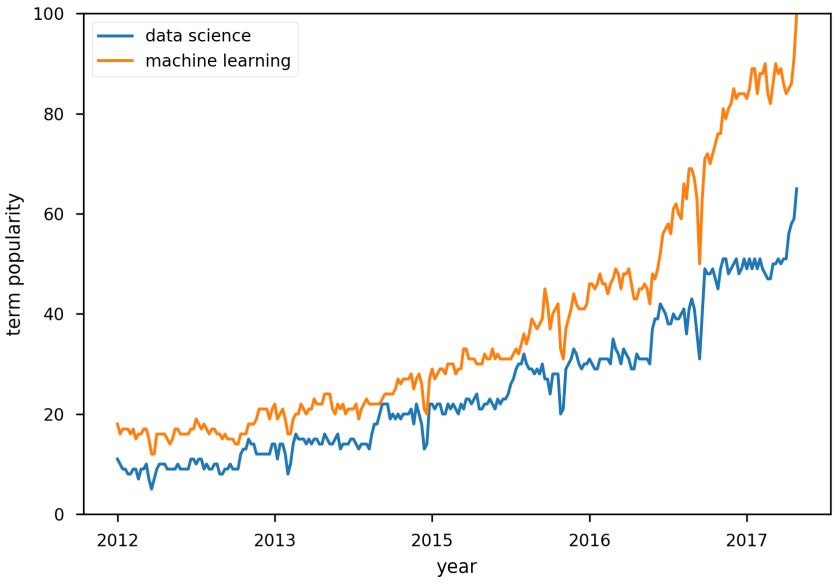


Figure 3.1: The Internet popularity over the past five years of two terms: *data science* and *machine learning*. The vertical axis represents the number of Google searches of an input term normalized with respect to its maximum.

A partial explanation to this phenomenon can be found in a recent article published on Science [Appenzeller, 2017], where the authors observed how the explosion of modern data collection abilities is leading the human kind toward another *scientific revolution*. Biomedical applications are prototypical in this sense. For instance, the volume of raw data acquired from a genome sequencer for a single DNA has a volume of approximately 140 GB [Marx, 2013]. Another example can be the 3D reconstruction of cardiac MRI acquisition which needs around

[1] Source: https://trends.google.com (last visit 2017-09).

20 GB for a single human heart, or the 3D CT scan which has a volume in the order of GB for each patient, and so on. It has been estimated that an average hospital currently stores more than 665 TB of data that needs to be analyzed and understood [2]. Such massive amounts of data have long overwhelmed human analysis and insights potential. This makes ML, and artificial intelligence in general, a key element for clinicians and scientists that try to make sense of large-scale observations.

But, what is *machine learning*? And how does it differ from classical statistics?

A unique answer to this question is not easy to provide. In fact, ML can be defined in different ways and from several standpoints. Let us see three remarkable examples.

1. Kevin P. Murphy in its *Machine Learning - A Probabilistic Perspective* [Murphy, 2012] defines ML as follows.

   > "[. . .] *a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty* [. . .]"

2. Trevor Hastie, a well-known applied statistician, in a famous seminar[3] held in October 2015 at the Stanford University, gave the following three definitions.

   > **Machine Learning** *constructs algorithms that can learn from data.*
   >
   > **Statistical Learning** *is a branch of applied statistics that emerged in response to machine learning, emphasizing statistical models and assessment of uncertainty.*
   >
   > **Data Science** *is the extraction of knowledge from data, using ideas from mathematics, statistics, machine learning, computer science, engineering...*

3. Carl E. Rasmussen in the preface of its renowned *Gaussian Processes for Machine Learning* [Rasmussen and Williams, 2006] introduces the difference between classical statistics and ML as follows.

   > "[. . .] *in statistics a prime focus is often in understanding the data and relationships in terms of models giving approximate summaries such as linear relations or independencies. In contrast, the goals in machine learning are primarily to make predictions as accurately as possible and to understand the behaviour of learning algorithms* [. . .]"

It looks like each author, according to his background, expertise and experience, provides a slightly different definition of ML. Trying to summarize these three standpoints, we can say that *ML is an interdisciplinary field that borrows the concept of data-driven model from statistics in order to devise algorithms that can exploit hidden patterns in current data and make accurate predictions on future data*.

As of today ML is the workhorse of data science.

[2] Source: https://goo.gl/jMYvwh (last visit 2018-01).

[3] Part of Data Science @ Stanford Seminar series. Source: https://goo.gl/UFgqxU (last visit 2018-01).

3.1 Supervised learning

Humans are remarkably good at *learning by examples*. When a kid is taught what a pencil looks like, he will be capable of understanding the concept of pencil from a limited number of guided observations. Similarly, when future radiologists are trained to distinguish between healthy tissues from tumors in MRI scans, they will be provided with several annotated biomedical images from which they will be able to generalize. The applied learning paradigm is characterized by the presence of two key objects: *data* and *labels*. In the last example, the MRI scans are the data, and their annotations (*e.g.* tumor *vs* healthy tissue) are the labels.

Supervised learning is the branch of ML in which predictive models are trained on labeled data. In the ML jargon, and in this thesis, one usually refers to *data* as collections of *samples* described by an arbitrarily large number of *predictors* (*features*) that are used as *input* in a training process having labels as *output*.

Input samples throughout this thesis are represented as d-dimensional vectors $\boldsymbol{x}$ belonging to an input space $\mathcal{X}$, where typically $\mathcal{X} \subseteq \mathbb{R}^d$ and labels are represented with the variable y belonging to an output space $\mathcal{Y}$. The nature of $\mathcal{Y}$ defines the learning task as *binary classification* if $\mathcal{Y} = \{-1, +1\}$, *multiclass classification* if $\mathcal{Y} = \{1, 2, \ldots, k\}$, *regression* if $\mathcal{Y} \subseteq \mathbb{R}$ or *vector-valued regression* if $\mathcal{Y} \subseteq \mathbb{R}^k$. The remainder of this section summarizes the methods that are most relevant with the data-driven strategies adopted to tackle the biomedical data science challenges described in the second part of in this thesis.

Given a set of input-output pairs $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n = (X, \boldsymbol{y})$, supervised learning methods aim at finding a function of the inputs $f(\boldsymbol{x})$ that approximates the output y. This translates into the minimization problem defined in Equation (3.1).

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \tag{3.1}$$

The loss function $L(f(\boldsymbol{x}), y)$ can be seen as a measure of *adherence* to the available training data. Several loss functions for regression and classification problems were proposed; Table 3.1 defines the most commonly adopted in biomedical studies while their visual representation is presented in Figure 3.2. Choosing the appropriate loss function for the problem at hand is crucial and there is no trivial solution for this problem. Different choices for $L(f(\boldsymbol{x}), y)$ identifies different learning machines, that are known under different names. The most popular methods will be presented in the next few sections.

Identifying a reliable data-driven model can be a very tricky task. Many unwanted and concurrent factors may be misleading and the solution may have poor predictive power for several reasons. Including:

1. the acquisition devices may introduce random fluctuations in the measures;

2. the amount of collected samples n may be small with respect to the number of observed variables d;

3. a non-negligible number of the measured variables may not be representative of the target phenomenon.

Table 3.1: Definition of the loss functions for regression (top) and classification (bottom) problems represented in Figure 3.2.

Loss function	$L(f(\boldsymbol{x}), y)$	Learning problem
Square	$(y - f(\boldsymbol{x}))^2$	regression
Absolute	$\lvert y - f(\boldsymbol{x})\rvert$	regression
ε-insensitive	$\lvert y - f(\boldsymbol{x})\rvert_\varepsilon$	regression
Zero-one	$\mathbb{1}\{y = f(\boldsymbol{x})\}$	classification
Square	$(1 - yf(\boldsymbol{x}))^2$	classification
Logistic	$\log(1 + e^{-yf(\boldsymbol{x})})$	classification
Hinge	$\lvert 1 - yf(\boldsymbol{x})\rvert_+$	classification
Exponential	$e^{-yf(\boldsymbol{x})}$	classification

From a modeling standpoint, every combination of the factors above can be seen as *noise* affecting the data. Precautions in the model formulation process must be taken in order to achieve solutions that are insensitive to small changes in the input data and that are, in general, *robust* to the noise effect.

Considering a ML model ($\hat{f}$) fitted on a collection of data ($\mathcal{D}$), the most desirable property of $\hat{f}$ is that it should be able to achieve good prediction performance not only on $\mathcal{D}$, but also on all the future, therefore unseen, data points $\mathcal{D}'$. In other words, assuming that the samples in $\mathcal{D}$ are affected by some kind of random[4] component, $\hat{f}$ should be a predictive function that does not *follow the noise*, but rather models the true input-output relationship. In ML, a model that fits well $\mathcal{D}$ but performs poorly on $\mathcal{D}'$ is said to be *overfitting*.

In ML literature, *regularization* is the most important countermeasure to overfitting and it is widely adopted, under several forms, to build predictive models out of noisy data.

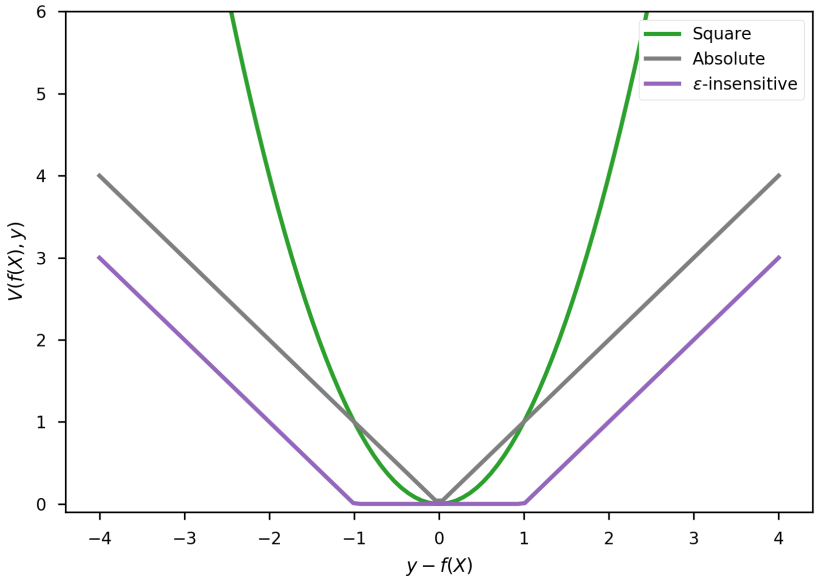
The original contribution of this thesis mainly relies on the application of data science and ML concepts to noisy domains. Therefore, regularization strategies are of primary interest in this discussion. For each learning algorithm described, particular emphasis will be put on the relevant *regularization* strategies.

In its broader definition *regularization* can be seen as the process of introducing additional information in order to solve a possibly ill-posed problem. As shown in Equation (3.2), this is typically translated in the use of a regularization penalty $\mathcal{R}(f)$, controlled by a regularization parameter λ [Tikhonov, 1963; Evgeniou et al., 2000].

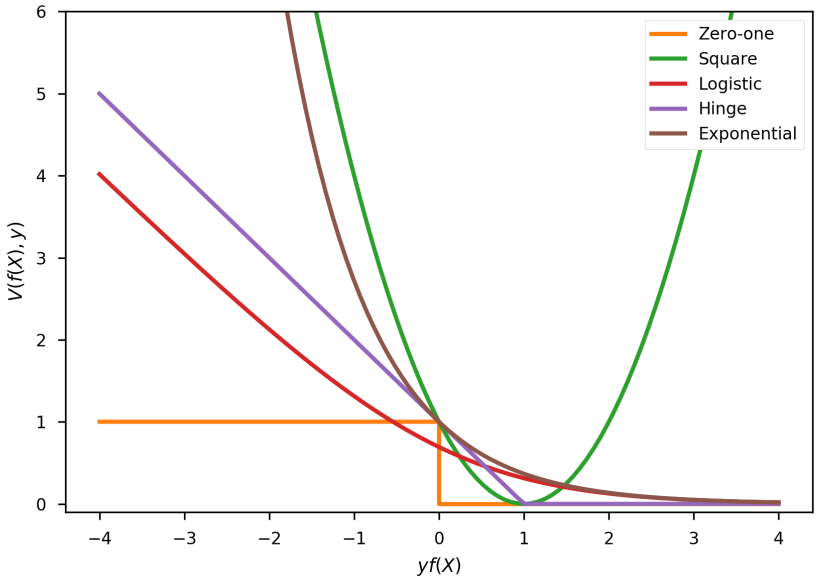
$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) + \lambda \mathcal{R}(f) \tag{3.2}$$

Choosing different $\mathcal{R}(f)$ implies inducing different effects on the solution and it also leads to the definition of different learning machines (see Section 3.1.1). With the regularization parameter λ it is possible to control the trade-off between adherence to the training data and strength of the effect induced by $\mathcal{R}(f)$. As an example, we can think of using a penalty that induces smoothness, such as the ℓ_2-norm, or sparsity, such as the ℓ_1-norm, in the solution. A pictorial representation of a learning machine working in overfitting, underfitting and optimal

[4] Here with *random* means "*uncorrelated with the input-output relationship*".



(a)



(b)

Figure 3.2: An overview on the most common loss functions for regression (a) and classification (b) problems plotted against the corresponding prediction error.

fitting regime in regression and classification cases can be seen in Figure 3.3 [5] and Figure 3.4 [6], respectively.

Supervised learning machines may rely on very different mathematical backgrounds such as generalized linear models, nonlinear deep neural networks, kernels, trees, ensemble of trees,

[5] Source https://losc.ligo.org/events/GW150914/ (last visit 2018-01).
[6] Source https://archive.ics.uci.edu/ml/datasets/HTRU2 (last visit 2018-01).

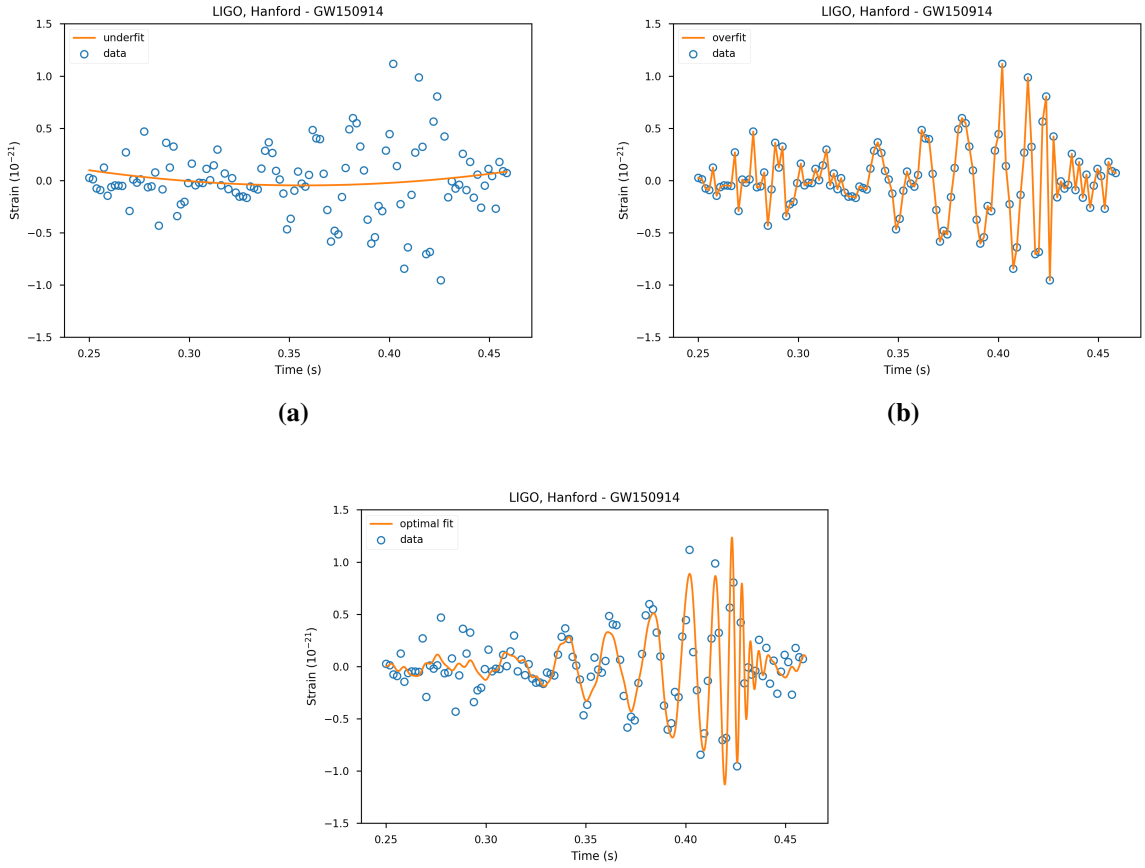


Figure 3.3: An example of underfit (a), overfit (b) and optimal fit (c) for a nonlinear regression problem. The data are a downsampled version ($f_s = 546\ Hz$) of the first observation of gravitational waves from a binary black hole merger detected on September 14th, 2015, 09:50:45 UTC at LIGO, Hanford (WA).

etc. Nevertheless, disregarding their nature, they all share the common structure defined in Equation (3.2). The solution of this problem can be achieved either by Empirical (or Structured) Risk Minimization (ERM) either by Maximum Likelihood/A Posteriori (MLE/MAP) Estimation. See Appendix A for more details on this two strategies, and their connection.

3.1.1 Regularization methods

Regularization methods is a broad class of models that include linear and nonlinear techniques for both regression and classification. The main characteristic of the methods falling in this class, is that they are particularly straightforward to express as in Equation (3.2). In fact, as described in [Evgeniou et al., 2000], they are easily identifiable by the use of one loss function $L(f(\boldsymbol{x}), y)$ and one, or more, regularization penalty $\mathcal{R}(f)$.

In the following sections an overview of the most popular regularization methods is presented.

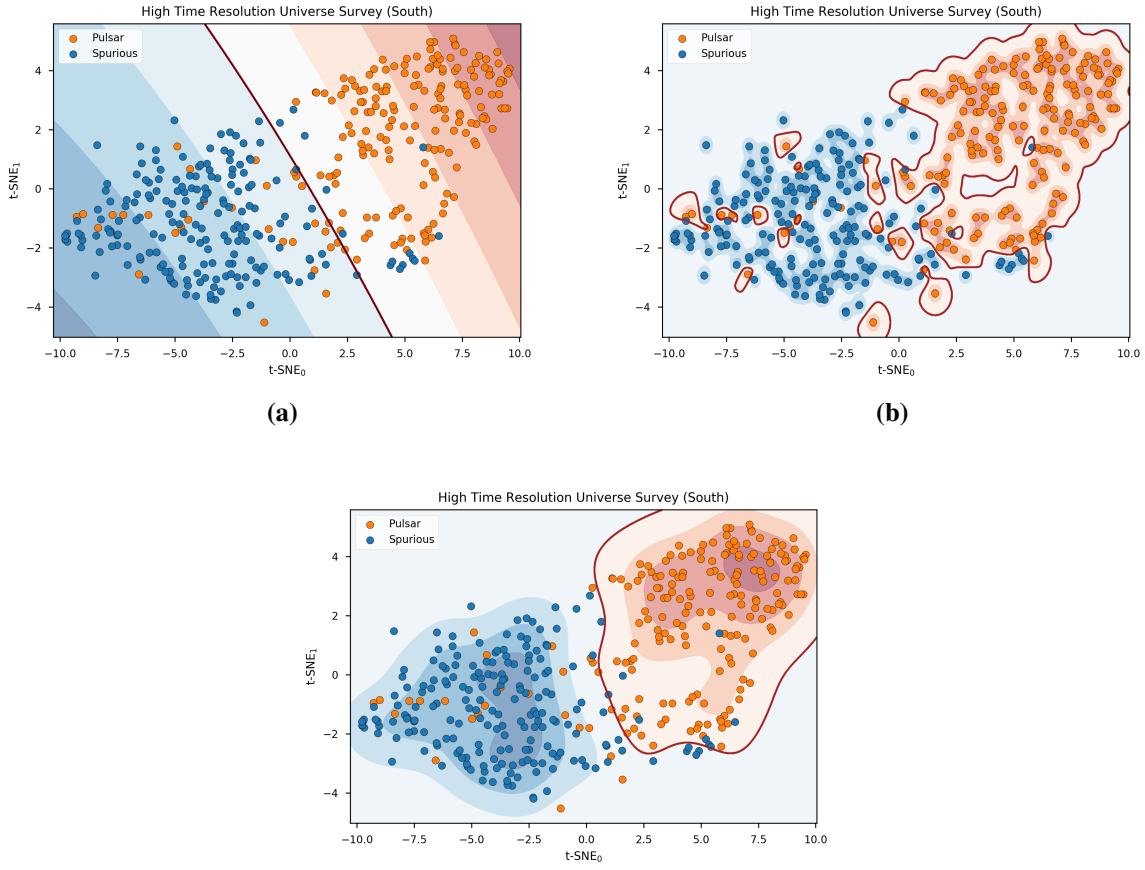


Figure 3.4: An example of underfit (a), overfit (b) and optimal fit (c) for a nonlinear binary classification problem. Each data point is a pulsar candidate randomly sampled from the High Time Resolution Universe Survey (South) dataset. The data are standardized and and projected on a two-dimensional plane by the t-SNE [Van der Maaten and Hinton, 2008].

3.1.1.1 Ordinary least squares

We start this discussion focusing on linear models $\hat{y} = f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{w}$ and taking into account the most popular loss function for regression problems: the square loss $L(\hat{y}, y) = (\boldsymbol{x}^T\boldsymbol{w} - y)^2$. The data fitting problem expressed in Equation (3.3) is known as *Ordinary Least Squares* (OLS), or simply as *linear regression*, and it does not include any regularization term.

$$\hat{\boldsymbol{w}}_{\text{OLS}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)^2 = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 \tag{3.3}$$

The minimization problem in Equation (3.3) is convex and differentiable and its solution can be achieved in closed-form as

$$\hat{\boldsymbol{w}}_{\text{OLS}} = (X^TX)^{-1}X^T\boldsymbol{y}$$

or by iterative minimization routines such as (stochastic) gradient descent-like algorithms [Boyd and Vandenberghe, 2004; Sra et al., 2012]. A pictorial representation of the solution of OLS can be seen in Figure 3.12a.

In case of multiple regression tasks, the OLS approach can be extended to vector-valued regression as well. In this case the least squares problem can be written as

$$\begin{aligned}\hat{W}_{\text{OLS}} &= \underset{W \in \mathbb{R}^{d \times k}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{k} (\boldsymbol{x}_i^T \boldsymbol{w}^t - y_i^t)^2 \\ &= \underset{W \in \mathbb{R}^{d \times k}}{\operatorname{argmin}} \frac{1}{n} \|XW - Y\|_F^2\end{aligned} \tag{3.4}$$

where $\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{t=1}^{k} |a_i^t|^2}$ is the *Frobenius* norm (also known as *Hilbert-Schmidt* norm) and it can be considered as an extension of the ℓ_2-norm to the matrix case. Lacking of appropriate regularization penalties, solving the problem in Equation (3.4) corresponds to solving k isolated regression problems, one for each task. The vector-valued OLS approach, even if theoretically legit, is rather uncommon in practical applications and a regularized version of Equation (3.4) is typically preferred (see following sections).

Even though mainly used for regression, the square loss can also be used to solve binary classification problems (see Table 3.1). In this case, it can be rewritten as

$$(\boldsymbol{x}^T \boldsymbol{w} - y)^2 = (1 - y \cdot \boldsymbol{x}^T \boldsymbol{w})^2 \tag{3.5}$$

exploiting the fact that the two classes are encoded with binary labels: $\boldsymbol{y} \in \{+1, -1\}^n$. For multiclass classification problems, strategies such as *One-vs-One* (OVO) or *One-vs-All* (OVA) can be adopted to reduce the problem to multiple binary classifications [Hastie et al., 2009].

OLS is probably the most naïve prediction strategy, nevertheless it is widely adopted in several studies. Let us see what happens when we use the OLS model on a real regression problem.

For this example, and the following ones, we take into account the dataset $\mathcal{D}_{\text{aging}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n=111}$ where each input sample $\boldsymbol{x}_i \in \mathbb{R}^{12}$ presents a set of measures describing the metabolic state of a healthy subject and $y_i \in \mathbb{N}_+$ is its age expressed in years. For the sake of this discussion a thorough description of $\mathcal{D}_{\text{aging}}$ at this point is irrelevant[7], we can simply think as the $d = 12$ variables as predictors of the outcome y and we look for some linear input-output relationship. In order to do that, we randomly split $\mathcal{D}_{\text{aging}}$ in two chunks obtaining a training and a test set of $n_{\text{tr}} = 74$ and $n_{\text{ts}} = 37$ samples respectively. Then, we fit the OLS model on the training set obtaining the weights vector $\hat{\boldsymbol{w}}_{\text{OLS}}$ represented in Figure 3.5. As we can see, in order to achieve a predictive model, OLS can only spread the weights across all the input variables. Evaluating $\hat{\boldsymbol{w}}_{\text{OLS}}$ on the test set, this model has a Mean Absolute Error (MAE) of 10.598 years and explains the 74.29% of the variance.

This result looks promising, but we will see in the next sections whether they can be improved with the use of some regularization penalty.

3.1.1.2 Ridge regression

In its original proposition, *ridge regression* [Hoerl and Kennard, 1970] is defined as a least squares problem penalized by the squared ℓ_2-norm of the regression coefficients, see Equation (3.6).

[7] This regression problem is widely described and analyzed in Chapter 5.

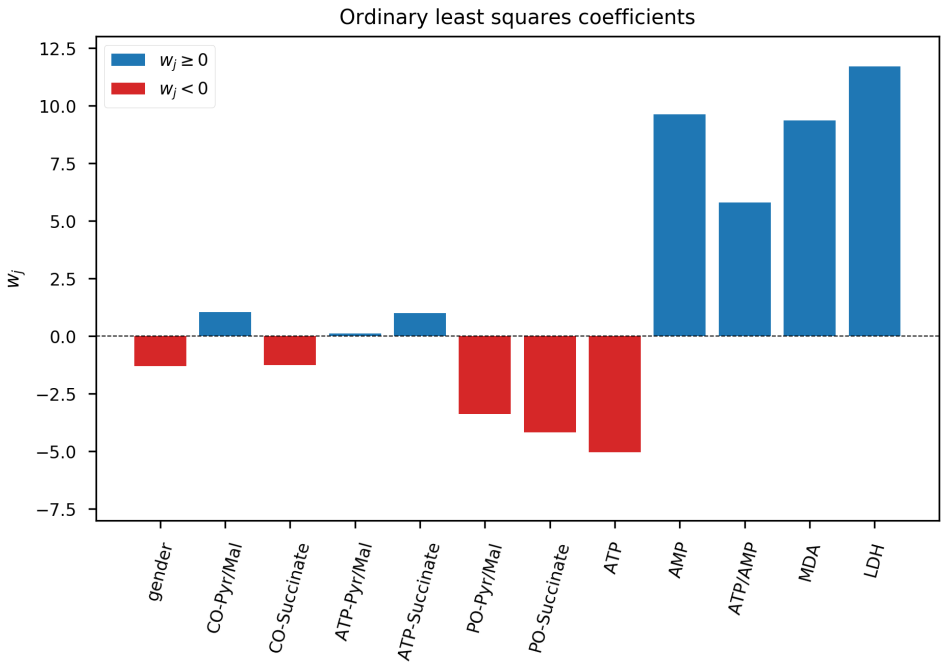


Figure 3.5: A pictorial representation of the vector $\hat{\boldsymbol{w}}_{\text{OLS}}$ obtained fitting an OLS model on 74 randomly selected training samples of $\mathcal{D}_{\text{aging}}$. Variables associated with positive (*i.e.* directly proportional to the output) and a negative (*i.e.* inversely proportional) weight are represented in blue and red, respectively.

$$\mathcal{R}_{\ell_2}(\boldsymbol{w}) = \sum_{j=1}^{d}(w_j)^2 = \|\boldsymbol{w}\|_2^2 \tag{3.6}$$

Therefore, the ridge regression minimization problem can be written as in Equation (3.7).

$$\hat{\boldsymbol{w}}_{\ell_2} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{argmin}} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)^2 + \lambda\sum_{j=1}^{d}(w_j)^2 = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{argmin}} \frac{1}{n}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2 \tag{3.7}$$

This penalty leads to smooth solutions as it shrinks the coefficients toward zero, but it does not achieve a parsimonious representation, as it always keep all the variables in the model. The ridge regression problem of Equation (3.7) is convex and differentiable and a pictorial representation of its solution in a 2D case is depicted in Figure 3.12b. The ridge coefficients $\hat{\boldsymbol{w}}_{\ell_2}$ can be estimated in closed-form as

$$\hat{\boldsymbol{w}}_{\ell_2} = (X^TX + \lambda nI)^{-1}X^T\boldsymbol{y}$$

where I is the $d \times d$ identity matrix. An estimate for the ridge coefficients can also be obtained with gradient descent-like optimization routines [Boyd and Vandenberghe, 2004; Sra et al., 2012].

The regularization parameter λ plays the fundamental role of balancing the trade-off between data adherence and smoothness of the solution. Penalizing the ℓ_2-norm of the regression coefficients, their value is shrunk toward zero. This results in an increased robustness of the solution to the noise affecting the training data.

In case of multiple outputs, the vector-valued ridge regression problem can be written as in Equation (3.8).

$$\begin{aligned}\hat{W}_{\ell_2} &= \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{k}(\boldsymbol{x}_i^T\boldsymbol{w}^t - y_i^t)^2 + \lambda\sum_{j=1}^{d}\sum_{t=1}^{k}|w_j^t|^2 \\ &= \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n}\|XW - Y\|_F^2 + \lambda\|W\|_F^2\end{aligned} \tag{3.8}$$

As already seen for vector-valued OLS, the Frobenius norm penalty does not induce any task coupling, hence solving the problem in Equation (3.8) still corresponds to individually solve k regression tasks.

This method can be applied to binary classification problems by using the margin loss function of Equation (3.5). Nevertheless, for ℓ_2-norm penalized classification problems the use of the *logistic loss* is usually preferred (see Section 3.1.1.6).

In deep learning literature, penalizing the regression coefficients with the ℓ_2-norm is known as *weight decay* [Krogh and Hertz, 1992]. Ridge regression can also be considered a form of *Tikhonov regularization* [Tikhonov, 1963] and of *regularization network* [Evgeniou et al., 2000].

Let us see what happens when this method is applied to a real regression problem. For ease of comparison, we take into account the dataset $\mathcal{D}_{\text{aging}}$, introduced in Section 3.1.1.1. Compared to OLS, ridge regression has the parameter λ that must be fixed before fitting the model. In this example, we estimated the best value $\hat{\lambda}_{\text{cv}}$ according to a standard grid-search cross-validation strategy. This consists in fixing a range of 30 possible values for λ (in a logarithmic scale from 10^{-3} to 10^2) and pick the best value as the one achieving the lowest validation error, estimated via (5-fold) cross-validation (see Section 3.4.1). Therefore, once the best value for the regularization parameter is fixed ($\hat{\lambda}_{\text{cv}} = 20.43$), in this case. the experimental setup used for OLS is preserved.

The ridge coefficients $\hat{\boldsymbol{w}}_{\ell_2}$ are represented in Figure 3.6. Comparing Figure 3.6 and Figure 3.5 we can see that the amplitude of the ridge regression coefficients is, in absolute value, decreased by the use of the ℓ_2 penalty. Indeed, several entries of $\hat{\boldsymbol{w}}_{\ell_2}$ are very small, but none of them is exactly zero. This is the expected behavior of the ℓ_2-norm penalty. Evaluating $\hat{\boldsymbol{w}}_{\ell_2}$ on the test set, this model has MAE $= 8.615$ years and explains the 81.19% of the variance, outperforming OLS.

3.1.1.3 Lasso

The Lasso [Tibshirani, 1996] can be defined as a least square problem penalized by the ℓ_1-norm of the regression coefficients, see Equation (3.9).

$$\mathcal{R}_{\ell_1}(\boldsymbol{w}) = \sum_{j=1}^{d}|w_j| = |\boldsymbol{w}|_1 \tag{3.9}$$

Therefore, the Lasso minimization problem can be written as in Equation (3.10).

$$\hat{\boldsymbol{w}}_{\ell_1} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i^T\boldsymbol{w} - y_i)^2 + \lambda\sum_{j=1}^{d}|w_j| = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \lambda|\boldsymbol{w}|_1 \tag{3.10}$$

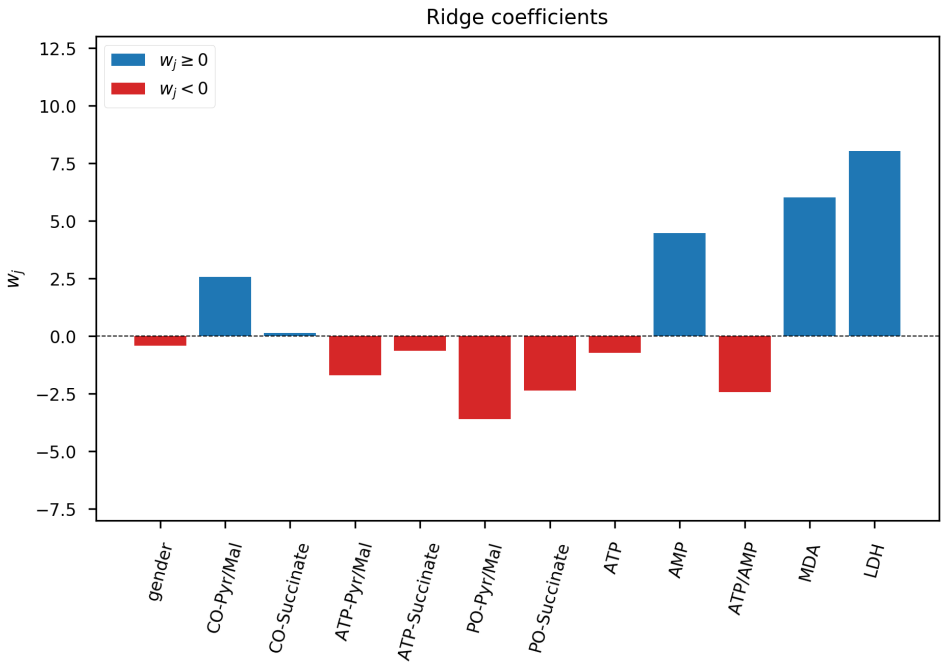


Figure 3.6: A pictorial representation of the vector $\hat{\boldsymbol{w}}_{\ell_2}$ obtained fitting a ridge regression model on 74 randomly selected training samples of $\mathcal{D}_{\text{aging}}$. Variables associated with positive (*i.e.* directly proportional to the output) and a negative (*i.e.* inversely proportional) weight are represented in blue and red, respectively.

The Lasso can be used to perform linear model fitting and, thanks to its desirable properties, it is a popular choice for embedded variable selection [Guyon and Elisseeff, 2003], as described in Section 3.2. At first, the ℓ_1-norm enforces sparsity in the solution, hence producing compact and easily interpretable results. Secondly, the Lasso optimization problem is convex and, although non-differentiable, it is computationally feasible even in very high dimensional scenarios. Popular minimization algorithms for the Lasso problem are, for instance, the *Fast Iterative Shrinkage-Thresholding Algorithm* (FISTA) [Beck and Teboulle, 2009] and the *coordinate descent algorithm* [Wu and Lange, 2008]. A pictorial representation of the Lasso solution can be seen in Figure 3.12c.

A popular application of the Lasso is to perform shrinkage and variable selection in survival analysis for Cox proportional hazard regression [Tang et al., 2017; Gui and Li, 2005; Tibshirani et al., 1997] and additive risk models [Ma and Huang, 2007]. Such ℓ_1-penalized methods are extensively applied in literature to predict survival time from molecular data collected from patients affected by different kinds of tumor.

The Lasso can also be extended to vector-valued regression problems by using the mixed $L_{2,1}$-norm, defined in Equation (3.11), as regularization penalty [Gramfort et al., 2012].

$$\mathcal{R}_{\ell_1}(W) = \sum_{j=1}^{d} \sqrt{\left(\sum_{t=1}^{k} |w_j^t|^2 \right)} = \|W\|_{2,1} \tag{3.11}$$

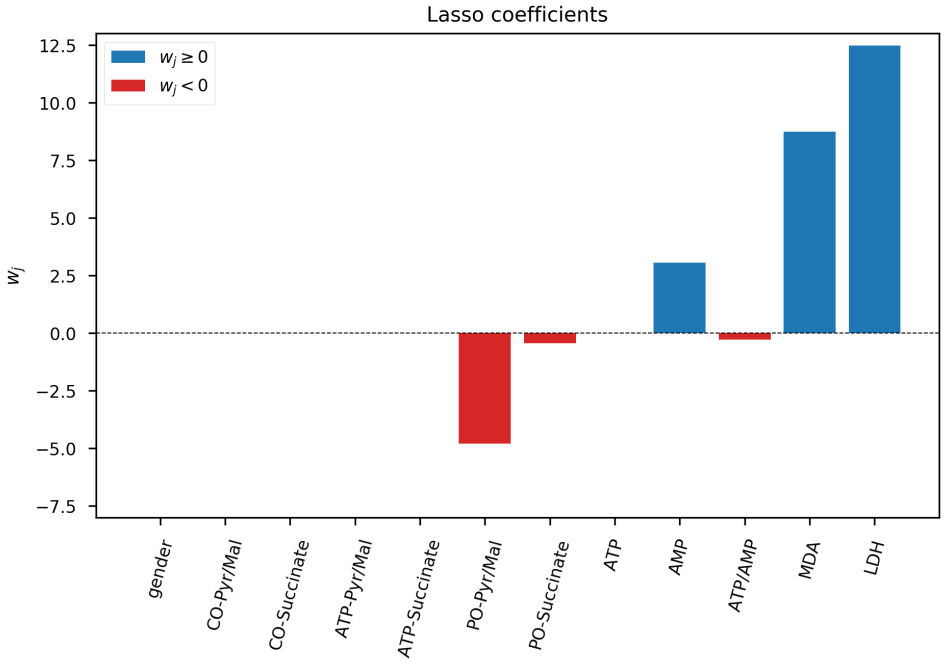


Figure 3.7: A pictorial representation of the vector $\hat{\boldsymbol{w}}_{\ell_1}$ obtained fitting a Lasso model on 74 randomly selected training samples of $\mathcal{D}_{\text{aging}}$. Variables associated with positive (*i.e.* directly proportional to the output) and a negative (*i.e.* inversely proportional) weight are represented in blue and red, respectively.

Therefore, the vector-valued regularization problem can be written as in Equation (3.12)

$$\begin{aligned}\hat{W}_{\ell_1} &= \underset{W \in \mathbb{R}^{d \times k}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{k} (\boldsymbol{x}_i^T \boldsymbol{w}^t - y_i^t)^2 + \lambda \sum_{j=1}^{d} \sqrt{\left(\sum_{t=1}^{k} |w_j^t|^2 \right)} \\ &= \underset{W \in \mathbb{R}^{d \times k}}{\operatorname{argmin}} \frac{1}{n} \|XW - Y\|_F^2 + \lambda ||W||_{2,1}\end{aligned} \tag{3.12}$$

and it is known as Multi-task Lasso [Lee et al., 2010]. Such norm enforces a row-structured sparsity in the regression weights, hence preserving the interpretability of the solution.

Originally proposed to solve regression problems, the Lasso can also be adopted in binary classification tasks; although, in this case, *sparse logistic regression* is often preferred [Wu et al., 2009] (see Section (3.1.1.6)).

Let us see what happens when the Lasso model is applied to a real regression problem. Once again, for ease of comparison with OLS and ridge, we take into account the dataset $\mathcal{D}_{\text{aging}}$, introduced in Section 3.1.1.1. The experimental setup in this case is identical to the one previously applied for ridge regression. The best value for the regularization parameter, chosen via grid-search cross-valudation, is $\hat{\lambda}_{\text{cv}} = 1.27$. The Lasso coefficients $\hat{\boldsymbol{w}}_{\ell_1}$ are represented in Figure 3.7.

Comparing $\hat{\boldsymbol{w}}_{\ell_1}$ with $\hat{\boldsymbol{w}}_{\ell_2}$ and $\hat{\boldsymbol{w}}_{\text{OLS}}$ (Figure 3.7, Figure 3.6 and Figure 3.5, respectively) we can observe that, for the first time, to only 6 variables, out of 12, a non-negative value is assigned. This is an example of the sparsity-enforcing effect of the ℓ_1-norm regularization penalty. The 6 variables with nonzero weight can be considered as *selected* for the prediction problem at hand. Evaluating $\hat{\boldsymbol{w}}_{\ell_1}$ on the test set, the Lasso has MAE $= 8.387$ years and explains the 81.51% of the variance, slightly outperforming ridge.

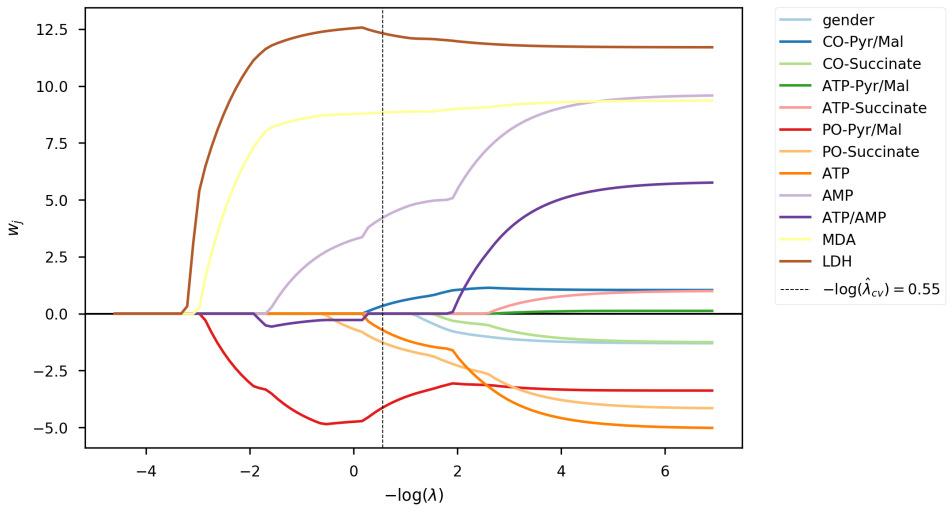


Figure 3.8: Profiles of the Lasso coefficients for the aging problem as λ decreases. The vertical dashed line represents the optimal value $\hat{\lambda}_{\text{cv}}$ estimated by grid-search (5-fold) cross-validation.

When using sparsity-enforcing penalties, such as the ℓ_1-norm, an insightful experiment can be to observe the regularization path. This can be done by iteratively fitting the model with decreasing values of the regularization parameter λ, usually expressed in logarithmic scale. An example of the Lasso path for the aging problem is reported in Figure 3.8.

Weights corresponding to features that are more likely to be relevant for the prediction problem should early move away from the horizontal axis. Conversely, as the regularization parameter increases, the model should enforce less sparsity, hence tolerating more and more irrelevant features having nonzero weight. The vertical dashed line in Figure 3.8 corresponds to $\hat{\lambda}_{\text{cv}}$ and it hits the profiles of the weights consistently with what shown in Figure 3.7.

When used for variable selection, the Lasso has two major drawbacks. First, in presence of groups of correlated variables, this method tends to select only one variable per group, ignoring the others. Secondly, the method cannot select more variables than the sample size [Waldmann et al., 2013; De Mol et al., 2009b]. The effect of such drawbacks is dramatic when using the Lasso in $n \ll d$ scenarios. In order to ameliorate this issues, several Lasso-inspired models were proposed [Meinshausen and Bühlmann, 2010; Hoggart et al., 2008; Zou, 2006]. In the next section we will describe one of the most popular and straightforward Lasso extensions: the Elastic-Net [Zou and Hastie, 2005].

3.1.1.4 Elastic-Net

The Elastic-Net method [Zou and Hastie, 2005; De Mol et al., 2009a] can be formulated as a least squares problem penalized by a convex combination of Lasso (ℓ_1-norm) and ridge regression (ℓ_2-norm) penalties, as in Equation (3.13).

$$R_{\ell_1\ell_2}(\boldsymbol{w}) = \sum_{j=1}^{d} (\alpha\,|w_j| + (1-\alpha)\,w_j^2) = \alpha\,|\boldsymbol{w}|_1 + (1-\alpha)\;\|\boldsymbol{w}\|_2^2 \tag{3.13}$$

Therefore, the Elastic-Net minimization problem can be written as in Equation (3.14)

$$\begin{aligned}\hat{\boldsymbol{w}}_{\ell_1\ell_2} &= \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i^T\boldsymbol{w}-y_i)^2 + \lambda\left[\sum_{j=1}^{d}(\alpha\,|w_j| + (1-\alpha)\,w_j^2)\right] \\ &= \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\,\|X\boldsymbol{w}-\boldsymbol{y}\|_2^2 + \lambda\left[\alpha\,|\boldsymbol{w}|_1 + (1-\alpha)\;\|\boldsymbol{w}\|_2^2\right]\end{aligned} \tag{3.14}$$

with $0 \leq \alpha \leq 1$, or equivalently as

$$\hat{\boldsymbol{w}}_{\ell_1\ell_2} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\,\|X\boldsymbol{w}-\boldsymbol{y}\|_2^2 + \tau\,|\boldsymbol{w}|_1 + \mu\;\|\boldsymbol{w}\|_2^2 \tag{3.15}$$

where $\tau = \lambda\alpha$ and $\mu = \lambda(1-\alpha)$. The first formulation of the problem, Equation (3.14), is more convenient when we want to control the overall amount of regularization with λ, given a certain amount of sparsity α. In fact, it is easy to see that fitting the Elastic-Net model for $\alpha = 0$ or $\alpha = 1$ is equivalent to solve ridge or Lasso regression, respectively. On the other hand, writing the Elastic-Net problem as in Equation (3.15) is more convenient when we want to separately control the ℓ_1- and ℓ_2-norm penalties, as in [De Mol et al., 2009b].

The Elastic-Net[8] model is widely adopted to perform linear model fitting and variable selection. Indeed, the combined presence of the two norms promotes sparse solutions where groups of correlated variables can be simultaneously selected, hence overcoming the variable selection drawbacks of the Lasso and making the Elastic-Net suitable for variable selection in $n \ll d$ scenarios. As already seen for the Lasso, the minimization problem in Equation (3.14) is convex and non-differentiable, due to the ℓ_1-norm, and it can be efficiently solved either by proximal forward-backward splitting strategies (*e.g.* FISTA [Beck and Teboulle, 2009]), or by coordinate descent [Wu and Lange, 2008]. A pictorial representation of the Elastic-Net solution can be seen in Figure 3.12d.

The Elastic-Net method is successfully applied in several biomedical fields, including gene expression [Hughey and Butte, 2015; De Mol et al., 2009b], genome-wide association studies [Waldmann et al., 2013] and other molecular data [Aben et al., 2016; Hughey and Butte, 2015].

The Elastic-Net can also be extended to vector-valued regression problems by using a convex combination of the $L_{2,1}$- and the Frobenius norms. This multiple output regression problem is known as Multi-task Elastic-Net [Chen et al., 2012] (MTEN) and it can be written as in Equation (3.16).

$$\begin{aligned}\hat{W}_{\ell_1\ell_2} &= \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{k}(\boldsymbol{x}_i^T\boldsymbol{w}^t - y_i^t)^2 + \lambda\left[\alpha\sum_{j=1}^{d}\sqrt{\left(\sum_{t=1}^{k}|w_j^t|^2\right)} + (1-\alpha)\sum_{j=1}^{d}\sum_{t=1}^{k}|w_j^t|^2\right] \\ &= \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n}\,\|XW-Y\|_F^2 + \lambda[\alpha\,||W||_{2,1} + (1-\alpha)\;\|W\|_F^2]\end{aligned} \tag{3.16}$$

Even if originally proposed to solve regression problems, as already seen for Lasso and ridge regression, the Elastic-Net can be adopted in binary classification tasks. Nevertheless, for classification problems, the use of the logistic loss is usually preferred.

[8] Actually, in the original paper the authors refer to Equations (3.14) and (3.15) as *naïve* Elastic-Net [Zou and Hastie, 2005] because empirical evidence show that the weights $\hat{\boldsymbol{w}}_{\ell_1\ell_2}$ may suffer from over-shrinking when appropriate rescaling strategies, also described in [De Mol et al., 2009b], are not applied.

Let us see what happens when the Elastic-Net model is applied to an actual regression problem. As usually, we tackle the aging task (introduced in Section 3.1.1.1). The experimental setup in this case is the same already adopted for the Lasso in Section 3.1.1.3. The only difference is that the grid-search cross-validation routine looks for the optimal values for the two regularization parameters $(\hat{\lambda}_{\text{cv}}, \hat{\alpha}_{\text{cv}}) = (0.57, 0.48)$ in a 2D grid consisting of 30×30 values (α candidates range from 0 to 1 in a linear scale, whilst λ candidates range from 10^{-3} to 10^{2} in a logarithmic scale). The Elastic-Net coefficients $\hat{\boldsymbol{w}}_{\ell_1\ell_2}$ are represented in Figure 3.9.

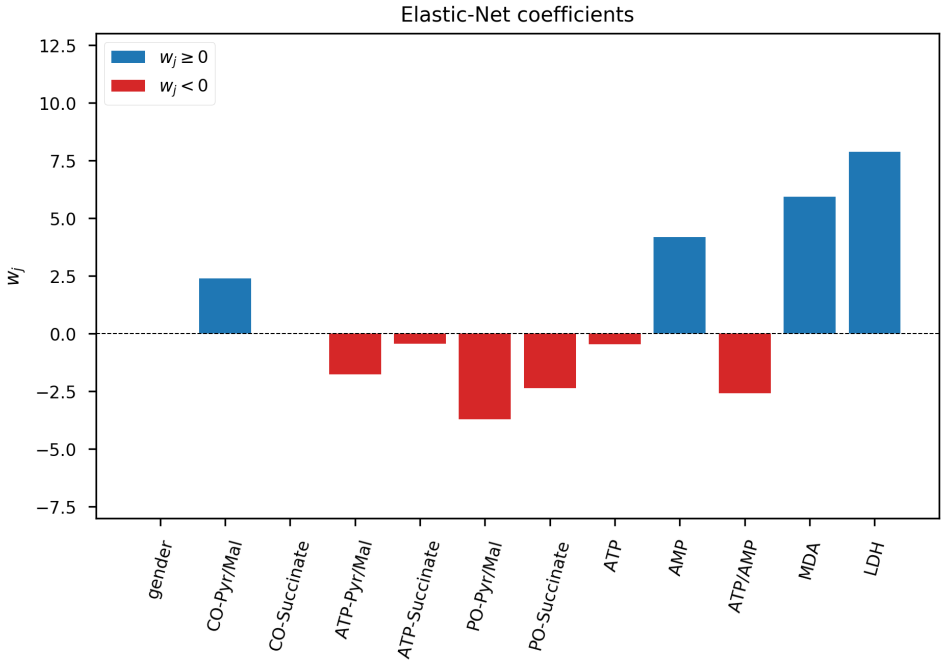


Figure 3.9: A pictorial representation of the vector $\hat{\boldsymbol{w}}_{\ell_1\ell_2}$ obtained fitting a Elastic-Net model on 74 randomly selected training samples of $\mathcal{D}_{\text{aging}}$. Variables associated with positive (*i.e.* directly proportional to the output) and a negative (*i.e.* inversely proportional) weight are represented in blue and red, respectively.

Comparing $\hat{\boldsymbol{w}}_{\ell_1\ell_2}$ in Figure 3.9 with $\hat{\boldsymbol{w}}_{\ell_1}$ in Figure 3.7 we can notice that in the Elastic-Net solution 10 variables have nonzero weight, that is more with respect to the 6 of the Lasso. This is the expected behavior of the added ℓ_2-norm, which helps the model to select groups of collinear variables. Evaluating the Elastic-Net solution on the test set, $\hat{\boldsymbol{w}}_{\ell_1\ell_2}$ achieves MAE $= 8.321$ years explaining the 82.13%, slightly outperforming Lasso, hence ranking first in this little challenge of square loss-based linear regression methods.

As already seen for the Lasso, we can inspect the Elastic-Net weights path, obtained fixing $\alpha = \hat{\alpha}_{\text{cv}}$ for decreasing values of λ, see Figure 3.10. The vertical dashed line in Figure 3.10 corresponds to $\hat{\lambda}_{\text{cv}}$ and it hits the profiles of the weights consistently with the Elastic-Net solution represented in Figure 3.7. The Elastic-Net, compared to the Lasso, produces smoother regularization paths in which the allegedly correlated variables enter the solution earlier.

Another comparison between the behavior of OLS, ridge, Lasso and Elastic-Net regression is presented in Figure 3.11. The four achieved solutions of the same aging regression problem are represented in a scatter plot where horizontal and vertical axis represents their ℓ_2- and ℓ_1-norm, respectively.

As expected,

1. the unpenalized solution $\hat{\boldsymbol{w}}_{\text{OLS}}$ shows the highest values for the two norms and it is placed

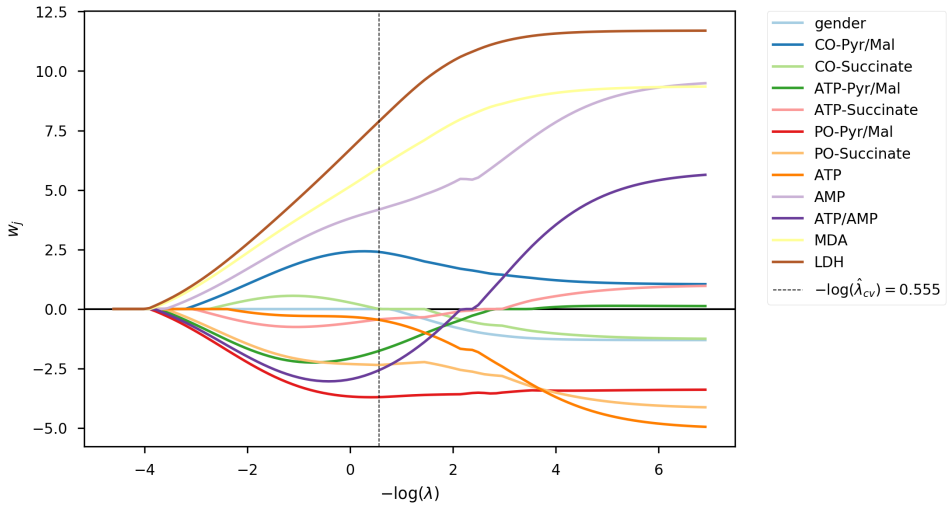


Figure 3.10: Profiles of the Elastic-Net coefficients for the aging problem as λ decreases. The vertical dashed line represents the optimal value $\hat{\lambda}_{\text{cv}}$ estimated by grid-search (5-fold) cross-validation.

in the top-right side of the plot,

2. the ridge solution $\hat{\boldsymbol{w}}_{\ell_2}$ has lowest ℓ_2-norm,
3. the Lasso solution $\hat{\boldsymbol{w}}_{\ell_1}$ has lowest ℓ_1-norm and
4. the Elastic-Net solution shows the lowest values for the two norms and it is placed in the bottom-left side of the plot.

This is consistent with the type of regularization imposed to each method. Interestingly, in this case, the method that performs better on the test set, Elastic-Net, has lowest norms.

The Elastic-Net penalty is not the only method in which sparsity is enforced on a group level. For example, (overlapping) group Lasso and graph Lasso penalties can be applied when the variables are naturally partitioned in (overlapping) groups, or when their interaction can be modeled by a graph. A detailed description of these methods is beyond the scope of this thesis and we refer to [Jacob et al., 2009; Witten and Tibshirani, 2009] and references therein for their comprehensive description.

3.1.1.5 Nuclear norm minimization

Dealing with vector-valued linear regression problems, $L_{2,1}$- and the Frobenius norms are not the only options. Another popular sparsity-enforcing penalty is the nuclear norm (also known as trace norm) that, given the matrix $W \in \mathbb{R}^{d\times k}$, is defined as

$$\|W\|_* = \text{trace}\left(\sqrt{W^T W}\right) = \sum_{j=1}^{\min[d,k]} \sigma_j(W)$$

where $\sigma_j(W)$ is the j^{th} singular value of W. The nuclear norm minimization problem, widely adopted for instance in matrix completion tasks, can be stated as in Equation 3.17.

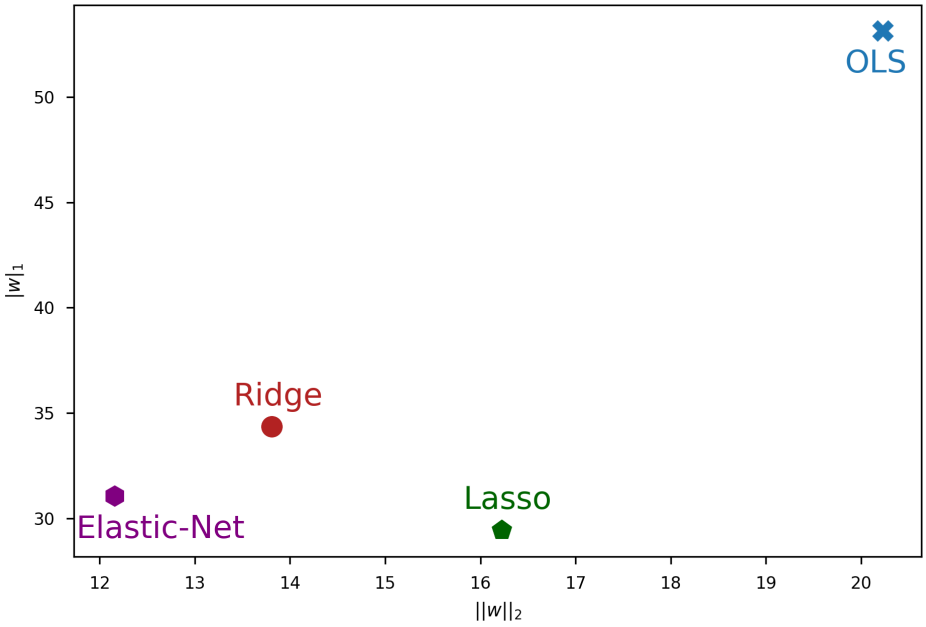


Figure 3.11: A comparison of the value of the ℓ_1 and ℓ_2 norms of the weights obtained by OLS, ridge, Lasso and Elastic-Net.

$$\begin{aligned}\hat{W}_* &= \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{k}(\boldsymbol{x}_i^T\boldsymbol{w}^t - y_i^t)^2 + \lambda \sum_{j=1}^{\min[d,k]}\sigma_j(W) \\ &= \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} \frac{1}{n}\|XW - Y\|_F^2 + \lambda||W||_*\end{aligned} \tag{3.17}$$

With the nuclear norm penalty it is possible to achieve a low-rank solution [Candès and Recht, 2009].

3.1.1.6 Logistic Regression

Logistic regression is one of the most popular linear methods for classification problems[9]. In its unpenalized form, it can be simply posed as the problem of minimizing the logistic loss (see Table 3.1) on a given training dataset. Therefore, logistic regression can be written as in Equation (3.18),

$$\hat{\boldsymbol{w}}_{\text{LR}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\log(1 + e^{-y_i\,\boldsymbol{x}_i^T\boldsymbol{w}}) \tag{3.18}$$

where the labels $y_i \in \{+1, -1\}, \forall i = 1, \ldots, n$. The minimization problem above is convex and differentiable, although as the gradient is nonlinear in $\boldsymbol{w}$, it does not have a closed-form solution. So, unpenalized logistic regression problems are typically solved by (stochastic) gradient descent-like techniques [Boyd and Vandenberghe, 2004; Sra et al., 2012].

Unpenalized logistic regression, although theoretically sound, is somewhat uncommon to meet in recent applied studies. As for the square loss, regularization penalties are typically used. For

[9] As counter-intuitive as it sounds.

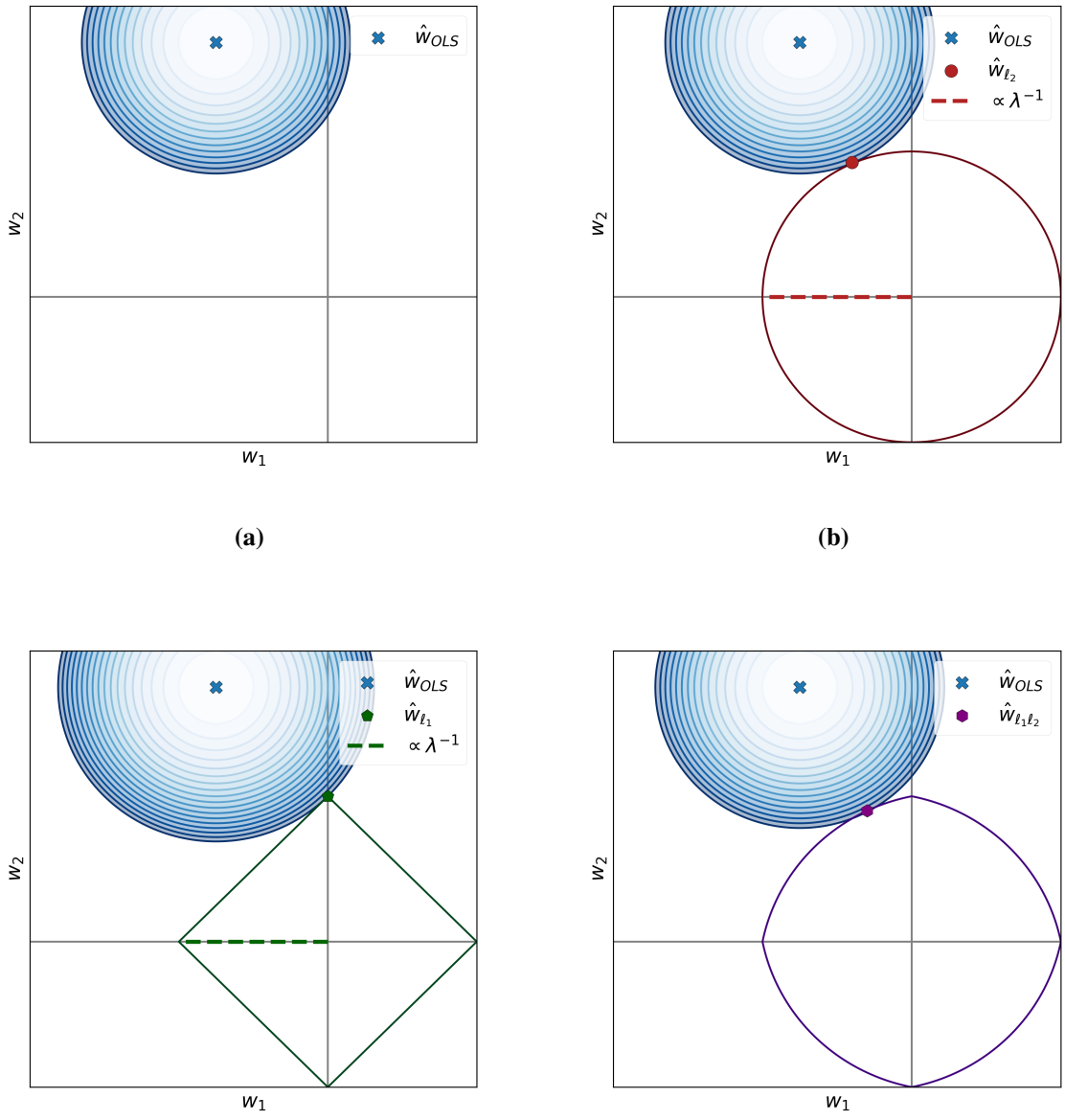


Figure 3.12: Pictorial representation of the contour lines of the square loss in a 2D regression problem with various penalties: (a) ordinary least squares (no penalty), (b) ridge regression (ℓ_2-norm penalty), (c) the Lasso (ℓ_1-norm penalty) and finally (d) the Elastic-Net (ℓ_1- and ℓ_2-norm penalties).

instance, we can have ℓ_2-regularized logistic regression,

$$\hat{\boldsymbol{w}}_{\ell_2} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i\, \boldsymbol{x}_i^T \boldsymbol{w}}) + \lambda \left\|\boldsymbol{w}\right\|_2^2 \tag{3.19}$$

or various forms of sparse logistic regression such as

$$\hat{\boldsymbol{w}}_{\ell_1} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i\, \boldsymbol{x}_i^T \boldsymbol{w}}) + \lambda |\boldsymbol{w}|_1 \tag{3.20}$$

which uses the Lasso penalty, or

$$\hat{\boldsymbol{w}}_{\ell_1\ell_2} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i\, \boldsymbol{x}_i^T \boldsymbol{w}}) + \lambda[\alpha |\boldsymbol{w}|_1 + (1-\alpha)\left\|\boldsymbol{w}\right\|_2^2] \tag{3.21}$$

which uses the Elastic-Net penalty. The sparse logistic regression minimization problem can be efficiently solved either by proximal forward-backward splitting strategies, such as FISTA [Beck and Teboulle, 2009], or by coordinate descent [Wu and Lange, 2008].

Moreover, multi-class classification with logistic regression can be achieved by OVO, AVA approaches, as well as with the multiclass generalization of logistic regression: *softmax* regression [Hastie et al., 2009] (also known as multinomial logistic regression), which can be expressed as

$$\hat{W}_{\text{LR}} = \underset{W\in\mathbb{R}^{d\times k}}{\operatorname{argmin}} -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{t=1}^{k} \mathbb{1}\{y_i = t\} \log\left(\frac{e^{\boldsymbol{x}_i^T \boldsymbol{w}^t}}{\sum_{l=1}^{k} e^{\boldsymbol{x}_i^T \boldsymbol{w}^l}}\right)\right] \tag{3.22}$$

where the indicator function $\mathbb{1}\{\cdot\}$ includes in the functional only the correctly classified samples.

3.1.1.7 Support Vector Machines

Support Vector Machines (SVMs) is a class of powerful ML algorithms that can be written as penalized models and that can used for both regression (SVR) and classification (SVC) problems [Evgeniou et al., 2000]. In the first case, the adopted loss function is Vapnik's ε-insensitive loss:

$$|y - \boldsymbol{x}^T\boldsymbol{w}|_\varepsilon = \begin{cases} 0 & \text{if } |y - \boldsymbol{x}^T\boldsymbol{w}| < \varepsilon \\ |y - \boldsymbol{x}^T\boldsymbol{w}| - \varepsilon & \text{otherwise} \end{cases} \tag{3.23}$$

and in the second case the Hinge loss:

$$|1 - y\, \boldsymbol{x}^T\boldsymbol{w}|_+ = \max[0, 1 - y\, \boldsymbol{x}^T\boldsymbol{w}] \tag{3.24}$$

as reported in Table 3.1, and shown in Figures 3.2a 3.2b. The standard formulation of SVM is penalized by the ℓ_2-norm [Vapnik, 2013]. Therefore, sticking to linear models, the SVR minimization problem can be written as

$$\hat{\boldsymbol{w}}_{\text{SVR}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} |y_i - f(\boldsymbol{x}_i)|_\varepsilon + \lambda \left\|\boldsymbol{w}\right\|_2^2 \tag{3.25}$$

while the SVC minimization problem is

$$\hat{\boldsymbol{w}}_{\text{SVC}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} |1 - y_i f(\boldsymbol{x}_i)|_+ + \lambda \|\boldsymbol{w}\|_2^2 \tag{3.26}$$

where, as usually, λ controls the trade-off between data adherence and smoothing of the solution. Equations (3.25) and (3.26) are known as the *primal* SVM problem. However, in order to generalize the SVM to nonlinear cases (see Section 3.1.2), it is often convenient to transform this minimization problem in its *dual* form. Several algorithms to find the solution of the SVM minimization problem in both primal and dual forms were developed, such as Newton's method or coordinate descent algorithm. See [Smola and Schölkopf, 2004; Shawe-Taylor and Sun, 2011] for an exhaustive review.

The standard formulation of SVM does not cope well with high-dimensional data, as no sparsity-enforcing penalty is adopted. Recently, ℓ_1-norm penalized SVM were proposed as well in [Zhu et al., 2004; Peng et al., 2016].

As for the square loss-based methods, let us apply (ℓ_2-penalized) SVR to the aging problem (introduced in Section 3.1.1.1). The experimental setup in this case is the same already adopted for Lasso and ridge regression (see Sections 3.1.1.3 and 3.1.1.2). The weights $\hat{\boldsymbol{w}}_{\text{SVR}}$ are represented in Figure 3.13. As expected, none of the is exactly zero and they look similar to ridge coefficients in Figure 3.6, as the two model share the same regularization penalty. Evaluating $\hat{\boldsymbol{w}}_{\text{SVR}}$ on the test set, the SVR model has MAE $= 9.280$ and explains the 77.74% of

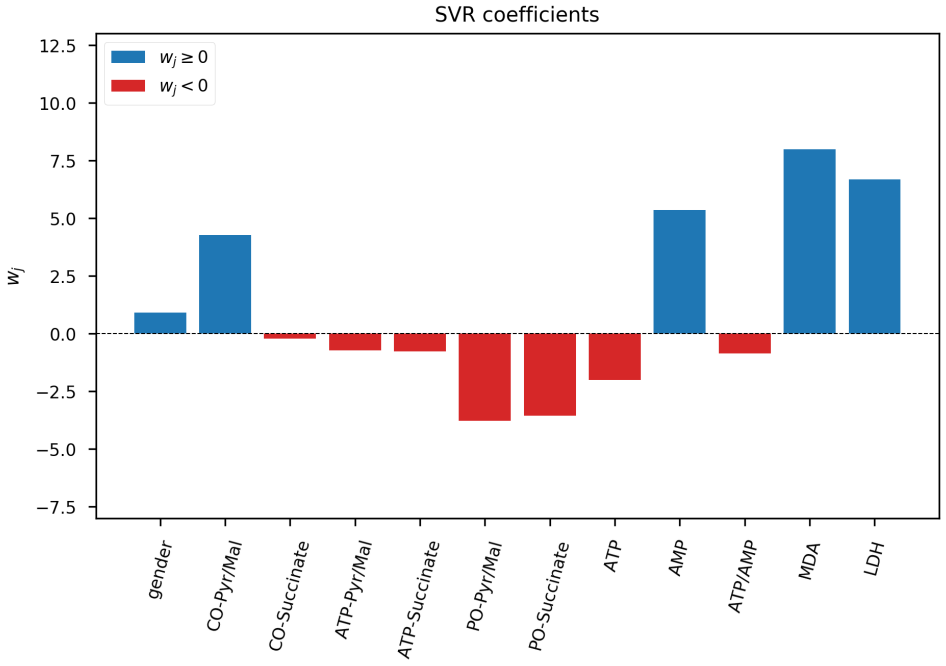


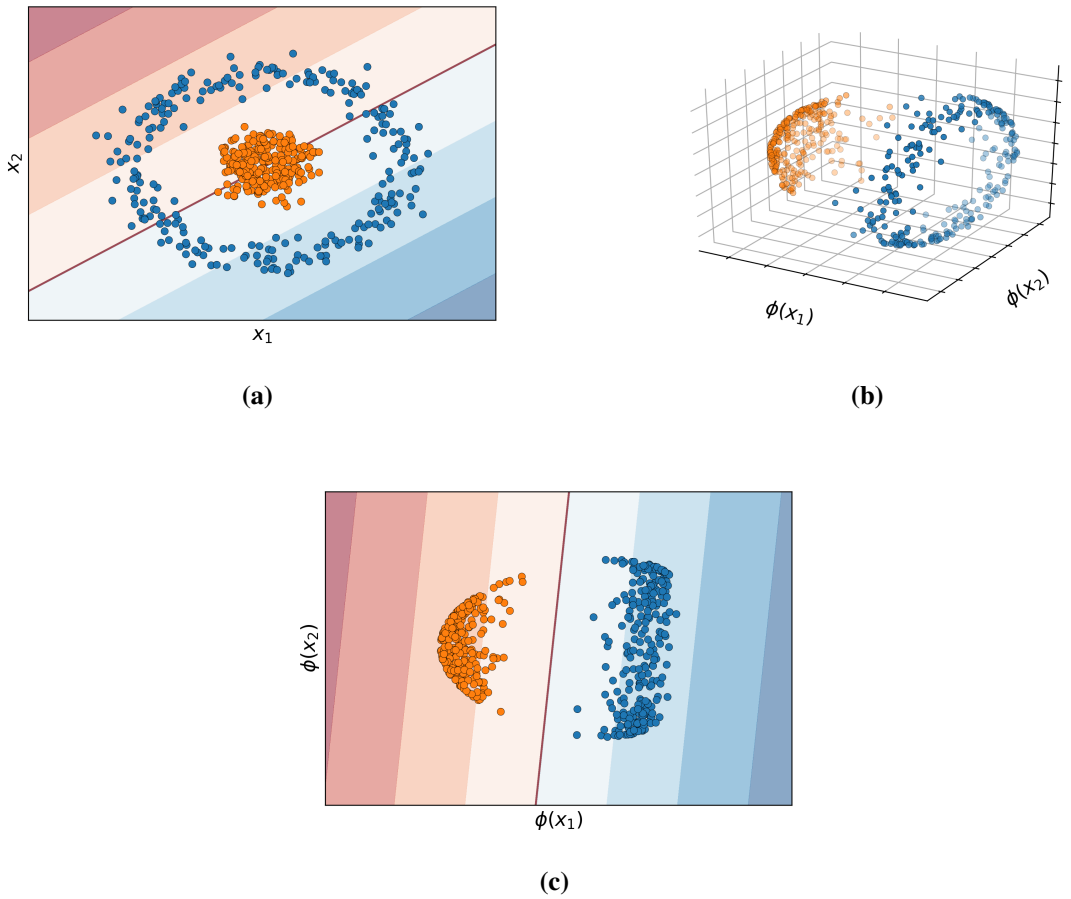
Figure 3.13: A pictorial representation of the vector $\hat{\boldsymbol{w}}_{\text{SVR}}$ obtained fitting a SVR model on 74 randomly selected training samples of $\mathcal{D}_{\text{aging}}$. Variables associated with positive (*i.e.* directly proportional to the output) and a negative (*i.e.* inversely proportional) weight are represented in blue and red, respectively.

the variance.

3.1.2 The kernel trick

For ease of reading, all the regularization methods presented so far were focused on learning linear input-output relationships $y = \boldsymbol{x}^T \boldsymbol{w}$. They can all be extended to the nonlinear case exploiting an elegant mathematical approach called: the *kernel trick*.

The basic idea of the kernel trick is to use a map that projects the features on a higher (possibly infinite) dimensional space in which the prediction problem is, to some extent, easier to solve. An example of kernel trick for nonlinear classification problem is presented in Figure 3.14.



(a) **(b)**

(c)

Figure 3.14: Pictorial representation of its kernel trick. Panel (a) shows a 2D classification problem where the input-output relationship is nonlinear. Panel (b) shows ITS *kernel explosion*, *i.e.* the projection of the 2D problem in a higher (3D) dimensional space in which the two classes are linearly separable, as shown in Panel (c).

We define the nonlinear feature mapping function as $\phi : \mathcal{X} \rightarrow \mathcal{F}$; for instance $\mathcal{X} \in \mathbb{R}^d$ and $\mathcal{F} \in \mathbb{R}^p$ with $p \geq d$, or even $p \gg d$. Given ϕ, the representer theorem [Smola and Schölkopf, 1998], states that the learning problem of Equation (3.2) admits a solution of the form

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha_i$$

where $\boldsymbol{\alpha} \in \mathbb{R}^n$ and the function k behaves like an inner product, *i.e.* it is: symmetric, positive definite, ad it can be defined as in Equation (3.27).

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \cdot \phi(\boldsymbol{x}_j) \tag{3.27}$$

In the ML literature, k is typically called the *kernel* function and it comes in handy to solve the dual version of the learning problems.

Let us see a practical example: kernel ridge regression. We recall that standard ridge regression problem can be rewritten as

$$\hat{\boldsymbol{w}}_{\ell_2} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} J(\boldsymbol{w})$$

where the objective function $J(\boldsymbol{w})$ is

$$J(\boldsymbol{w}) = \frac{1}{n}\left\|X\boldsymbol{w} - \boldsymbol{w}\right\|_2^2 + \lambda\left\|\boldsymbol{w}\right\|_2^2 = \frac{1}{n}(X\boldsymbol{w} - \boldsymbol{y})^T(X\boldsymbol{w} - \boldsymbol{y}) + \lambda\boldsymbol{w}^T\boldsymbol{w}$$

Therefore, the solution of the ridge regression problem can be evaluated by imposing $\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0$ which leads to the following linear system of equations.

$$X^TX\boldsymbol{w} - X^T\boldsymbol{y} + \lambda n\boldsymbol{w} = 0 \Rightarrow (X^TX + \lambda nI)\boldsymbol{w} = X^T\boldsymbol{y}$$

This can either be solved with respect to $\boldsymbol{w}$ (see Section 3.1.1.2) obtaining the solution of the *primal* form

$$\boldsymbol{w} = (X^TX + \lambda nI)^{-1}X^T\boldsymbol{y} \tag{3.28}$$

or it can be rewritten as

$$\boldsymbol{w} = \frac{1}{\lambda n}X^T(\boldsymbol{y} - X\boldsymbol{w}) = X^T\boldsymbol{\alpha}$$

where $\boldsymbol{\alpha} = \frac{1}{\lambda n}(\boldsymbol{y} - X\boldsymbol{w})$ is the *dual* variable. Therefore, the dual formulation of the problem becomes

$$\begin{aligned}\boldsymbol{\alpha} &= \frac{1}{\lambda n}(\boldsymbol{y} - X\boldsymbol{w}) \\ &\Rightarrow \lambda n\boldsymbol{\alpha} = (\boldsymbol{y} - X\boldsymbol{w}) \\ &\Rightarrow \lambda n\boldsymbol{\alpha} = (\boldsymbol{y} - XX^T\boldsymbol{\alpha}) \\ &\Rightarrow (XX^T + \lambda nI)\boldsymbol{\alpha} = \boldsymbol{y} \\ \Rightarrow \boldsymbol{\alpha} &= (K + \lambda nI)^{-1}\boldsymbol{y}\end{aligned} \tag{3.29}$$

where K is the $n \times n$, symmetric and positive semi-definite kernel matrix, with entries $K_i^j = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. In this case $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T\boldsymbol{x}_j$, and this corresponds to the linear kernel. Nevertheless, several other options are available, see Table 3.2 or [Bishop, 2006] for more details. Choosing a different kernel for ridge regression simply boils down to the choice a different formulation for the kernel function k. Plugging the obtained kernel matrix K in Equation (3.29) it is possible to achieve the desired solution.

When the problem cannot be effectively solved with standard kernels, new ones can be easily built. For instance, a simple way to construct a new kernel is to devise an explicit feature map $\phi(\boldsymbol{x})$. Otherwise, new kernels can be built by combining simpler kernel used as building blocks. Furthermore, we can argue that evaluating $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can be compared to a measure of *similarity* between the two input samples. Therefore, when a particular notion of *similarity* between points is available and this can be exploited to design *ad hoc* kernels. Exploiting prior knowledge on the problem to design new kernels is known as kernel engineering [Shawe-Taylor and Cristianini, 2004; Bishop, 2006].

The advantage of the kernel trick is not only statistical, but also computational. Indeed, we may notice that solving the primal problem (*i.e.* in $\boldsymbol{w}$) requires $O(n^3)$ operations, while solving

Table 3.2: Popular kernel functions. RBF stands for Radial Basis Function.

name	formulation
linear	$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$
polynomial	$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}' + c)^d$
RBF	$k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|^2}$
sigmoid	$k(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\gamma \boldsymbol{x}^T \boldsymbol{x}' + c)$

the dual (*i.e.* in $\boldsymbol{\alpha}$) requires $O(d^3)$ operations which is better for large scale and relatively low-dimensional problem.

A similar derivation can also be obtained applying the kernel trick to logistic regression and SVM [Bishop, 2006; Shawe-Taylor and Cristianini, 2004].

3.1.3 Decision trees

Decision trees are simple, yet powerful, methods for regression and classification. They are based on recursive binary splits of the feature space, in which they fit a very simple model, such as a constant. Fitting a decision tree on a given dataset results in partitioning the feature space in *cuboid* regions with edges aligned to the axes. In each region, there is a separate model to predict the target variable. In regression problems this can be the average of the samples falling in the region, whereas for classification problems, it can be the assignment to the most represented class. In particular, when decision trees are applied to classification problems, the feature space partitioning aims at keeping samples with the same labels grouped together. In this thesis, we refer to one of the most popular method for decision tree-based classification and regression, known as CART [Breiman et al., 1984].

An example of application of this technique for classification and regression is shown in Figure 3.15.

Fitting a decision tree implies learning from the data the structure of the tree itself, including the selection of the input variable for each splitting recursion, as well as the optimal splitting threshold. Moreover, the value of the prediction for each region must also be defined.

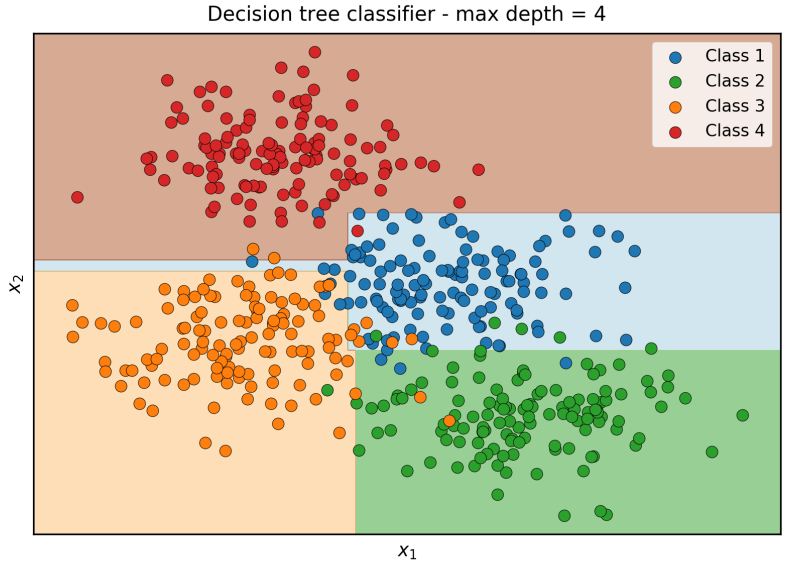
Let us take into account the usual setting: $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n = (X, \boldsymbol{y})$. Even when the number of nodes is fixed, defining at the same time: which variables to split, their splitting threshold and the predictive value is computationally unfeasible as it has a combinatorially large number of possible solutions. Therefore, starting from the root note, *i.e.* the whole input space, and fixing a maximum tree depth, we resort to a greedy approximation.

Let the data at the node m be denoted by Q_m, then for each candidate split $\theta = (j, s_m)$, consisting of the j^{th} feature and the threshold s_m, we define two half-planes as follows.

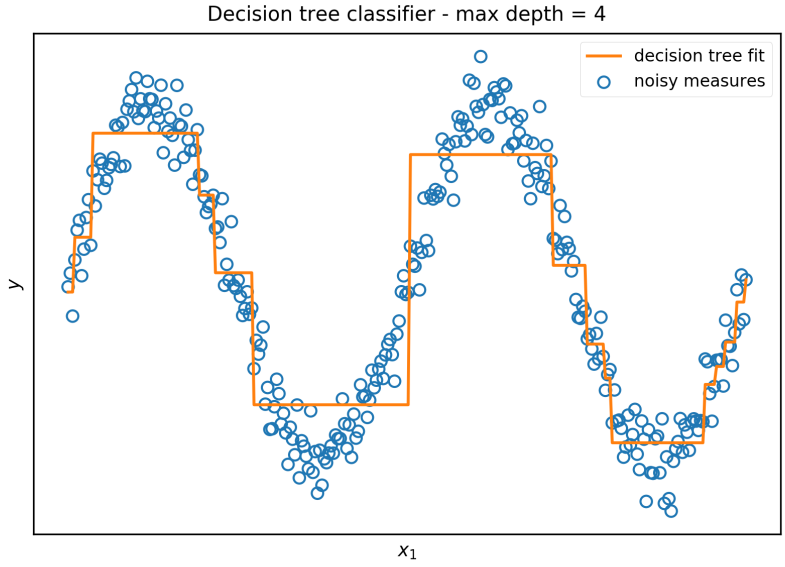
$$Q_{m_1}(\theta) = (\boldsymbol{x}_i, y_i) | \boldsymbol{x}_i^j \leq s_m \text{ and } Q_{m_2}(\theta) = (\boldsymbol{x}_i, y_i) | \boldsymbol{x}_i^j > s_m$$

Then, at each node an impurity measure is evaluated using a suitable impurity function $H(Q_m)$, which depends on the given task. Therefore, we can formulate an objective function like

$$J(Q_m, \theta) = \frac{n_1}{n_m} H(Q_{m_1}(\theta)) + \frac{n_2}{n_m} H(Q_{m_2}(\theta))$$



(a)



(b)

Figure 3.15: Two examples of decision tree applications for: (a) 2D multiclass classification, (b) 1D regression.

where n_1 and n_2 is the number of samples in region Q_{m_1} and Q_{m_2}, respectively, while $n_m = n_1 + n_2$. For each node we then select the best parameter solving the optimization problem in Equation (3.30).

$$\hat{\theta}_m = \underset{\theta}{\operatorname{argmin}}\, J(Q_m, \theta) \tag{3.30}$$

This procedure is recursively applied to $Q_{m_1}(\hat{\theta}_m)$ and $Q_{m_2}(\hat{\theta}_m)$ until the maximum tree depth, that we have fixed at the beginning, is reached.

Dealing with regression problems, a typical impurity measure can be, for instance, the Mean Squared Error (MSE)

$$H_{\text{MSE}}(Q_m) = \frac{1}{n_m} \sum_{\boldsymbol{x}_i \in R_m} \left(y_i - \bar{y}_m\right)^2 \tag{3.31}$$

where $\bar{y}_m = \frac{1}{n_m} \sum_{\boldsymbol{x}_i \in R_m} y_i$, is the average output of Q_m, *i.e.* the training samples in region R_m; or the Mean Absolute Error (MAE)[10], which is simply defined as below.

$$H_{\text{MAE}}(Q_m) = \frac{1}{n_m} \sum_{\boldsymbol{x}_i \in R_m} |y_i - \bar{y}_m| \tag{3.32}$$

On the other hand, dealing with (multiclass) classification problems, where the classes are identified by the index $k = 1, \ldots, K$, typical impurity measures are, for instance, the Gini criterion

$$H_{\text{Gini}}(Q_m) = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) \tag{3.33}$$

where $p_{mk} = \frac{1}{n_m} \sum_{\boldsymbol{x_i} \in R_m} \mathbb{1}(y_i = k)$ is the proportion of class k observation on the region R_m; or the Cross-Entropy (CE)

$$H_{\text{CE}}(Q_m) = -\sum_{k=1}^{K} p_{mk} \log(p_{mk}) \tag{3.34}$$

which is an impurity measure also heavily applied as loss function for other learning machines (see Appendix A).

The main free parameter of this model is the maximum depth of the tree, *i.e.* its size. Deeper trees are capable of modeling complex input-output relationships with respect to shallow ones. Obviously, too deep trees can overfit the training data. Therefore, the optimal depth of the tree should be estimated from the data via, for instance, grid-search cross-validation.

Decision trees are important in biomedical data science applications as they are easily interpretable and capable of modeling nonlinear input-output relationships. Indeed, the prediction is given following a number of binary decisions which, in some cases, may mimic the way doctors perform diagnosis on their patients. Relatively small decision trees can actually be represented graphically. For instance, applying a this method to the aging problem presented in Section 3.1.1.1, we obtain the tree shown in Figure 3.16. As usually, the main free parameter of the model, *i.e.* the maximum depth of the tree, is chosen via grid-search cross-validation (and it is 3, in this case).

A very well-known weakness of decision trees is their instability. In fact, decision trees are not robust to noise affecting the input data. Even minor perturbations of a single feature may generate a different split which propagates down to all the splits below. In practice, this is alleviated by the use of decision trees as base learners of ensemble strategies (see Section 3.1.4). More details on this learning paradigm and its applications can be found reading Chapter 9 of [Hastie et al., 2009] or Chapter 14 of [Bishop, 2006].

[10] We previously came across this measure when evaluating the performance of learning methods on the aging problem, see Sections 3.1.1.1, 3.1.1.2, 3.1.1.3, *etc*.

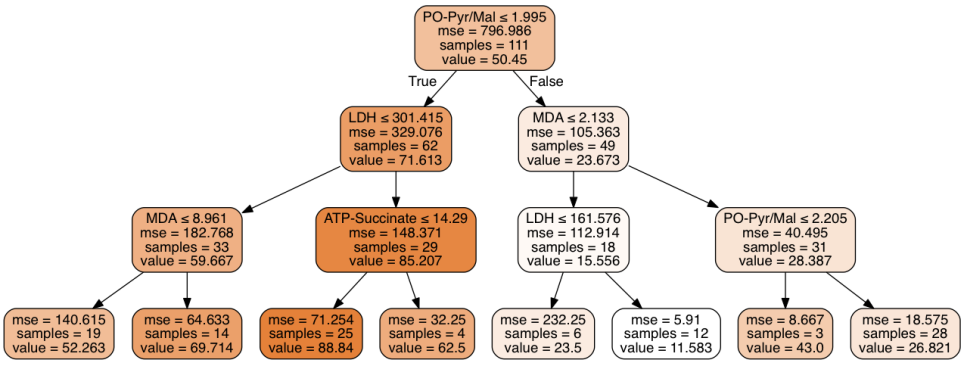


Figure 3.16: The graph structure learned from the aging problem (see Section 3.1.1.1). The maximum depth of the tree is 3 and it is chosen via grid-search cross-validation.

3.1.4 Ensemble methods

The key idea of ensemble methods is to build a predictive model by aggregating a collection of multiple *base learners* that are trained to solve the same problem [Zhou, 2012].

Bagging is a common ensemble strategy that consists in fitting multiple models $f_b(\boldsymbol{x})$ for $b = 1, \dots, B$ each one on a *bootstrap* sampling[11] $\mathcal{D}_b$ of the training dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$. For each sample $\boldsymbol{x}_i$, the bagging estimate $\hat{f}(\boldsymbol{x}_i)$ is obtained by combining the predictions of the base learners $\hat{f}_b(\boldsymbol{x}_i)$. For instance, in case of classification tasks, the bagged model may select the most predicted class casting a vote among the B base learners. On the other hand, for regression problems, the bagging estimate can be a (weighted) mean, or the median, of the predictions of the B base learners. Decision trees (see Section 3.1.3) are typical base learner for bagged estimators.

Boosting is another popular ensemble strategy that, unlike bagging, performs predictions by sequentially fitting a collection of base learner that cast a weighted vote [Hastie et al., 2009]. At each boosting step, the weight corresponding to samples that were misclassified at the previous iterations increases. Therefore, each successive classifier is somewhat forced to learn the relationships between input and output that were previously missed. From a theoretical standpoint, it would be possible to boost any learning machine, nevertheless boosting methods are mainly used with decision trees as base learners [Hastie et al., 2009].

3.1.4.1 Random Forests

Random Forests (RF) are ensembles of decision trees, proposed by Leo Breiman in the early 2000s [Breiman, 2001] to ameliorate the instability of decision trees. In RF each tree is grown on a bootstrap sample from the training data, typically to its maximum depth. To increase robustness to noise and diversity among the trees, each node is split using the best split among a subset of features randomly chosen at that node. The number of features on which the trees are grown is one of the free parameters of the model. The final prediction is made by aggregating the prediction of M trees, either by a majority vote in the case of classification problems, or by averaging predictions in the case of regression problems. Another important free parameter of the model is the number of trees in the ensemble. RF are a bagging approach, which works on the assumption that the variance of individual decision trees can be reduced by averaging

[11] Random sampling with replacement.

trees built on many uncorrelated subsamples. Moreover, increasing the number of trees in the ensemble, RF does not overfit the data. Therefore, this parameter is typically chosen to be *as large as possible*, consistently with the available hardware and computational time [Breiman, 2001].

RF can provide several measures of *feature importance*, computed by looking at the increase in prediction error when data for a feature is permuted while all other features remain unchanged. Feature selection based on RF if most often performed using one of these measures. However, several techniques for applying regularization to random forests have been proposed. These techniques broadly fall under two categories:

1. cost-complexity pruning, which consists in limiting tree depth, resulting in less complex models [Ishwaran et al., 2008; Kulkarni and Sinha, 2012];
2. Gini index penalization [Deng and Runger, 2013; Liu et al., 2014].

In addition, [Joly et al., 2012] proposed to use an ℓ_1-norm to reduce the space-complexity of RF.

RF naturally handles numerical, ordinal and categorical variables, multiple scales and non-linearities. They also require little parameter tuning. This makes them popular for the analysis of diverse types of biological data, such as gene expression, Genome-wide Association Studies (GWAS) data or mass spectrometry [Qi, 2012]. In practice, feature selection schemes that rely on RF may be unstable [Kursa, 2014], therefore feature selection stability measures must be adopted to avoid drawing inconsistent conclusions.

Let us see what happens when RF are applied to the aging problem (see Section 3.1.1.1). The experimental setup is identical to the one previously applied for ridge and Lasso regression (see Section 3.1.1.2 and 3.1.1.3), where the only parameter optimized via grid-search cross-validation is the maximum number of features each tree can grow on (and in this case it is chosen as 4). The number of trees in the ensemble is fixed to 500. The RF model on the test set achieves MAE $= 6.321$ and Explained Variance EV $= 88.17\%$. Figure 3.17 shows the feature importance measure estimated by this model.

As we can see, the features that are most relevant for RF are comparable to the ones selected by Elastic-Net (see Figure 3.12d). For RF models it is interesting to investigate the effect of an increasing value for the number of trees in the forest. As shown in Figure 3.18, increasing the number of trees in the forest does not lead to overfit. In fact, test error (Figure 3.18a) and explained variance (Figure 3.18b), after approximately 250 trees, reach a plateau region instead of growing.

3.1.4.2 Gradient Boosting

Gradient boosting [Friedman, 2001] (GB) is one of the most widely applied boosting methods in biological problems. This technique iteratively combines the predictions obtained by several base learners, such as decision trees, into a single model. The key idea behind GB is that, under some general hypothesis on the cost function, boosting can be seen as an iterative gradient method for numerical optimization. In particular, in GB at each boosting iteration a new base learner is fitted on the residuals obtained at the previous boosting iteration. GB has several desirable properties [Mayr et al., 2014], such as its capability to learn nonlinear input-output relationship, its ability to embed a feature importance measure (as RF) and its stability in case

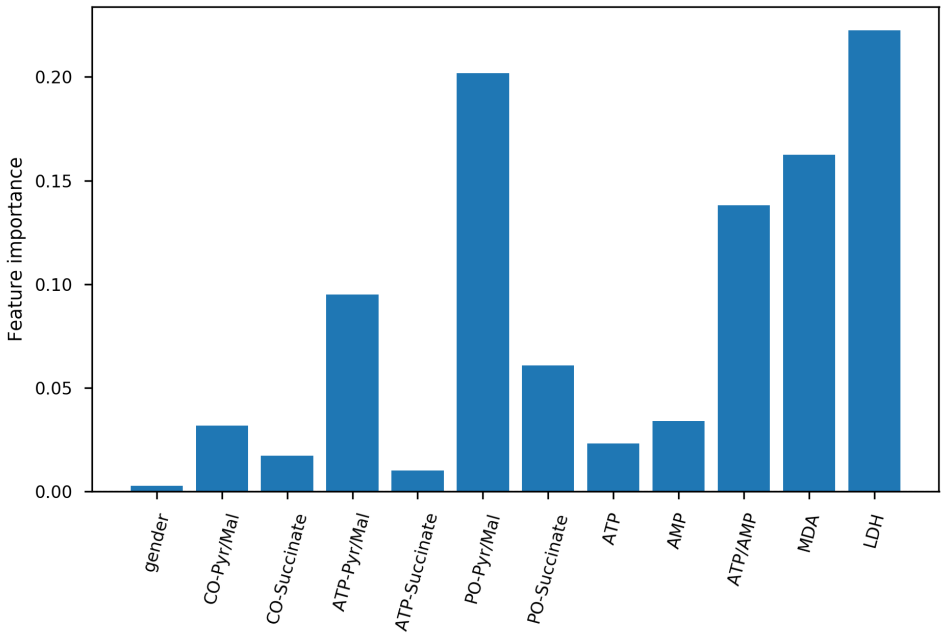


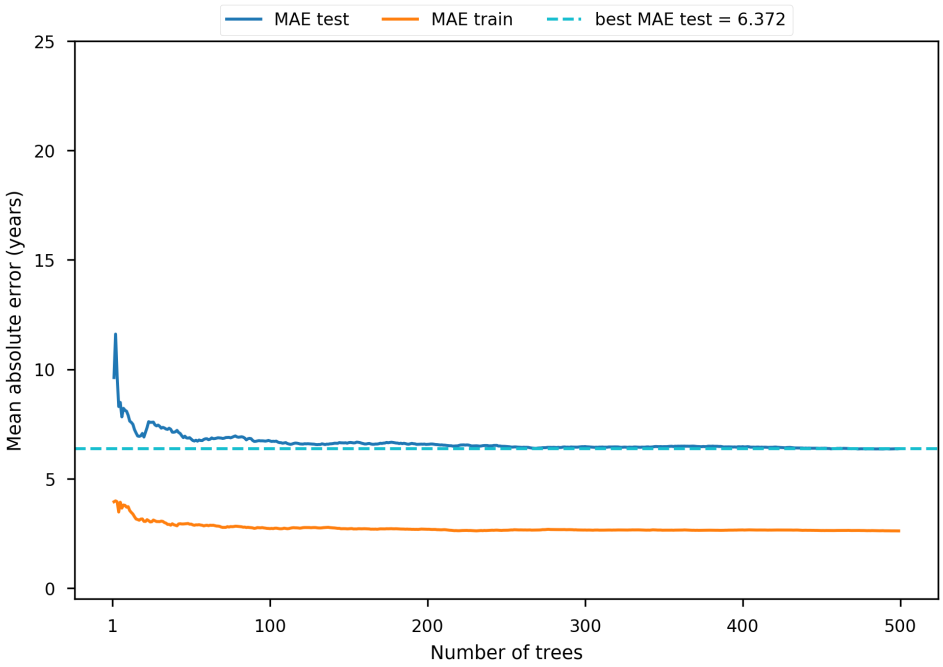
Figure 3.17: A pictorial representation of the feature importance achieved by a RF model with 500 trees growing on 4, out of 12, features.

of high-dimensional data [Buehlmann, 2006]. When used for feature selection, GB shows interesting performance when casted in the stability selection framework [Meinshausen and Bühlmann, 2010], leading to an effective control of the false discovery rate [Everitt, 2006].

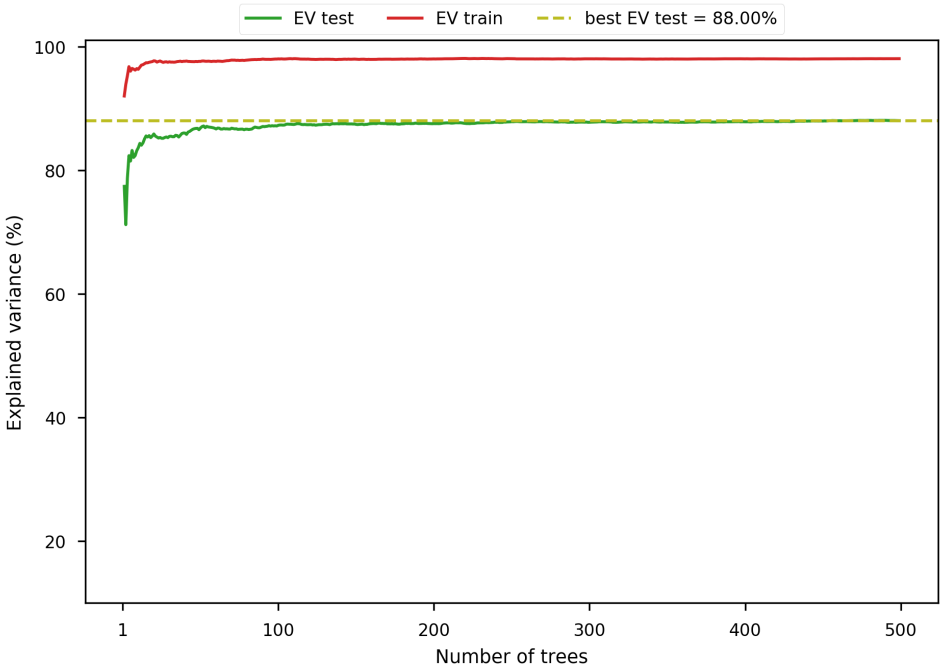
As for most of the learning machines, GB may suffer of overfitting. The main regularization parameter to control is the number of boosting iterations M, *i.e.* the number of base learners, fitted on the training data. This is typically optimized by cross-validated grid-search, information criteria-based heuristics [Tutz and Binder, 2006, 2007] or early stopping [Prechelt, 1998]. Regularization in GB can also be controlled by shrinking the contribution of each base learner by a constant $0 < \nu < 1$ that controls the learning rate of the boosting procedure. In order to achieve comparable predictive power, smaller values of ν imply larger number of M, so there is a trade-off between them. As usually, the base learners are decision trees and another important parameter to tune is their maximum depth [Hastie et al., 2009].

In a recent paper [Lusa et al., 2015], the authors show that in high-dimensional balanced binary classification problems, if the base learner is likely to overfit the training data, the use of *Stochastic GB* [Friedman, 2002] is preferable. The latter is a modified version of the original method, where each base learner is fitted on a random extraction without resubmission of a fraction η of the training data, where η is another regularization parameter to choose.

Let us see what happens when GB is used to tackle the aging problem (see Section 3.1.1.1). The experimental setup is identical to the one previously applied for Elastic-Net regression (see Section 3.1.1.4), where the two parameters optimized via grid-search cross-validation are the maximum depth of the trees (in a range from 3 to 20) and the number of boosting iterations (with a maximum value of 500). The selected tree depth is 8 and the number of boosting iterations is 31. The GB model on the test set achieves MAE $= 7.955$ and Explained Variance EV $= 84.88\%$. Figure 3.19 shows the feature importance measure estimated by this model. As we can see, the features that are most relevant for GB are comparable to the ones relevant for RF (see Figure 3.17).



(a)



(b)

Figure 3.18: The effect of the number of trees in a RF model in terms of MAE, panel (a), and EV, panel (b). As expected, larger forests do not lead to overfit, *i.e.* the training error does not reach zero.

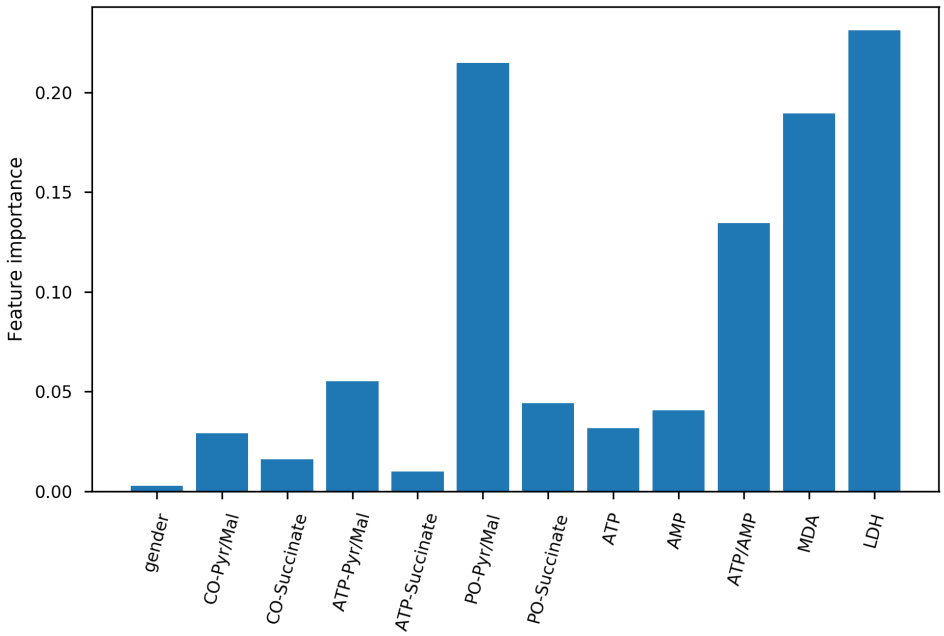


Figure 3.19: A pictorial representation of the feature importance achieved by a GB model with learning rate 0.1 after 31 boosting iterations. Each tree has maximum depth of 11 and it grows on 4, out of 12, features.

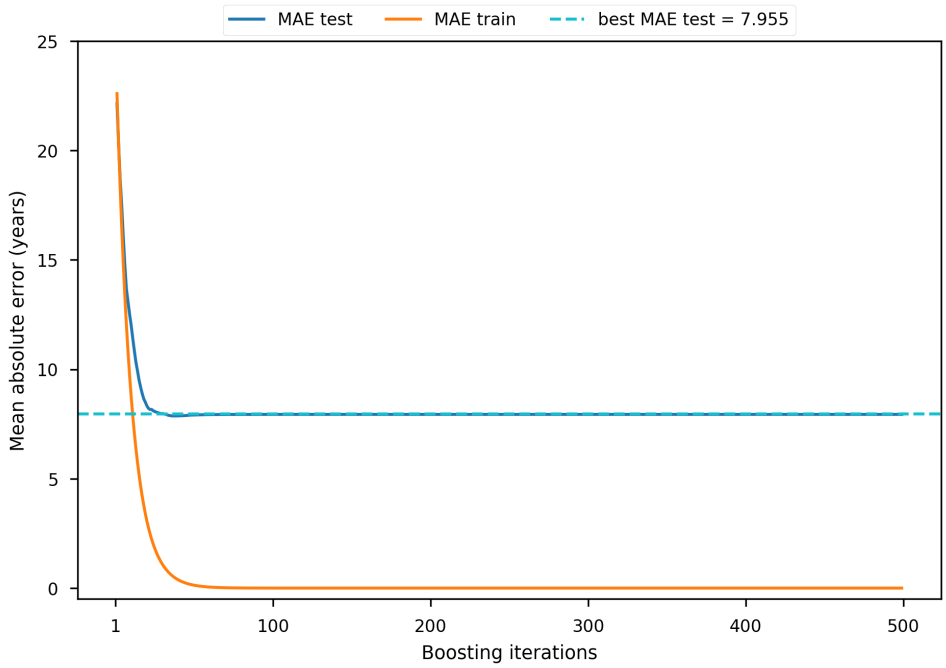
As already seen for RF, we shall investigate the effect of increasing boosting iterations for GB. As shown in Figure 3.20, after approximately 50 boosting iterations the model reaches perfect overfit of the training set, *i.e.* zero training error (Figure 3.20a) and 100% trained explained variance (Figure 3.20b).

3.1.5 Deep learning

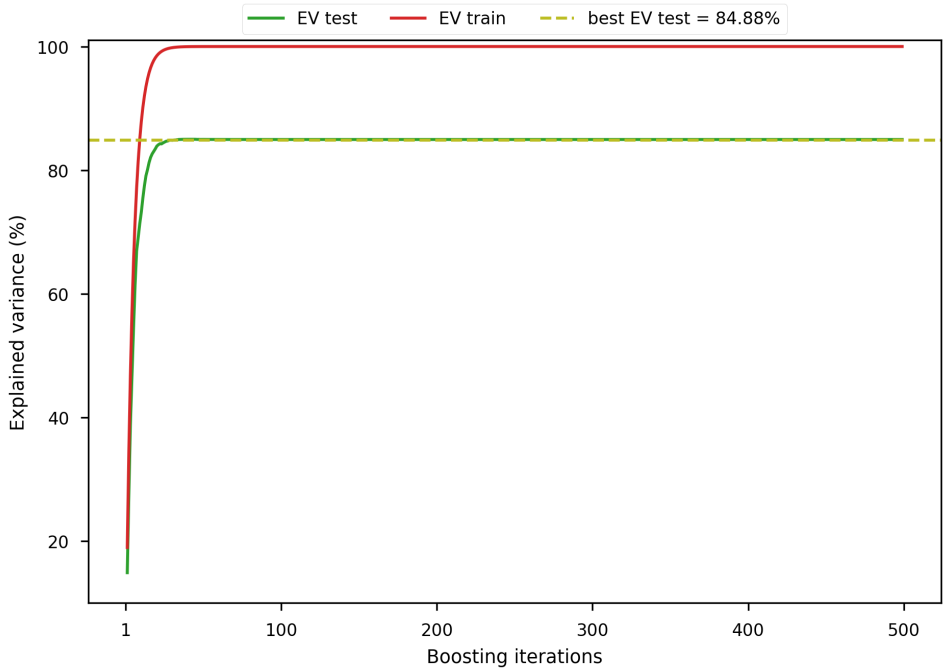
Deep Learning (DL) is a branch of ML that, in the recent past, is becoming extremely appealing thanks to the high predictive power that it has empirically shown on real-world problems. This section is definitely not an attempt to summarize the heterogeneous plethora of the DL methods presented in literature so far. Indeed, even a dedicated thesis may not be enough to accomplish this task. In this section we will only get a grasp on the mechanisms behind the supervised DL methods applied in Part II[12].

One of the main characteristics of DL methods is that, starting from raw data, they aim at learning a suitable data representation (see Section 3.3) and a prediction function, at the same time. DL methods stand on the shoulders of the classical, and shallow, Neural Networks of the 80s, and they can actually be seen as their extension. In DL the the final prediction is achieved by composing several layers of nonlinear transformations. The intuition behind DL method is that, starting from raw data, their multi-layer architecture can achieve representations at a more abstract level, leading to top performance in prediction tasks. DL architectures can be devised to tackle binary or multiclass classification [Angermueller et al., 2016; Leung et al., 2014] as well as single or multiple output regression [Chen et al., 2016; Ma et al., 2015].

[12] Unsupervised DL methods were presented in literature as well, but as they are not used in Part II, their discussion is not presented here. We recall to [Chollet, 2018; LeCun et al., 2015] for a more comprehensive overview.



(a)



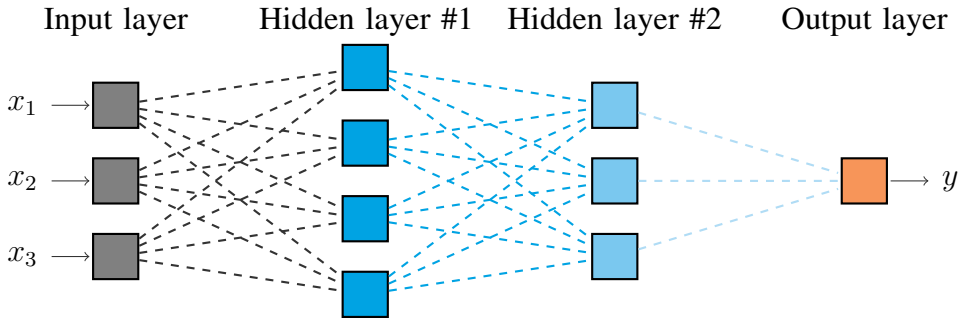
(b)

Figure 3.20: The effect of the number of boosting iterations in a GB model (learning rate 0.1) in terms of MAE, panel (a), and EV, panel (b). As expected, increasingly large boosting iterations lead to perfect overfit, *i.e.* zero training error.

3.1.5.1 Multi-Layer Perceptron

In order to understand the principles guiding DL methods, we sketch here the ideas behind the most basic one: the Multi-Layer Perceptron (MLP), also known as deep Feedforward Neural Network. Typically, MLPs are structured as fully connected graphs organized in *layers* that can be of three different types: *input*, *hidden* and *output* (see Figure 3.21). Each node of the graph is called *unit*. The number of units in the input layer matches the dimensionality of the raw data (d), while number and type of output units are related to the learning task. For multiclass classification problems, with K classes, the output layer has K units, each one representing the probability of the input sample to be assigned to one specific class. On the same line, for multiple-output regression problems, with K tasks, each one of the K units corresponds to the prediction for a given task. Size and number of the hidden layers can be chosen according to the problem at hand and the available computational resources.

Figure 3.21: A pictorial representation of the structure of an MLP with two hidden layers having four and three hidden units, respectively. According to the nature of the output layer, this network topolgy can be adopted either for regression or binary classification problems starting from raw samples in a three-dimensional space.



In MLPs the information flows through the graph from the input to the output. Each layer l transforms its input data $\boldsymbol{x}^{l-1}$ by composing an affine transformation and an activation function $\sigma(\boldsymbol{x})$ that acts element-wise on its input vector. In other words, defining as p_{l-i} the number of units in the layer $l-i$, the layer l applies the transformation

$$\boldsymbol{x}^l = \sigma(\boldsymbol{x}^{(l-1)} W_l + b_l)$$

where $W_l \in \mathbb{R}^{p_{l-1} \times p_l}$ and $\boldsymbol{b} \in \mathbb{R}^{p_l}$ are the weights of the model that are learned from the data.

The function $\sigma(\boldsymbol{x})$ is known as *activation function* and it can be defined in different ways. In classical neural networks, activation functions are modeled as sigmoids (*e.g.* $\sigma(x) = \tanh(x)$, $\sigma(x) = (1 + e^{-x})^{-1}$) whilst in modern DL architectures the most used activation function is the Rectified Linear Unit: $f(x) = \max(0, x)$.

Particular attention must be paid when fitting deep models as they can be prone to overfit the training set [Angermueller et al., 2016]. The network topology itself defines the *degrees of freedom* of the model: deeper and wider networks can approximate well very complicated input-output relationship, but also the noise affecting the data. Although, tuning the number of hidden layers and their size is not the recommended strategy to prevent from overfitting, as it may lead to suboptimal solutions.

Regularization in MLPs can be controlled by penalizing the weights of the network. A common regularization strategy consists in adding an ℓ_2-norm penalty in the objective function, as in Equation (3.2). In the DL community this procedure is known as weight decay [Krogh and

Hertz, 1992]. Although less common, the ℓ_1-norm can also be adopted as regularization penalty, as in [Leung et al., 2014].

Training MLPs, and deep networks in general, consists in solving a minimization problem via suitable optimization algorithms [Ruder, 2016]. All these methods iteratively update the weights of the network in order to decrease the training error. Another fundamental regularization strategy is the *Early stopping* [Prechelt, 1998]. This consists in interrupting the fitting process as soon as the error on an external validation set increases [Angermueller et al., 2016].

Another common regularization strategy in DL is *Dropout* [Srivastava et al., 2014]. This techniques consists in temporarily deactivating a defined number of randomly chosen units of the network at training phase. This reduces the degrees of freedom of the model and it implicitly allows to achieve an ensemble of several smaller networks whose predictions are combined. The use of dropout alone can improve the generalization properties, as in [Chen et al., 2016], but it is usually adopted in combination with weight decay or other forms of regularization, as in [Leung et al., 2014].

Training MLPs implies dealing with non-convex optimization problems which are usually tackled via stochastic gradient descend, or similar methods [Ruder, 2016; Sra et al., 2012].

MLP architectures are also used as final layers of another class of DL methods called: Convolutional Neural Network (CNN). CNN are a different class of DL methods that exploit convolutional masks with weights estimated from the data to learn a suitable feature representation. This class of methods is largely applied to computer vision and image recognition problems. Describing CNN architectures is beyond the scope of this thesis, as they are not mentioned in the Part II. A thorough description of CNN can be found here [Chollet, 2018; Goodfellow et al., 2016].

Let us see what happens when a relatively simple MLP is applied to the aging problem (see Section 3.1.1.1). The experimental setup is the same used for RF in Section 3.1.4.1, where the only parameter optimized via grid-search cross-validation is the weight decay constant. The resulting MLP has 1 hidden layer with 1024 units, ReLu activation and constant weight decay 0.001. Such MLP is trained with early stopping and, after 329 iterations reaches MAE $= 9.14$ explaining the 78.84% of the variance of the test set. The relatively worse performance of this method, compared, for instance, to RF, may be due to the fact that this dataset has a reduced number of samples. It is known that training DL models from scratch, in order to reach optimal solutions, it is necessary to have more training samples than conventional (shallow) learning machines.

3.1.5.2 Long-Short Term Memory Network

Long-Short Term Memory Networks (LSTMs) belong to the class of Recurrent Neural Networks (RNNs) [Goodfellow et al., 2016].

LSTM use special units, called *memory cells*, which have the ability to learn long-term dependencies [LeCun et al., 2015]. Instead of just applying an element-wise nonlinear function, such as sigmoid or hyperbolic tangent, to the affine transformation of inputs and recurrent units, LSTM cells have an internal self-loop together with a system of *input*, *output* and *forget gates* [Goodfellow et al., 2016]. Compared to standard RNNs, LSTMs can be trained via gradient descent as they do not suffer from the vanishing/exploding gradient problem [Bengio et al., 1994]. In the recent past, this deep learning architectures demonstrated to achieve promising

results in several sequences learning tasks, such as speech recognition [Graves and Schmidhuber, 2005], image captioning [Vinyals et al., 2015], or time-series forecast [Schmidhuber et al., 2005].

3.2 Feature selection

In real-world data science challenges, it is very often the case that the number of acquired variables d is high. It may even happen that d heavily outnumbers the number of collected samples n. This setting is usually referred to as *large d small n* scenario, or simply $n \ll d$. Practical examples of this case can be easily found in the context of applied life sciences, *e.g.* gene expression, DNA sequencing, proteomics, spectroscopy, medical imaging and so on.

In such cases, the main goal of the analysis is often to identify a meaningful subset of relevant variables that are the most representative of the observed phenomenon. In ML, this is known as variable/feature selection and several techniques addressing this task were presented so far [Guyon et al., 2002]. Variable selection does not only increase the prediction power of the learning machine, but it also promotes model interpretability, that is crucial in biology [Altmann et al., 2010].

Variable selection techniques are traditionally organized in three groups: filter, wrapper and embedded methods [Guyon and Elisseeff, 2003].

Filters In these techniques, *irrelevant* features are filtered out from the feature set before any learning step. A fair majority of filter approaches is based on some variable ranking criteria that are usually based on the definition of a scoring function $F(\boldsymbol{x}^j, \boldsymbol{y})$, where $\boldsymbol{x}^j$ is the $j-th$ variable and $\boldsymbol{y}$ is the output vector. Usually high values of the scoring function correspond to highly relevant features. Once the scoring function is evaluated for each feature, the entire feature set is sorted by means of their score. Varying the selection threshold one may generate different set of nested features. The selection of one of such subsets can be done after an actual learning step, *i.e.* when a reliable estimate of the prediction error is available. Variable selection techniques that belong to this class may be based on some statistical scores, *e.g.* t-test, ANOVA, Pearson correlation, or some entropy measures, *e.g.* the mutual information [Everitt, 2006]. Statistical score-based algorithms may fail when the assumptions on the probability distribution function does not hold for the available data. On the same line, information theoretic techniques should need the probability distribution of the data, which is usually unknown.

Single variable classifiers is another class of methods that, to some extent, may fall into this family. They still define a scoring function F, but such function is a measure of the predictive power of each features. For instance, in a classification scenario one may select the list of variables that, individually, have best accuracy, or false positive or false negative classification rate. Moreover, in case of regression problems best residual sum of squares or R^2 score may be used (see Section 3.4.3). A major drawback of this class of methods is that features that are not relevant by themselves will be discarded even if they would have been relevant used in combination with some other ones.

Wrappers The key idea behind the algorithms of this second family is that the variable selection process should not only depend on input and outputs, but also on a specified learning algorithm. These algorithms select a feature subset according to the prediction

performance achieved by some learning algorithms evaluated on subsamples of the original feature set. ML methods are here used as black boxes that, given some input, simply return a prediction performance score. Exhaustive search over the whole space of all possible features subset, that has cardinality 2^d, is clearly unfeasible even in low dimension. In order to cope with this problem several *greedy* techniques have been proposed. Among them, the most popular algorithms are the *matching pursuit* (also known as forward stage-wise selection) [Cai and Wang, 2011] and the Recursive Feature Elimination scheme (RFE) [Guyon et al., 2002]. While the first starts from an empty vector and greedily incorporate the most relevant features, RFE starts with the whole feature set and then iteratively exclude the least promising variables.

Embedded The main characteristic of these methods is that they can learn and perform variable selection simultaneously. Embedded variable selection can be achieved by two families of learning machines: tree-based and regularization methods. Decision trees can be used for variable selection as only some of the input variables are used to partition the feature space and variables not involved in any split are can be discarded, see Section 3.1.3. Moreover, ensemble of decision trees natively offer a feature importance measure 3.1.4. On the other hand, regularization methods as extensively discussed in Section 3.1.1, can easily be used for variable selection by exploiting sparsity enforcing penalties. Only the variables having nonzero weight will be considered as selected by the model. Regardless of the learning machine, regularization can be introduced in several ways and it is of fundamental use in order to achieve the following desired properties [Okser et al., 2014]:

- identify models with good generalization properties, even with a limited amount of collected samples;
- achieve solutions that are robust to noise;
- learn the data structure when unknown;
- exploit prior knowledge on the data structure;
- reduce the feasible set in order to help solving inverse problems.

Assessing the stability of the selected variables is a major issue that is typically faced when performing variable selection on high-dimensional problems. This observation is even more evident in $n \ll d$ cases. Classical variable selection strategies, *e.g.* the Lasso (see Section 3.1.1.3), are sensitive to random sample extraction. This reflects in having a, possibly, very different list of selected variables each time the algorithm is evaluated on a cross-validation training split of the data. Variable selection stability is widely studied in literature and different strategies to achieve stable variable selection were proposed in literature [Kuncheva, 2007; De Mol et al., 2009a; Hofner et al., 2015; Nogueira and Brown, 2016].

In particular, in [Meinshausen and Bühlmann, 2010] the authors propose the so-called *stability selection* framework. For a given variable selection strategy (let us think about the Lasso, for simplicity) this can be seen as an extension of standard regularization paths (see Section 3.1.1). For increasing values of the free parameter of the chosen variable selection strategy (*e.g.* the regularization parameter λ), a *selection probability* is estimated for each variable. Such probability is estimated by evaluating the variable selection algorithm on several random extractions of a portion of the training set. This produces the so-called *stability path*. A stable list of selected variables can therefore be obtained by including the variables that, for a given value of the free parameter, are selected with high probability. As shown in the original paper,

this framework is not heavily influenced by the values of free parameter and probability threshold. However, they can be chosen by validation on an external validation set, which is fundamental in order to avoid selection bias [Ambroise and McLachlan, 2002]. Similar strategies are also explored in [De Mol et al., 2009b; Barbieri et al., 2016]

3.3 Unsupervised learning

In Section 3.1 we started the discussion of supervised learning methods with a metaphorical comparison with the human ability of learning from examples. Actually, in order to understand concepts, human beings need way less supervision than modern learning algorithms. In some circumstances it may be even the case that little/no supervision at all is needed to accomplish a task.

For instance, let us say that a kid growing with a Brittany dog at home visit some friend having a Bernese Mountain Dog. The kid, so far, has only being taught what his dog looks like and these two dog breeds definitely look different one another. Nevertheless, the kid will certainly be able to understand that both of them are animals and, in particular, dogs. Who taught the kid the appearance of every possible dog breed?

The learning paradigm in which we try to infer some interesting properties, or recurrent patterns from unlabeled samples is known as unsupervised learning and, quoting Kevin P. Murphy, "*Unsupervised learning is arguably more typical of human and animal learning*." [Murphy, 2012].

In this section, given a set of input data $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$, with samples represented as d-dimensional vectors $\boldsymbol{x}_i \in \mathbb{R}^d$ ($\forall i = 1, \ldots, n$), our interest will be directed toward two different goals: (i) grouping similar objects together (Section 3.3.1) and (ii) achieving a meaningful low-dimensional representation of our data (Section 3.3.2). In both cases we will try to infer the properties of the probability distribution $p(\boldsymbol{x})$ from which the collected data are assumed to be drawn.

3.3.1 Cluster analysis

Cluster analysis, also known as *data segmentation* or *clustering*, is the class of unsupervised learning algorithm which aim at grouping together similar objects, by some definition of *similarity*.

This task is fairly common in biomedical studies, where cluster analysis can be used, for instance, to perform patients stratification or to find groups of correlated biological measures, such as gene expression, blood exams, and so on. This section presents an overview on the most popular clustering algorithms[13].

3.3.1.1 The k-means algorithm

k-means is the most intuitive strategy for cluster analysis.

[13] A good reference for the clustering algorithms proposed in literature is *Section 2.3. - Clustering* of the SCIKIT-LEARN documentation http://scikit-learn.org/stable/modules/clustering (last visit 2018-01).

Given n samples $\boldsymbol{x}_i \in \mathbb{R}^d$, the goal of k-means is to partition the data space into K non-overlapping groups containing similar samples. Let $\boldsymbol{\mu}_k \in \mathbb{R}^d$ be the *prototypical* vector associated with the k^{th} cluster, we can think of this as the center of mass of the cluster, *i.e.* its centroid. Fixing the number of cluster K, the output of the k-means algorithm is a set of cluster centroids as well as the assignment of each sample to a single, *i.e.* the closest, cluster. For ease of notation, we introduce a binary indicator function r_k defined as

$$r_k(\boldsymbol{x}_i) = \begin{cases} 1 & \text{if } \boldsymbol{x}_i \text{ is assigned to cluster } k \\ 0 & \text{otherwise} \end{cases}$$

which is known as 1-of-K encoding. Then, we can define the objective function

$$J(\boldsymbol{\mu}) = \sum_{i=1}^{n} \sum_{k=1}^{K} r_k(\boldsymbol{x}_i) d(\boldsymbol{x}_i, \boldsymbol{\mu}_k)$$

where $d(a, b)$ is some distance measure. In case $d(a, b)$ is the squared Euclidean distance, which is the typical choice for k-means, the objective function can be rewritten as in Equation (3.35).

$$J(\boldsymbol{\mu}) = \sum_{i=1}^{n} \sum_{k=1}^{K} r_k(\boldsymbol{x}_i) \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_k \right\|_2^2 \tag{3.35}$$

Our goal is to find the centroids $\boldsymbol{\mu}_k$ for $k = 1, \ldots, K$ that minimize $J(\boldsymbol{\mu})$. We can argue that, given the best centroids, the samples can be simply associated to the cluster with the nearest centroid. So, let us focus on the problem of finding $\boldsymbol{\mu}_k$. Fixing $r_k(\boldsymbol{x})$ for all k, the objective function in Equation (3.35) is quadratic in $\boldsymbol{\mu}$, thus the minimization problem can be easily solved setting its derivative to zero, see Equation (3.36).

$$2 \sum_{i=1}^{n} r_k(\boldsymbol{x}_i)(\boldsymbol{x}_i - \boldsymbol{\mu}_k) = 0 \Rightarrow \boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} r_k(\boldsymbol{x}_i) \boldsymbol{x}_i}{\sum_{i=1}^{n} r_k(\boldsymbol{x}_i)} \tag{3.36}$$

Intuitively, $\boldsymbol{\mu}_k$ corresponds to the mean of the points assigned to the cluster k.

The k-means algorithm starts with random centroids. Then the points are assigned to the cluster having the nearest centroid. Successively, the centroids position is updated. At each step the value of $J(\boldsymbol{\mu})$ is reduced, hence the convergence of k-means is assured. Nevertheless, the algorithm may converge to a local minimum, achieving a suboptimal solution.

Defining an optimal value of K for any dataset is still an open problem. The k-means algorithm works with a number of cluster fixed a priori[14]. K can be given from some prior knowledge on the problem, or it can be estimated following some heuristics. Otherwise, an estimate of K can be defined by optimizing data-driven coefficients like the silhouette score (see Section 3.4.3), or some information criteria, such as AIC or BIC [Bishop, 2006]. These estimates can be done with or without cross-validation [Fiorini et al., 2017b].

As shown in [Arthur and Vassilvitskii, 2007], smart choices of the initial position of the clusters can substantially improve the result. For instance, the initial $\boldsymbol{\mu}_k$, for $k = 1, \ldots, K$, can be chosen to be *far away* from each other. This solution is called k-means++ and it follows the steps summarized below.

[14] Some relevant work on automatic cluster discovery is presented in [Ball and Hall, 1967; Pelleg et al., 2000; Muhr and Granitzer, 2009].

1: choose the first centroid $\boldsymbol{\mu}_1$ at random from the samples in the dataset;
2: compute the distance between $\boldsymbol{\mu}_1$ and all the other points in the dataset: $D = d(\boldsymbol{x}_i, \boldsymbol{\mu}_1)$ $\forall i = 1, \ldots, n$;
3: choose a new data point as centroid $\boldsymbol{\mu}_2$ following a weighted probability distribution $\propto D^2$;
4: update the distance vector as $D = \min[d(\boldsymbol{x}_i, \boldsymbol{\mu}_1), d(\boldsymbol{x}_i, \boldsymbol{\mu}_2)]$ $\forall i = 1, \ldots, n$;
5: repeat steps 3 and 4 until k centroids are defined;
6: use this centroids initialization to run standard k-means.

Even though this initial seeding comes with an increased computational time at the beginning, then the k-means algorithm converges super fast, leading to a decreased overall computational time. Moreover, the achieved solution has considerably better final cluster reconstruction, compared to standard k-means, hence k-means++ initialization is usually preferable.

3.3.1.2 Spectral clustering

Spectral clustering is a popular clustering algorithm initially proposed in [Shi and Malik, 2000] and then further analyzed and explained in [Ng et al., 2002; Von Luxburg, 2007]. Assuming the usual setup, the problem of clustering can be translated in the problem of inferring a *similarity graph* $\mathcal{G} = (V, E)$, where each data point $\boldsymbol{x}_i$ corresponds to a vertex and the weight s_{ij} of the edge connecting $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, is proportional to the similarity between the two points. For instance, in the so-called ε-neighborhood graph, the vertexes corresponding to $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are connected if $s(\boldsymbol{x}_i, \boldsymbol{x}_j) > \varepsilon$, for a given ε. Our aims is then to achieve a graph structure where the edges between points of the same group is high and it is low between points belonging to different groups.

Therefore, let us consider a set of points $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$, the steps of the unnormalized spectral algorithm are summarized as follows[15].

1: fix a symmetric and non-negative similarity function $s(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and compute the similarity matrix $S = \begin{bmatrix} 1 & \ldots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{1n} & \ldots & 1 \end{bmatrix}$;
2: construct a similarity graph $\mathcal{G}$ from S, let W be its adjacency matrix and D its degree matrix;
3: compute the unnormalized graph Laplacian $L = D - W$;
4: compute the first k eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ of L and organize them column-wise in the matrix $V \in \mathbb{R}^{n \times k}$;
5: for $i = 1, \ldots, n$ let $\boldsymbol{\psi}_i \in \mathbb{R}^k$ be the vector corresponding to the i^{th} row of V;
6: cluster the points $\{\boldsymbol{\psi}_i\}_{i=1}^n$ using k-means into K clusters.

As for the k-means algorithm, finding K is a difficult problem. A common solution for spectral clustering is the *eigengap heuristics*, described in [Von Luxburg, 2007].

[15] Normalized version this algorithm is identical but the formulation of L which is substituted by the normalized graph Laplacian [Von Luxburg, 2007].

3.3.1.3 Hierarchical clustering

As suggested by the name, hierarchical clustering techniques produces hierarchical (tree) representation of a given dataset, where each level of the hierarchy consist in groups of samples (clusters). Individual data points can be found at the leaves of the learned tree, while all the data points are grouped in a single cluster at the root level.

There are two main families of hierarchical clustering: the *agglomerative* (bottom-up) and the *divisive* (top-down) approaches. Both of them are iterative techniques. The first, at each iteration, groups together the most similar points, hence creating more and more large clusters, while the divisive strategy works the other way around: by iteratively dividing groups of points. From now on we will refer to agglomerative clustering only, but similar considerations can be followed for the divisive approach as well.

Hierarchical clustering techniques have two main degrees of freedom: the choice of point-wise and group-wise similarity, also known as *affinity* and *linkage*, respectively. The first is used to compute the initial value of the similarity matrix

$$S_0 = \begin{bmatrix} 0 & \dots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{1n} & \dots & 0 \end{bmatrix}$$

while the second is used to update it once the two most similar data points are collapsed into a single cluster $S_0 \in \mathbb{R}_+^{n \times n} \to S_1 \in \mathbb{R}_+^{n-1 \times n-1}$. While any symmetric and non-negative function is a valid affinity, the choices of for linkage function are more interesting and worth mentioning. Let G and H be two clusters, there are four main linkage options:

Single $\quad l(G,H) = \max_{\substack{\boldsymbol{x}_i \in G \\ \boldsymbol{x}_j \in H}} s(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Complete $\quad l(G,H) = \min_{\substack{\boldsymbol{x}_i \in G \\ \boldsymbol{x}_j \in H}} s(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Average $\quad l(G,H) = \frac{1}{n_G n_H} \sum_{\boldsymbol{x}_i \in G} \sum_{\boldsymbol{x}_j \in H} s(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Ward $\quad l(G,H) = \frac{n_G n_H}{n_G + n_H} \left\| \boldsymbol{\mu}_G - \boldsymbol{\mu}_K \right\|^2$

where n_A and $\boldsymbol{\mu}_A$ are the number of points and the centroid of the cluster A, respectively.

Interestingly, finding the number of clusters K translates here into the problem of pruning the tree.

3.3.2 Dimensionality reduction and feature learning

Most of the ML methods aim at learning some hidden relationship in a set of data. Typically, the dataset at hand is a collection of d measures from n samples $\boldsymbol{x}_i \in \mathcal{X} \in \mathbb{R}^d$ and we want our ML algorithms to crunch such data in order to perform some learning task, *e.g.* classification, regression, clustering and so on.

It is often the case that the feature space $\mathcal{X}$ is not the most suitable to our purposes. Sometimes projecting the data in a different space $\mathcal{X}' \in \mathbb{R}^p$, maybe with $p < d$ or even $p \ll d$, can greatly

ease the further learning steps. We have already explored a similar concept talking about the kernel trick in Section 3.1.2.

In particular, heavily decreasing the dimensionality of the data is helpful as it achieves (lossy) data compression, reducing the computational time of further analyses. Moreover, dimensionality reduction eases data visualization and understanding. These aspects are really important when dealing with large scale biomedical data.

This section presents a compact overview of the dimensionality reduction and unsupervised feature learning algorithms that are most relevant with the biomedical data science projects presented in the second part of this thesis.

3.3.2.1 Principal component analysis

Principal Component Analysis (PCA) [Jolliffe, 2002] is probably the most popular dimensionality reduction technique. PCA is typically defined as linear projection of the data onto a lower dimensional space where the variance is maximized.

In order to understand PCA in general, we first focus on how to find the first principal component, *i.e.* the main direction in which the data shall be projected in order to preserve as much variance as possible. Let $\boldsymbol{u}_1$ be the d dimensional vector which identifies the direction of the first principal component. As we are not interested in the magnitude of this vector, but rather in its direction, without loss of generality we can assume $\boldsymbol{u}_1^T \boldsymbol{u}_1 = 1$.

The projection of each data point $\boldsymbol{x}_i$ on the direction identified by $\boldsymbol{u}_1$ can easily be found as $\boldsymbol{x}'_i = \boldsymbol{u}_1^T \boldsymbol{x}_i$. We shall notice that the variance of the projected data can be expressed as

$$\sigma^2_{\boldsymbol{x}'_i} = \frac{1}{n} \sum_{i=1}^{n} \left[\boldsymbol{u}_1^T \boldsymbol{x}_i - \boldsymbol{u}_1^T \bar{x} \right]^2 = \boldsymbol{u}_1^T S \boldsymbol{u}_1$$

where $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i$ is the empirical mean of the samples and S is the data covariance matrix, defined as $S = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \bar{x})(\boldsymbol{x}_i - \bar{x})^T$. Therefore, finding $\boldsymbol{u}_1$ can be casted in the following optimization problem

$$\max_{\boldsymbol{u} \in \mathbb{R}^d} \left[\boldsymbol{u}_1^T S \boldsymbol{u}_1 + \lambda_1 (1 - \boldsymbol{u}_1^T \boldsymbol{u}_1) \right]$$

where λ_1 is a Lagrangian multiplier. By setting the derivative with respect to $\boldsymbol{u}_1$ equal to zero we can easily write

$$S \boldsymbol{u}_1 = \lambda_1 \boldsymbol{u}_1 \Rightarrow \boldsymbol{u}_1^T S \boldsymbol{u}_1 = \lambda_1$$

which means that the variance is maximum when $\boldsymbol{u}_1$ is the eigenvector of S with maximum eigenvalue λ_1. The eigenvector $\boldsymbol{u}_1$ is hence known as the principal components.

Additional components can be iteratively found by choosing new directions, orthogonal to the ones already identified, that maximize the projected variance.

3.3.2.2 Multi-dimensional scaling

Multi-dimensional scaling (MDS) [Borg and Groenen, 2005] is a dimensionality reduction technique which aims at finding a low-dimensional projection that maximally preserves the pairwise distance between data points.

Adopting the Euclidean distance metric, MDS is similar to PCA. However, the MDS algorithm can be extended to a wide range of similarity measures.

3.3.2.3 Isomap

Isomap [Tenenbaum et al., 2000] is a non-linear dimensionality reduction technique that can be seen as an extension of MDS.

In this algorithm, the similarity between the data points are evaluated as geodesic distances along some manifold. At first the algorithm finds the k nearest neighbors of each sample. Then it builds a graph where each sample is a node and the edges of neighboring nodes are weighted with the Euclidean distance of the corresponding samples. Finally, the geodesic distance is approximated by summing the weights of the edges corresponding to the shortest path between each pair of points. The final low-dimensional projection is obtained performing MDS on such distance matrix.

3.3.2.4 t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (t-SNE) [Van der Maaten and Hinton, 2008] is a relatively recently proposed nonlinear dimensionality reduction technique specifically developed for high-dimensional data visualization. t-SNE aims at learning a low-dimensional data projection that reflects the original data similarities as much as possible.

The algorithm has two main steps. At first t-SNE devises pairwise probability distributions over high-dimensional samples such that the similar ones are picked together with high probability. Then, the algorithm defines another probability distribution on a low-dimensional map identified by minimizing the Kullback–Leibler divergence between the two distributions. The minimization problem is solved by gradient descent.

3.4 Model selection and evaluation

The previous chapters present an overview of ML techniques and algorithms that are most relevant with the data science challenges presented in the second part of this thesis. The described algorithms are very different for both goals and mathematical structure. However, they all share a common trait: the presence of one, or more, free parameters. These free parameters must be specified before the actual learning step and, in ML literature, they are usually referred to as *hyperparameters*[16]. As opposed to the model parameters, which are learned from data, hyperparameters are tuning knobs that must be user-defined.

In order to better understand this difference, let us see a practical example: a regularized least square approach (such as Ridge or Lasso). In this case, the weights of the input variables w_j (for $j = 1, \ldots, d$) are learned from training data and they are the model *parameters*; whereas the regularization parameter λ is the only hyperparameter of the model. This simple example has only one hyperparameter, but more complicated models, such as gradient boosting or deep

[16] This term is inherited from the filed of statistics, where an hyperparameter is a parameter of a prior distribution probability. This sometimes generates confusion and possible misunderstanding in technical literature.

architectures (see Section 3.1.5 and 3.1.4.2), present way more hyperparameters to tune (*e.g.* depth and size of the tree, learning rate, number of layers, *etc*).

In order to successfully carry out real-world data science challenges, ML practitioners need two key ingredients: (i) reliable model assessment strategies and (ii) robust protocols for model selection. The first point often boils down to the use of a robust cross-validation technique to estimate the generalization error [Molinaro et al., 2005], whilst the second is typically tackled with (exhaustive) search over a grid of possible values. In ML jargon, each different choice of an hyperparameter set corresponds a different model. The process of finding the best performing set of hyperparameters is typically called model selection.

3.4.1 Model assessment strategies

The performance of a ML model can be assessed by estimating its generalization error, that is a measure of the performance the model is expected to achieve on future data. As this is, in general, not possible, the predictive power of a given learning machine should be evaluated on previously unseen data, simulating future observation. Resampling protocols are the most popular class of techniques to simulate the acquisition of future data [Molinaro et al., 2005] and these methods can be successfully applied to estimate the generalization error.

Moreover, particular attention should be paid when the data analysis pipeline embeds a variable selection step. In fact, as shown in [Ambroise and McLachlan, 2002], an error estimated on the same set of data used for variable selection can be heavily affected by *selection bias*. This form of overfitting leads to an overoptimistic estimate of the model performance. Hence, in order to avoid too optimistic estimations of the generalization error, ML practitioners should rely on nested resampling protocols, as shown for instance in [De Mol et al., 2009b].

This section briefly presents the cross-validation resampling routines that are most relevant for the second part of this thesis.

Monte Carlo cross-validation This strategy repeatedly splits the n samples of the dataset in two mutually exclusive sets. For each split, $n \cdot (1/\nu)$ samples are labeled as *validation* set and the remaining $n \cdot (1 - 1/\nu)$ as *training* set. The data points of the two sets are randomly sampled without replacement from the entire dataset. At each repetition, the learning machine is fitted on the training set and its predictive power is assessed by evaluating a performance metric (see Section 3.4.3) on the independent test set. If needed, a variable selection step can be part of the training phase. Monte Carlo cross-validation gives the opportunity to assess not only the value of the generalization error, that can be obtained, for instance, averaging across iterations, but also to infer its probability distribution. However, a major drawback of this method is its computational burden that increases with the number of repetitions, that could be arbitrarily large.

Bootstrap This cross-validation strategy is similar to the Monte Carlo approach, but it presents a substantial difference. In this strategy, new versions of the dataset are generated by uniform random extraction with replacement of n' samples out of n. The extracted data are used as training set, while the left out samples are used for validation. Typically the value of n' is chosen close to n. For large n, bootstrapped datasets are expected to have $\approx 63.2\%$ of unique samples, while the remaining 36.8% being duplicates.

***K*-fold cross-validation** This strategy splits the data in K non-overlapping subsets that are iteratively kept aside and used to estimate the prediction error of a model trained on the remainder of the data. The final estimation of the prediction error on future data can be done by averaging on the K estimates obtained during the cross-validation procedure. The number of splits K is here limited by the cardinality of the dataset at hand. The major advantage of a K-fold with respect to a Monte Carlo cross-validation strategy is its much lighter computational workload. However, as the number of splits K cannot be arbitrarily large, this method is not suitable for estimating the probability distribution of the generalization error. When $K = n$ the validation set has only 1 sample and this strategy degenerates in the so-called *leave-one-out* cross-validation scheme.

3.4.2 Model selection strategies

As anticipated at the beginning of this section, hyperparameters tuning is still considered, by the majority of the ML practitioners, more an *art* than a science.

Nevertheless, different statistically sound techniques to perform hyperparameters optimization were presented over the years . At the time of writing, model selection is a hot-topic in ML research. It often happens that several ML solutions adopted to tackle data science competitions[17] are based on the same model. What really makes the difference, and defines the most competitive prediction strategy, is to adopt an efficient and effective model selection strategy.

The model selection procedure applied in the data science challenges described in this thesis are carried out by one of the strategies described below.

Grid-search cross-validation This is a very straightforward hyperparameter optimization strategy which, for given ML model and cross-validation scheme, takes as input a multi-dimensional grid of possible hyperparameter values and creates a different model for each hyperparameter tuple. Then, for each model, a given performance metric (see Section 3.4.3) is estimated by the adopted cross-validation scheme. The best hyperparameter tuple is chosen as the one achieving top cross-validation prediction score. In other words, this strategy systematically searches for the best hyperparameters across multiple possibilities. This brute-force algorithm typically returns top performing models, but its is very computational demanding.

Randomized-search cross-validation This can be considered as an alternative to standard grid-search. The main difference is that, instead of trying each possible hyperparameters tuple of a given grid, this strategy generates a fixed number of models each having hyperparameters randomly sampled from given distributions. All these models are then assessed using some cross-validated performance metric (see Section 3.4.3) and the best hyperparameters tuple is chosen as the one corresponding to the top performing model. Compared to standard grid-search, this algorithm may perform slightly worse, but it comes with a way less demanding computational burden.

Interestingly, both these hyperparameters optimization strategies involve independently fitting multiple models multiple times. In practice, this embarrassingly parallel problem is in practice often accelerated by parallel computing strategies. This allows to reduce the computational

[17] Such as the ones hosted on the Kaggle website: https://www.kaggle.com (last visit 2018-01).

time needed to test several models exploiting the computing capabilities of modern multi-core processors, or multi-node high-performance computing facilities, where available. More details about computational requirements and implementation of modern ML models will be further discussed in Section 3.5.

3.4.3 Performance metrics

Dealing with real-world datasets, it is very important to have a quantitative and robust way to assess the performance of the applied algorithm. Of course, according to the learning task, different performance metrics should be used. We have already met some of them in the previous section. Here, a comprehensive overview of the most representative performance metrics is provided. To ease readability, the performance metrics are organized according to the corresponding learning task.

Regression The ideal metric for a regression task is a quantitative measure of how distant the predictions $\hat{\boldsymbol{y}}$ are with respect to the actual output values $\boldsymbol{y}$. The following list shows different ways to measure such distance.

(i) *Mean Absolute Error* (MAE): this is probably the most intuitive way to assess the regression performance. MAE is very well suited for single task regression and time-series forecasting problems. MAE is scale-dependent, hence not suitable for comparisons across different domains.

$$\text{MAE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{3.37}$$

(ii) *Mean Squared Error* (MSE): widely applied to assess the regression performance, MSE is a second order measure, hence it incorporate bias and variance of the model. MSE of an unbiased estimator corresponds to its variance. MSE is scale-dependent.

$$\text{MSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{3.38}$$

(iii) *Root Mean Squared Error* (RMSE): this is simply defined as the square root of MSE.

$$\text{RMSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{3.39}$$

(iv) *Residual Sum of Squares* (RSS): this data discrepancy measure is mainly used to measure the performance of least squares methods, as it corresponds to the square loss function.

$$\text{RSS}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{3.40}$$

(v) *Coefficient of determination* (R^2): this metric measures the proportion of the variation of the output that is correctly predicted. This metric is not scale dependent, the best possible score is $\text{R}^2 = 1.0$, while a constant model predicting the average of the

Table 3.3: A prototypical confusion matrix for a binary classification problem. The two classes are encoded as ± 1, the number of true positive examples is n_+, whereas the number of true negative examples is n_-; hats denote estimated values. The extension for multiclass problems is straightforward, once a positive class is defined.

		Predicted value $\hat{\boldsymbol{n}}_{+\mathbf{1}}$	**Predicted value** $\hat{\boldsymbol{n}}_{-\mathbf{1}}$
True value	$\boldsymbol{n}_{+\mathbf{1}}$	True Positive	False Negative
	$\boldsymbol{n}_{-\mathbf{1}}$	False Positive	True Negative

output achieves $\mathrm{R}^2 = 0.0$. R^2 is defined as below, where $\bar{y}$ is the average output, $\bar{y} = \frac{1}{n}\sum_{j=1}^{n} y_j$.

$$\mathrm{R}^2(\hat{\boldsymbol{y}}, \boldsymbol{y}) = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.41}$$

(vi) *Explained Variance* (EV): similar to R^2, this metric is not scale-dependent and it can be used to compare the outcome of different regression problems. EV is often expressed as percentage, best possible value is 100%, lower is worse. EV is defined as below, where $\mathrm{Var}(\boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$.

$$\mathrm{EV}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = 1 - \frac{\mathrm{Var}(\boldsymbol{y} - \hat{\boldsymbol{y}})}{\mathrm{Var}(\boldsymbol{y})} \tag{3.42}$$

Classification Compared to regression, assessing the performance of a classifier seems like a much trivial task. It turns out that this is actually not true; assessing the classification performance by different metrics may let you draw very different conclusions about the quality of $\hat{\boldsymbol{y}}$. What follows is a list of most important classification metrics, where the meaning of True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) is illustrated in the prototypical confusion matrix (for binary classification) represented in Table 3.3.

(i) *Accuracy*: this is the most intuitive classification metric, it simply consists in the fraction of correct predictions. The accuracy measure can be used for binary as well as multiclass classification tasks. The best possible score is 1.0, while the expected score of a random classifier is the fraction of the most represented class.

$$\mathrm{Acc}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(\hat{y}_i = y_i) \tag{3.43}$$

(ii) *Balanced accuracy*: for heavily unbalanced binary classification problems this metric is preferable with respect to the standard classification accuracy, as the balanced accuracy of the random classifier is 0.5 by construction.

$$\mathrm{Bacc}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{2} \cdot \left[\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{TN}} + \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}\right] \tag{3.44}$$

(iii) *Precision*, also known as positive predictive value: this is the fraction of TP over the total number of samples classified as positive. Precision is defined for binary classification, but it can be extended to multiclass problems selecting the positive class.

$$\text{Prec}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.45}$$

(iv) *Recall*, also known as sensitivity or True Positive Rate (TPR): this is the fraction of TP over the total number of actually positive samples. Recall is defined for binary classification, but it can be extended to multiclass problems selecting the positive class.

$$\text{Rec}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.46}$$

(v) *F_1-score*, also known as F-measure: this is the harmonic mean of precision and recall and it can be used to control both of them at the same time.

$$\text{F}_1(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}} \tag{3.47}$$

(vi) *Matthews correlation coefficient* (MCC), also known as φ-coefficient: MCC values ranges from -1 to $+1$, where MCC $= 1$ is the optimal classification, MCC $= 0$ is the score achieved by the random classifier, even in cases of unbalanced problems, and MCC $= -1$ is the inverse prediction. MCC is defined for binary classification, but it can be extended to multiclass problems selecting the positive class.

$$\text{MCC}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \tag{3.48}$$

(vii) *Area Under Receiver Operating Curve* (AUC): the Receiver Operating characteristic Curve (ROC) is a plot used to assess the performance of a binary classifier as its classification threshold is varied. In ROC analysis, the horizontal axis is the False Positive Rate (FPR), also known as fall-out, which is defined as

$$\text{FPR}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3.49}$$

while the vertical axis is the recall, defined in Equation (3.46). An example of ROC analysis is reported in Figure 3.22. The ROC of a random classifier is simply the bottom-left to top-right diagonal Plots in the upper triangular area indicate *good* classifiers. A fundamental metric for classification problems is the AUC. AUC ranges from 0 to 1, where optimal classification leads to AUC $= 1$ and the random classifier achieves AUC $= 0.5$.

Clustering Quantitatively measuring the output of a clustering algorithm is not a trivial task, in particular when no ground truth on the clusters is provided. We report here a list of the most common clustering performance metrics.

(i) *Silhouette*: the main characteristic of this metric is that it can be used when no ground truth on the clustering is provided [Rousseeuw, 1987]. The silhouette coefficient ranges from -1 to 1 and it is higher for compact and well-separated clusters. Silhouette score around 0 are associated to partially overlapping clusters.

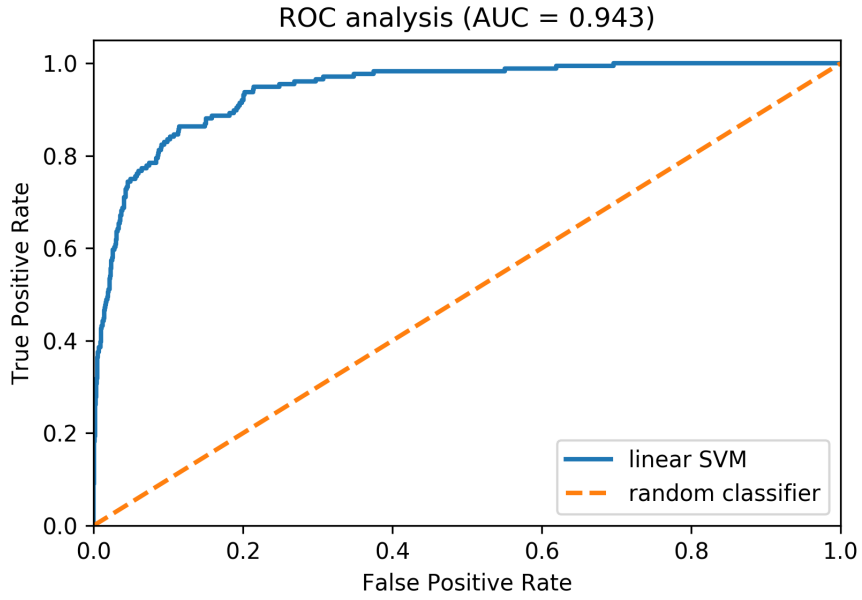


Figure 3.22: An example of ROC analysis on the MNIST dataset [LeCun et al., 2010]. In this example a linear SVM is trained on only 1% of the dataset, where 200 additional noisy features were added to make the classification problem harder.

Let $a(\boldsymbol{x}_i)$ be the average dissimilarity of $\boldsymbol{x}_i$ with all the other points in the same cluster; let also be $b(\boldsymbol{x}_i)$ be the average distance of $\boldsymbol{x}_i$ to all the points in the nearest cluster. We can interpret $a(\boldsymbol{x}_i)$ as a measure of point-to-cluster assignment, where the higher $a(\boldsymbol{x}_i)$ is, the more the point $\boldsymbol{x}_i$ is correctly assigned. On the other hand, $b(\boldsymbol{x}_i)$ can be seen as a clusters separability index. The silhouette coefficient is defined as follows.

$$s(\boldsymbol{x}_i) = \frac{b(\boldsymbol{x}_i) - a(\boldsymbol{x}_i)}{\max[a(\boldsymbol{x_i}), b(\boldsymbol{x}_i)]} \tag{3.50}$$

The quality of the achieved clustering can be graphically assessed visualizing the silhouette coefficient for each sample, as in Figure 3.23[18], or it can be globally measured by its average score $\bar{s} = \frac{1}{n}\sum_{i=1}^{n} s(\boldsymbol{x}_i)$.

(ii) *Homogeneity*: a clustering satisfies the homogeneity property if each cluster contains only members of a single class. In order to evaluate the homogeneity, the clustering ground truth must be provided. In this sense, this can be seen as a supervised measure.

(iii) *Completeness*: a clustering satisfies the completeness property if all members of a given class are assigned to the same cluster. As for the homogeneity, this is a supervised measure.

(iv) *V-measure*: this is simply defined as the harmonic mean of completeness and homogeneity.

(v) *Adjusted Rand Index* (ARI): this index is a chance normalized measure of the similarity between two cluster assignments which is robust with respect to permutations of the cluster identifier. This index ranges from -1 and 1, where -1 stands for independent cluster assignments and 1 is perfect match. As it requires the clusters

[18] Image created with ADENINE, see Chapter 4.

ground truth, this is a supervised metric. Let $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ be a ground truth and an estimated cluster assignment, respectively, ARI is defined as follows.

$$\text{ARI}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{\text{RI} - E[\text{RI}]}{\max(\text{RI}) - E[\text{RI}]} \tag{3.51}$$

where RI (the unadjusted Rand Index) is given by

$$\text{RI} = \frac{g + h}{\psi}$$

where g is the number of pairs of elements that are in the same set both in $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$, h is the number of pairs of elements that are in different sets in both $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ and ψ is the total number of pairs (without ordering).

(vi) *Adjusted Mutual Information* (AMI): similar to ARI, this is an information theory-based metric to measure the similarity between two cluster assignments. Let $Y = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_R\}$ and $\hat{Y} = \{\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_S\}$ be a ground truth and an estimated clustering partitions, respectively, AMI is given by

$$\text{AMI}(\hat{Y}, Y) = \frac{\text{MI} - E[\text{MI}]}{\max[H(\hat{Y}), H(Y)] - E[\text{MI}]} \tag{3.52}$$

where H is the entropy and MI is the Mutual Information, defined as

$$\text{MI}(\hat{Y}, Y) = \sum_{i=1}^{R} \sum_{j=1}^{S} P(i, j) \log \left[\frac{P(i, j)}{P(i) P'(j)} \right]$$

where $P(i)$ is the probability that a randomly picked sample falls in $\boldsymbol{y}_i$, likewise for $\hat{\boldsymbol{y}}$ and $P'(j)$. While, $P(i, j)$ is the probability that a random sample falls in both clusters $\boldsymbol{y}_i, \hat{\boldsymbol{y}}_j$.

3.5 Hardware and software solutions

ML is currently experiencing a renewed wave of interest. Several new technologies, both software and hardware, are specifically designed for ML applications. The number of ML practitioners and researcher is growing fast and the number of problems that are now tackled with ML is rapidly evolving. Therefore, a description of the current ML technological state of the art is completely pointless as it would become outdated in no time.

Nevertheless, describing the philosophy that the most appreciated ML software solutions are currently adopting is more interesting, as it may last longer. The vast majority of ML and data science software libraries presents two stacks: a high level interface, mainly developed for Python, R, Julia or Lua, and a set of highly optimized algorithms, written in C++ or Fortran, that take care of the heavy computational workloads.

In modern ML algorithms, linear algebra operations are the most common and computational demanding tasks to run. In order to carry them out as efficiently as possible ML software developers are currently following two different strategies: (i) exploiting the computational

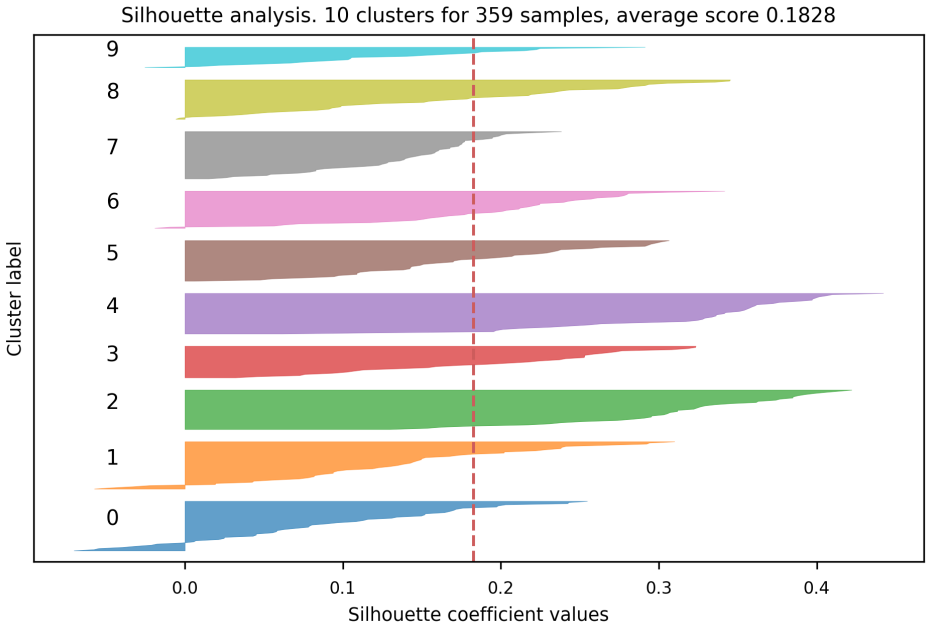


Figure 3.23: An example of Silhouette analysis where a 10 clusters k-means runs on a randomly sampled 80% of the MNIST dataset [LeCun et al., 2010].

power offered by multi-core CPUs by using optimized BLAS libraries (such as OpenBLAS or Intel® MKL), or (ii) reducing the computational time by offloading the heavy linear algebra operations on GPU cards.

The first solution is inherited from standard scientific computing and it is adopted from the very beginning of ML diffusion. Instead, the second is a relatively recent solution and it is currently a hot-topic in ML software development. This is mainly due to the diffusion of deep learning applications that started to exponentially rise from 2012, when AlexNet [Krizhevsky et al., 2012], a deep CNN implemented in CUDA, ranked first in the ImageNet large scale visual recognition challenge achieving a dramatically improved top-5 accuracy with respect to the previous editions. Since then, the interest of ML researcher toward GPU computing is heavily grown.

Another key aspect that characterize ML applications is the high number of independent tasks that need to be executed. As an example, we can think about hyperparameters optimization via grid-search cross-validation (see Section 3.4.2), where a large number of independent models is trained on the same set of data. In large-scale ML research, this computationally heavy workload can be distributed on the computing nodes of HPC architectures. A popular library in ML that exploit this parallel programming paradigm is Dask, a dynamic task scheduling manager developed for large-scale data intensive applications (see Chapter 4).

All the data science challenge described in the second part of this PhD thesis are tackled with Python-based software solutions. In particular, the numerical computations are carried out by NumPy[19] and SciPy[20], while the data management is left to pandas[21]. Moreover, for the vast majority of standard ML models (such as generalized linear models or ensemble learning

[19] Source: http://www.numpy.org (last visit 2018-01).
[20] Source https://www.scipy.org (last visit 2018-01).
[21] Source https://pandas.pydata.org (last visit 2018-01).

strategies) I took advantage of SCIKIT-LEARN [Pedregosa et al., 2011][22], which is one of the most popular ML libraries available online. I also relied on custom solutions, such as L1L2PY[23] and MINIMAL[24], for ML models that are not available via SCIKIT-LEARN API.

[22] Source http://scikit-learn.org (last visit 2018-01).
[23] Soruce: https://github.com/slipguru/l1l2py
[24] Source https://github.com/samuelefiorini/minimal

Part II

"*Machine learning isn't mathematics or physics, where major advances can be done with a pen and a piece of paper. It's an engineering science.*"

François Chollet, Deep Learning with Python [Chollet, 2018]

4 ADENINE: a HPC-oriented tool for biological data exploration

This chapter presents the first original contribution described in this thesis, which is the development of a ML framework designed for biological data exploration and visualization called ADENINE*. The goal of this framework is to help biomedical data scientists achieving a first and quick overview of the main structures underlying their data. This software tool encompasses state of the art techniques for missing values imputing, data preprocessing, dimensionality reduction and clustering.* ADENINE *has a scalable architecture which seamlessly work on a single workstation as well as on a high-performance computing facility.* ADENINE *is capable of generating publication-ready plots along with quantitative descriptions of the results. The end of the chapter presents an exploratory analysis of a biological dataset carried out with* ADENINE*.*

4.1 Exploratory data analysis

In biology, as well as in any other scientific domain, exploring and visualizing the collected measures is an insightful starting point for every analytical process. For instance, the aim of a biomedical study can be detecting groups of patients that respond differently to a given treatment, or inferring possible molecular relationships among all, or a subset, of the measured variables. In both cases, data scientists will be asked to extract meaningful information from collections of complex and possibly high-dimensional measures, such as gene sequencing data, biomedical imaging, patients assessment questionnaires, and so on.

In these cases, a preliminary Exploratory Data Analysis (EDA) is not only a good practice, but also a fundamental step to run before further and deeper investigations can take place. Representative examples are, for instance, Kaggle competitions (source: `www.kaggle.com`). Kaggle is the most popular community-based data science challenge platform online. Browsing through the hosted competitions, we can see that people run EDAs to get a grasp on the datasets before testing their own prediction strategy.

Running EDA, despite being a valuable and widely applied data science practice, is definitely a nontrivial task. Indeed, for a given dataset, it is usually unclear which algorithm will provide the most insightful results. A common practice is to empirically test some of them and compare their results from a qualitative/quantitative standpoint. However, applying EDA methods on large-scale data can be burdensome or even computationally unfeasible.

In the last few years, a fair number of data exploration software and libraries were released. Such tools may be grouped in two families: GUI-based and command-line applications.

Among the first group we recall Divvy [Lewis et al., 2013], a software tool that performs dimensionality reduction and clustering on input data sets. *Divvy* is a light framework; however, its collection of C/C++ algorithm implementations does not cover common strategies such as

kernel principal component analysis or hierarchical clustering (see Section 3.3.2), it does not offer strategies to perform automatic discovery of the number of clusters and it is only available for macOS. One of the most popular GUI-based data analysis software is KNIME[1]. KNIME has a rather user-friendly interface where creating EDA pipelines is as simple as drawing small trees, where each node is a box representing some algorithm. KNIME is also suitable for large datasets as it can distribute the computation on Oracle Grid Engine[2] clusters. The main drawback of such GUI-based softwares is that they do not allow to automatically generate EDA pipelines. Moreover, plugging in hand-crafted algorithm is not easy. One possibility is to use KNIME Meta Nodes, but programming relatively complex algorithms completely relying on this tool can be cumbersome.

Nextflow[3] is a command-line workflow framework that allows to easily distribute the computations of complex pipelines. One of the strength of *Nextflow* pipelines is that they are made by chaining together differently many processes that can even be written in differently many languages. *Nextflow* is not just a simple computational framework, it is also a proper Domain Specific Language (DSL) which extends Groovy[4]. Although very powerful, *Nextflow* is more tailored toward the bioinformatics and computational biology area than a more general biomedical data science audience. Using *Nextflow*, everything must be hard-coded, therefore plugging in custom algorithms is as hard as using standard libraries. Similar considerations can be made for Snakemake[5], which is a pythonic workflow management system which is mainly tailored for the parallelization of the analysis of DNA data.

The most notable project that spans between the two families is *Orange*[6] [Demšar et al., 2013], a data mining software suite that offers both visual programming front-end and Python APIs. In the context of data exploration, *Orange* can be successfully employed. However, in order to test different data analysis pipelines, each one must be manually created as it does not support their automatic generation. Moreover, large data sets are difficult to analyze as it can run only on a single workstation, lacking of distributed computing support.

4.2 ADENINE overview

This chapter introduces ADENINE[7], a command-line Python tool for data exploration and visualization that, combining different EDA methods, creates textual and graphical analytical reports of large scale, data collections.

Missing data imputing, preprocessing, dimensionality reduction and clustering strategies are considered as building blocks for data analysis pipelines. The user is simply required to specify the input data and to select the desired blocks. ADENINE, then, takes care of generating and running the pipelines obtained by all possible combinations of the selected blocks. Table 4.1 shows the list of building blocks available in the current release. Every algorithm implementation in ADENINE is inherited, or extended, from SCIKIT-LEARN [Pedregosa et al., 2011] which, as of today, is definitely the most complete ML open source Python library available. Moreover,

[1] Source: www.knime.com.
[2] Source http://www.univa.com/products/.
[3] source: www.nextflow.io.
[4] Source http://groovy-lang.org/.
[5] Source https://snakemake.bitbucket.io.
[6] If I have to pick one, this is definitely my favorite – Ed.
[7] Source : http://slipguru.github.io/adenine.

virtually any custom algorithm can be easily plugged in an ADENINE pipeline, as long as it is implemented following the SCIKIT-LEARN standard, see Section 4.3.

Step	Algorithm	Reference
Imputing	mean	
	median	
	most frequent	
	k-nearest neighbors	[Troyanskaya et al., 2001]
Preprocessing	recentering	
	standardize	
	normalize	
	min-max	
Dimensionality reduction	principal component analysis (PCA)	[Jolliffe, 2002]
	incremental PCA	[Ross et al., 2008]
	randomized PCA	[Halko et al., 2011]
	kernel PCA	[Schölkopf et al., 1997]
	isomap	[Tenenbaum et al., 2000]
	locally linear embedding	[Roweis and Saul, 2000]
	spectral embedding	[Ng et al., 2002]
	multi-dimensional scaling	[Borg and Groenen, 2005]
	t-distributed stochastic neighbor embedding	[Van der Maaten and Hinton, 2008]
Clustering	k-means	[Bishop, 2006]
	affinity propagation	[Frey and Dueck, 2007]
	mean shift	[Comaniciu and Meer, 2002]
	spectral	[Shi and Malik, 2000]
	hierarchical	[Hastie et al., 2009]
	DBSCAN	[Ester et al., 1996]

Table 4.1: Pipeline building blocks currently available in ADENINE.

ADENINE is developed with the aim of speeding-up EDAs on large biomedical data collections. It has a scalable architecture, that allows its pipelines to seamlessly run in parallel as separate Python processes on a single workstation or as separate tasks in a High-Performance Computing (HPC) cluster facility. This remarkable feature allows to explore and visualize massive amounts of data in a reasonable computational time. Moreover, as ADENINE makes large use of NUMPY and SCIPY, it automatically benefits from their bindings with optimized linear algebra libraries (such as OpenBLAS or Intel® MKL). Thanks to this hybrid parallel architecture, with ADENINE it is possible to get a grasp on the main structures underlying some data in a reduced computational time.

In ADENINE, any tabular data format (`.csv`, `.tsv`, `.json`, and so on) can be given as input. Moreover, as genomic is one of the major sources of data in the biological world, ADENINE also natively supports data integration with the NCBI Gene Expression Omnibus (GEO) archive [Barrett et al., 2013]; which datasets can be simply retrieved by specifying their GEO accession number.

4.3 Software description

ADENINE is developed around the data analysis concept of *pipeline*. An ADENINE pipeline is a

sequence of the following fundamental steps:

1. missing values imputing;
2. data preprocessing;
3. dimensionality reduction and
4. unsupervised clustering.

For each task, is it possible to use either the off-the-shelf algorithms in Table 4.1, either any other custom algorithm as long as it is implemented in Python as a SCIKIT-LEARN `Transformer` object. A `Transformer` is simply a Python class having a `fit_transform` method, see for instance snippet below.

```
class DummyTransformer(object):
    """A dummy transformer class."""

    def fit(self, X, y):
         """Fit DummyTransformer."""
         return self

    def transform(self, X):
        """Transform X."""
        return X

     def fit_transform(self, X, y=None):
         """Fit to data, then transform it."""
         return self.fit(X, y).transform(X)
```

SCIKIT-LEARN also provides an utility object called `FunctionTransformer` that returns a `Transformer` given any arbitrary input callable (*i.e.* function), which can be very resourceful for our purposes.

Data collected in biomedical research studies often present missing values and devising imputing strategies is a common practice to deal with such issue [De Souto et al., 2015]. ADENINE offers an improved version of the `Imputer` class provided by SCIKIT-LEARN. In addition to the pre-existent feature-wise `"mean"`, `"median"` and `"most_frequent"` strategies, this extension adds `"nearest_neighbors"`, *i.e.* an implementation of the k-nearest neighbors imputing method proposed for microarray data in [Troyanskaya et al., 2001].

Collecting data from heterogeneous sources may imply dealing with variables lying in very different numerical ranges and this could have a negative influence on the behavior of further data analysis techniques. To tackle this issue ADENINE offers different strategies to preprocess data, such as re-centering, standardizing or rescaling. We recall here that any other preprocessing step, such as feature exclusion, log-transformation and so on, can be implemented as `Transformer` to be easily chained in an ADENINE pipeline.

ADENINE includes a set of linear and nonlinear dimensionality reduction and manifold learning algorithms that are particularly suited for projection and visualization of high-dimensional

data. Such techniques rely on the fact that it is often possible to *decrease* the dimensionality of the problem estimating a low-dimensional embedding in which the data lie, as described in Section 3.3.2.

Besides offering a wide range of clustering techniques, ADENINE implements strategies and heuristics to automatically estimate parameters that yield the most suitable cluster separation. The selection of the optimal number of clusters in centroid-based algorithms follows the B-fold cross-validation strategy[8] presented in Algorithm 1, where $\mathcal{S}(X, \hat{y})$ is the mean silhouette coefficient of the input samples, see Equation (3.50).

Algorithm 1 Automatic discovery of the optimal clustering parameter.

for clustering parameter k in $k_1 \dots k_K$ **do**
  for cross-validation split b in $1 \dots B$ **do**
    $X_b^{tr}, X_b^{vld} \leftarrow b$-th training, validation set
    $\hat{m} \leftarrow$ fit model on X_b^{tr}
    $\hat{y} \leftarrow$ predict labels of X_b^{vld} according to $\hat{m}$
    $s_b \leftarrow$ evaluate silhouette score $\mathcal{S}(X_b^{vld}, \hat{y})$
  end for
  $\bar{S}_k = \frac{1}{B} \sum_{i=1}^{B} s_i$
end for
$k_{opt} = \text{argmax}\, k(\bar{S}_k)$

For affinity propagation [Frey and Dueck, 2007] and k-means (see Section 3.3.1.1) clustering parameters can be automatically defined (`"preference"` and `"n_clusters"`, respectively). Mean shift [Comaniciu and Meer, 2002] and DBSCAN [Ester et al., 1996] offer an implicit cluster discovery. For hierarchical and spectral clustering (see Sections 3.3.1.3 and 3.3.1.2), no automatic discovery of clustering parameters is offered. However, graphical aids are generated to evaluate clustering performance such as dendrogram tree and a plot of the eigenvalues of the Laplacian of the affinity matrix[9], respectively.

In order to speed-up EDA on large-scale datasets, ADENINE exploits two main parallel computing paradigms: multiprocessing and multithreading. The latter is provided by the use of optimized linear algebra libraries of NUMPY and SCIPY and it comes *for free*. So, let us focus on multiprocessing. This simply consists in dividing the workload among different tasks which can possibly run on different nodes of an HPC infrastructure. The input of each ADENINE pipeline is the raw dataset to which successive transformations and algorithms are applied. Therefore, each pipeline is independent and this makes ADENINE suitable for the application of simple distributed computing schemes. ADENINE offers different job distribution strategies.

1. When ADENINE is running on a single workstation (or on a single node of an HPC architecture), each pipeline is parallelized by using the standard multiprocessing Python library. Each pipeline run is mapped to a job. All the jobs are organized in a queue and the multiprocessing library automatically assigns the cores available to the next job. In this way, ADENINE is capable of exploiting all the cores of the machine in which it is running.

2. When an HPC architecture is available, ADENINE is capable of distributing its pipelines across multiple nodes. In particular, the following two options are available.

[8] We call it B-fold instead of K-fold, as in Section 3.4.1, to avoid confusion with the number of clusters, which is indicated with the letter "K" as well.

[9] See the *eigengap heuristics* [Von Luxburg, 2007].

MPI ADENINE implements an MPI-based[10] master/slave job distribution strategy. A single master task T_0 creates a queue of jobs $Q = \{J_1, \ldots, J_M\}$, where M is the number of pipelines. The jobs in Q are assigned to a pool of slaves $S = \{T_1, \ldots, T_S\}$. Once the slave T_j (with $j > 0$) has completed its task, it communicates the results to T_0 and becomes idle. At this point, T_0 assigns to T_j the next task in Q. This procedure is repeated several times, until Q is empty. The number of slaves S can be controlled by the user with the parameter `np` of the `mpirun` command. S must be carefully chosen according to the size of the dataset, the number of pipelines generated and the available hardware. Thanks to MPI, the assignment of each task to its node is completely transparent to the user. When all the computation is done, the results are collected in a single result folder.

Dask ADENINE can also exploit the Dask parallel computing library[11]. While MPI is a recognized standard in the HPC world, Dask is relatively recent project developed for dynamic task scheduling focused on big data applications. In particular, ADENINE exploits the Dask-based backend of Joblib[12], which is the parallel computing library mainly used in SCIKIT-LEARN to parallelize independent tasks. Dask can be used by simply running a `dask-scheduler` executable process on one node and the `dask-worker` executable on several processes, which can run on other nodes of the HPC architecture. After an initial hand-shake between the scheduler and its workers, a list of jobs to run can be sent to the scheduler which automatically distributes the workload among its workers and collects the results. Using Dask in ML applications, where communications between tasks is usually negligible, can be easier with respect to MPI.

4.4 Usage example

In this section we show how ADENINE can be used to perform two EDAs on a gene expression microarray data set obtained from the GEO repository (accession number GSE87211). This data set was collected in a recent medical study that aimed at understanding the underlying mechanism of colorectal cancer (CRC) as well as identifying molecular biomarkers, fundamental for the disease prognostication. It is composed of 203 colorectal cancer samples and 160 matched mucosa controls. The adopted platform was the Agilent-026652 Whole Human Genome Microarray, which was used to measure the expression of 34127 probe sets.

ADENINE offers a handy tool to automatically download the data set from the GEO repository given only its accession name. It also let the user select phenotypes and/or probe sets of interest. Given these preferences, ADENINE automatically converts the data set from the *SOFT* format to a comma-separated values text file. To download the remote GEO data set specifying the tissue type as phenotype of interest we used the following command.

```
$ ade_GEO2csv.py GSE87211 --label_field characteristics_ch1.3.tissue
```

This automatically creates `GSE87211_data.csv` and `GSE87211_labels.csv` which contain gene expression levels and tissue type of each sample, respectively.

[10] Source: http://mpi-forum.org/.
[11] Source: https://dask.pydata.org.
[12] Source: https://pythonhosted.org/joblib/.

The first EDA aims at stratifying the samples according to their tissue type (mucosa or rectal tumor) this can be performed by executing the following command.

```
$ ade_run.py config.py
```

Where `config.py` is a configuration file which should look like the snippet below.

```
from adenine.utils import data_source
data_file = 'GSE87211_data.csv'
labels_file = 'GSE87211_labels.csv'
X, y, feat_names, index = data_source.load(
'custom', data_file, labels_file, samples_on='rows', sep=',')
step1 = {'Recenter': [True]}
step2 = {'KernelPCA': [True, {'kernel': ['linear', 'rbf', 'poly']}],
         'Isomap': [True, {'n_neighbors': 5}]}
 step3 = {'KMeans': [True, {'n_clusters': ['auto', 2]}]}
```

Each `step` variable refers to a dictionary having the name of the building block as key and a list as value. Each list has a *on**off* trigger in first position followed by a dictionary of keyword arguments for the class implementing the corresponding method. When more than one method is specified in a single step, or more than a single parameter is passed as list, ADENINE generates the pipelines composed of all possible combinations.

The configuration snippet above generates eight pipelines with similar structure. Each one has samples centered in zero, then projected on a 2D space by isomap or by linear, Gaussian or polynomial kernel PCA. k-means clustering with $K = 2$ and with automatic cluster discovery is eventually performed on each dimensionality-reduced data set, as in Algorithm 1. Results of such pipelines are all stored in a single output folder. Once this process is completed, plots and reports can be automatically generated running the following command.

```
$ ade_analysis.py results/ade_output_folder_YYYY-MM-DD_hh:mm:ss
```

The aim of the second EDA is to uncover the relationships among a set of genes known from the literature to be strongly associated with CRC. Specifically this signature is composed of the following genes: APC, KRAS, CTNNB1, TP53, MSH2, MLH1, PMS2, PTEN, SMAD4, STK11, GSK3B and AXIN2 [Schulz, 2005]. We also considered probe sets measuring expression level of the same gene, and we labelled them with a progressive number. Three partially overlapping sublists compose this signature.

S1) Genes fundamental for the progression of CRC (*i.e.* APC, KRAS, CTNNB1, TP53).

S2) Genes relevant in the *Wnt signaling pathway*, which is strongly activated in the first phases of CRC (*i.e.* APC, CTNNB1, GSK3B, AXIN2).

S3) Genes involved in hereditary syndromes which predispose to CRC (*i.e.* APC, MSH2, MLH1, PMS2, PTEN, SMAD4, STK11) [Schulz, 2005].

A reduced version of the GEO data set that comprises only such genes can be easily created calling `ade_GEO2csv.py` with the option `--signature GENE_1,GENE_2,...,GENE_N`. Moreover, the option `--phenotypes P_1,P_2,...,P_M` can be used to keep only mucosa or rectal tumor samples. To run such experiment, one simply needs to select and activate the hierarchical clustering building block and to follow the same steps presented above.

4.5 Results

In the first EDA, we compared the clustering performance achieved by the eight ADENINE pipelines and we reported in Figure 4.1 an intuitive visualization of the results achieved by the top three, evaluated in terms of silhouette score [Rousseeuw, 1987]. As expected, the top performing pipelines show a clear separation between the two sample groups, as the k-means algorithm devises a domain partitioning that is consistent with the actual tissue types.

For the second EDA, the relationships among the probe sets corresponding to the genes of the signature are separately explored learning a different hierarchical clustering tree for mucosa (Figures 4.2a) and CRC samples (Figure 4.2b), separately. The two trees are learned from different tissues, nevertheless they show some remarkable similarities. For instance, the pairs TP53-TP53.1 and MSH2-PMS2.1 always share a common parent. Interestingly, the first is a relationship between probe sets of the same gene, and the second is confirmed in literature, as MSH2 and PMS2 are both involved in hereditary non-polyposis CRC, a syndrome that predisposes for CRC. Moreover, two probe sets of the the genes of *S1*, namely APC and CTNNB1, are consistently close to the root of the two trees. This suggest that the expression level of these two genes highly differs from the others. Two interesting differences between the two trees can also be noticed. First, most of the elements of the sublist *S3*, which contains genes that enhance the risk of developing CRC, tend to be grouped together in Figure 4.2b, while the same observation cannot be done for Figure 4.2a. Secondly, probe sets of the genes belonging to sublists *S2* and *S3* tend more to be closely connected in Figure 4.2b than in Figure 4.2a.

4.6 Conclusions

In this Chapter we presented ADENINE, a biomedical data exploration and visualization tool that can seamlessly run on single workstations as well as on HPC clusters. Thanks to its scalable architecture, ADENINE is suitable for the analysis of large and high-dimensional data collections, that are nowadays ubiquitous in biomedicine. This software natively supports the integration with the GEO repository. Therefore, a user provided with the accession number of the data set of interest can select target phenotypes and genotypes and ADENINE takes care of automatically downloading the data and plugging them into the computational framework. ADENINE offers a wide range of missing values imputing, data preprocessing, dimensionality reduction and clustering techniques that can be easily selected and applied to any input data. We showed ADENINE capabilities performing two EDAs on a CRC gene expression data set. From the obtained results we can observe that a clear discrimination between CRC and control samples can be achieved by unsupervised data analysis pipeline. Moreover, a meaningful description of the relationships among the group of genes strongly associated with CRC can be represented as hierarchical trees.

ADENINE is developed for biomedical data scientists, with it can be used for any kind of tabular data. Hence meeting the needs of a more general purpose class of scientists that aim at understanding complex high-dimensional data.

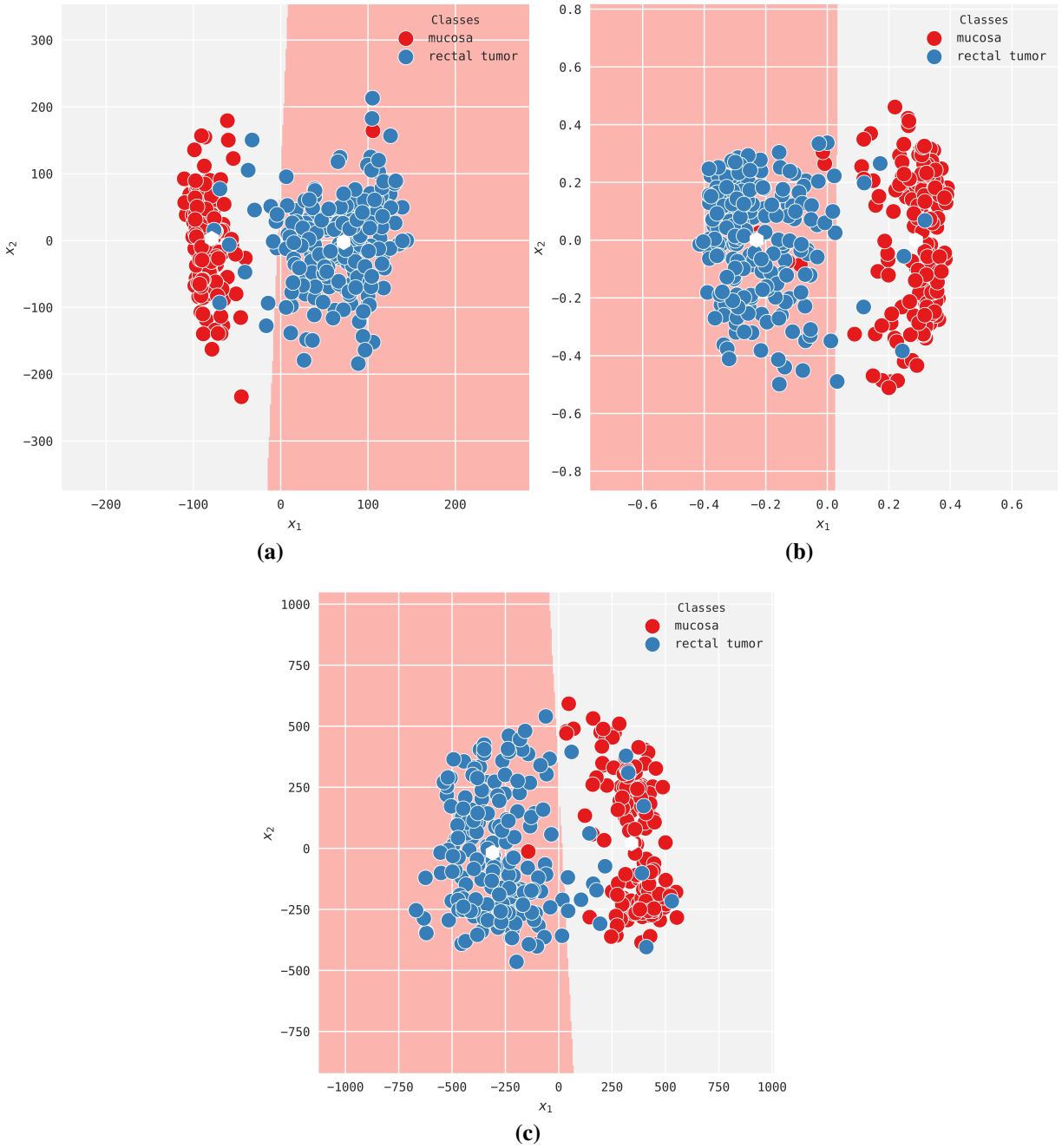
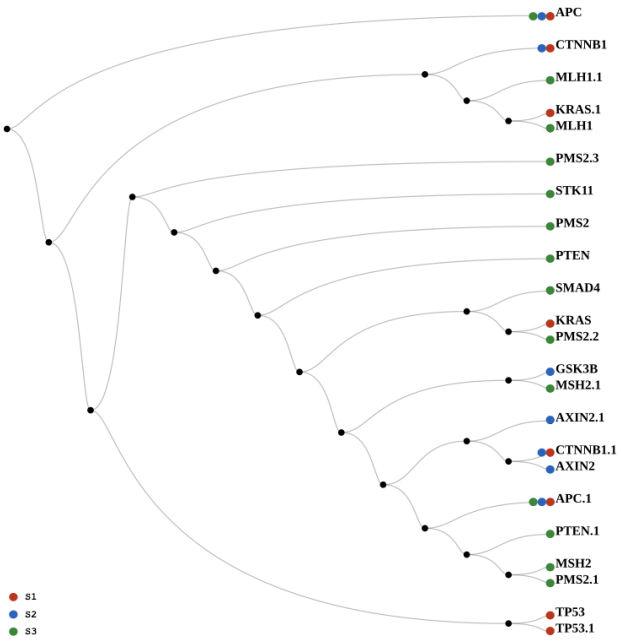
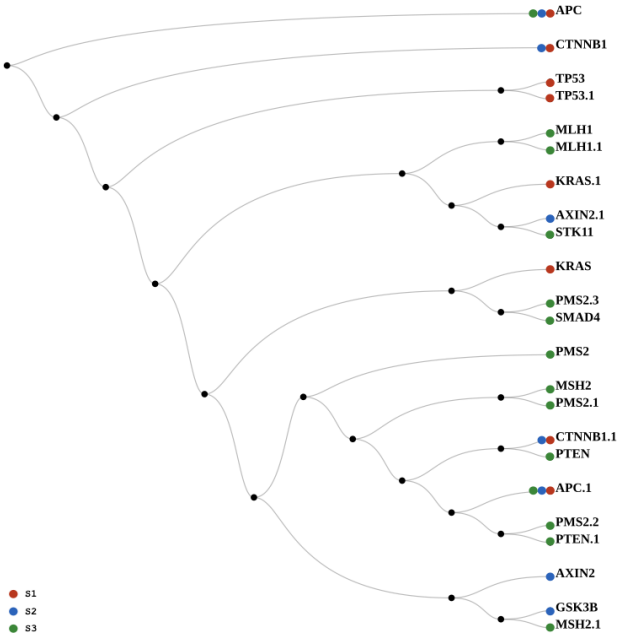


Figure 4.1: Three different 2D projections of the samples of the GEO gene expression data set used in this work. Projections on the left (a), middle (b) and right (c) panes are obtained via linear PCA, Gaussian PCA and isomap, respectively. The color of each point corresponds to the actual tissue type, while the background color is automatically learned by the k-means clustering algorithm. White hexagons correspond to cluster centroids.



(a)



(b)

Figure 4.2: An example of hierarchical trees visualization learned by two ADENINE pipelines on mucosa (a) and CRC (b) samples. Each probe set is color coded according to the corresponding sublist. This visualization provides insights on the underlying structure of the measured gene expression level.

5 Machine learning-based molecular aging clock

In this chapter, we study the changes of energy metabolism during the physiological aging. To this aim we measure a set of molecular biomarkers from peripheral blood mononuclear cells obtained from healthy volunteers with age between 8 *and* 106 *years. With such biomarkers it is possible to quantify oxidative phosphorylation efficiency,* ATP/AMP *ratio, lactate dehydrogenase activity and level of malondialdehyde. After a thorough preliminary data exploration, we develop a regression model that, starting from such measures, is capable of predicting the age of an individual.*

5.1 Introduction: aging and metabolism

In this chapter we present the first biomedical data science challenge of the thesis. This consists in devising a ML model capable of predicting the age of an individual starting from a set of molecular biomarkers collected from 118 volunteers[1].

This chapter describes the extensive EDA and the thorough model selection procedures which lead to the development of the final predictive model. Before diving into the details of the experimental setup, let us see some preliminary biological notions of how aging influences our metabolism.

Aging is a multifactorial process characterized by a progressive decline of physiological functions [Campisi, 2013] which leads to an increment of vulnerability and the relative risk of disease and death [Bratic and Trifunovic, 2010].

Aging represents the primary risk factor for several chronic pathologies, such as cancer, cardiovascular disorders, diabetes and neurodegeneration [López-Otín et al., 2013]. Different molecular pathways seem involved in the aging process, including deregulated autophagy, mitochondrial dysfunction, telomere shortening, oxidative stress, systemic inflammation and metabolism dysfunction [López-Otín et al., 2013; Riera et al., 2016].

For several years, aging has been considered the result of damages accumulation due to an excessive production of reactive oxygen species. A recent paper, [Thompson et al., 2017] proposed an involvement of epigenetic modifications. This led to the development of an *aging clock* based on the degree of DNA methylation, which increases with age [Horvath, 2013].

The *Mitochondrial Theory of Aging* [Harman, 1972; Sastre et al., 2000] derives from the concept that mitochondria are the main source of oxidative stress [Cadenas and Davies, 2000; Turrens, 2003; Dai et al., 2014] and the fact that mitochondrial DNA displays a great rate of mutation together with a less efficient repair machinery with respect to nuclear DNA [Short et al., 2005]. After some mitochondrial DNA mutation threshold, irreversible oxidative damages propagate throughout the genome. This phenomenon leads to dysfunction of mitochondrial metabolism [Genova et al., 2004] accelerating the oxidative stress production [Wallace, 2010].

[1] This was referred to as *the aging problem* throughout Chapter 3.

As shown in [McKerrell et al., 2015], mononuclear cells isolated from peripheral blood, are an excellent model to evaluate the metabolic status of an entire organism. In fact, the molecular alterations identified in peripheral blood cells of aged normal subjects are known to be statistically correlated with degenerative diseases [Jaiswal et al., 2014].

5.2 Data collection

The study presented in this chapter is performed on mononuclear cells isolated from peripheral blood obtained from a population of 118 volunteers[2] with age between 8 and 106 years. In order to preserve the collected blood samples, the vacutainer tubes were transferred into the laboratory and analyzed within 24 hours from collection. All chemicals were purchased from Sigma Aldrich (St. Louis, MO, USA) and Ultrapure water (Milli-Q; Millipore, Billerica, MA, USA) was used throughout. All other reagents were of analytical grade. Data collection and further analysis were managed by a team of specialized biologists at the IRCCS Istituto G. Gaslini, Genoa, IT. The following quantities were measured on each blood sample.

ATP This complex molecule is the main responsible for storing and exchanging energy in cells and it is often referred to as the *energy currency* of the cell. From a chemical point of view, ATP is made of an adenine base attached to a ribose sugar, which, in turn, is attached to three phosphate groups. ATP is heavily involved in the cellular aerobic respiration pathway. High levels of ATP correspond to high energetic state. ATP intracellular concentration, measured in mM/ml, is an important molecular biomarker to evaluate the energetic state of a cell.

AMP This molecule is one of the main derivatives of ATP. In fact, AMP ca be obtained when two phosphate groups are removed from ATP, releasing energy that can be transferred to other molecules to trigger further cell reactions. So, when a cell is in good health, *i.e.* high energetic level, AMP is low and ATP is high. AMP intracellular concentration is measured in mM/ml and it is considered as an important molecular biomarker for the cellular energetic state. ATP and AMP quantification was based on the enzyme coupling method presented in [Ravera et al., 2013].

ATP/AMP ratio Measuring ATP and AMP cellular concentration may not be enough to predict the age of an individual by assessing the energetic state of the peripheral blood cells. A more representative and interesting quantity can be their ratio, so ATP/AMP ratio was calculated and added to the feature set.

Oxygen consumption Aerobic cellular respiration requires oxygen to produce ATP. Therefore, the cellular oximetric level is an important molecular biomarker for the metabolic assessment. Oxygen consumption was measured with an amperometric electrode in a closed chamber, magnetically stirred, at $37°\mathrm{C}$. The oxygen consumption measure, expressed in nmol $O_2/(\mathrm{min} \cdot \mathrm{mg})$, is repeated in two versions, adding two different substrates, *i.e.*: (i) a combination of 5 mM pyruvate with 2.5 mM malate or (ii) 20 mM succinate. In the first oximetric measure, which from now on will be referred to as CO-PYR/MAL, the substrate stimulates the pathway composed by Complexes I, III and

[2] All participants provided their written informed consent to participate in this study, which was approved by the Ethics Committee of the IRCCS Istituto G. Gaslini, Genoa, IT.

Table 5.1: The 12-dimensional feature set.

Measure	Feature name
Gender of the individual	GENDER
ATP intracellular concentration	ATP
AMP intracellular concentration	AMP
ATP/AMP ratio	ATP/AMP
Oxygen consumption under pyruvate + malate	CO-PYR/MAL
Oxygen consumption under succinate	CO-SUCCINATE
ATP synthesis under pyruvate + malate	ATP-PYR/MAL
ATP synthesis under succinate	ATP-SUCCINATE
P/O ratio under pyruvate + malate	PO-PYR/MAL
P/O ratio under succinate	PO-SUCCINATE
Glycolytic flux	LDH
Lipid peroxidation	MDA

IV. On the other hand, in the second oximetric measure, which we call CO-SUCCINATE, the substrate activates the pathway composed by Complexes II, III and IV, as described in [Cappelli et al., 2017].

ATP synthesis We already covered the importance of ATP to evaluate the metabolic state of cells. So, we also measured ATP synthesis, expressed in nmol ATP$/(\min \cdot \text{mg})$, by the highly sensitive luciferin/luciferase method. The same two substrates used for the oximetric evaluation were adopted. In the remainder of the chapter we refer to such measures as ATP-PYR/MAL and ATP-SUCCINATE, accordingly.

P/O ratio In order to assess the efficiency of oxidative phosphorylation, we also evaluated the ratio between ATP synthesis and oxygen consumption under both the substrates. We call the two obtained features as PO-PYR/MAL and PO-SUCCINATE, respectively.

Glycolytic flux In order to assay the glycolytic flux, we measured the activity of the Lactate Dehydrogenase (LDH), expressed in U$/$mg. This enzyme is important to evaluate the other metabolic pathway not mentioned so far, *i.e.* the anaerobic respiration.

Lipid peroxidation The uncoupled oxidative phosphorylation metabolism is often associated with an increment in the oxidative stress production [Dai et al., 2014], which induces damages on proteins, nucleic acid and membrane. Therefore, we evaluated the level of malondialdehyde (MDA) as a marker of lipid peroxidation. The measure of MDA, expressed in μM$/$mg, follows the protocol in [Ravera et al., 2015].

The categorical feature GENDER is encoded as $[0, 1]$ for *male* ad *female*, respectively. Each example of this dataset is then described by the 12-dimensional feature set summarized in Table 5.1.

5.3 Exploratory data analysis

In this section we investigate the relationship between the collected molecular biomarkers and the age of 118 healthy individuals which volunteered to participate to this study. Unfortunately,

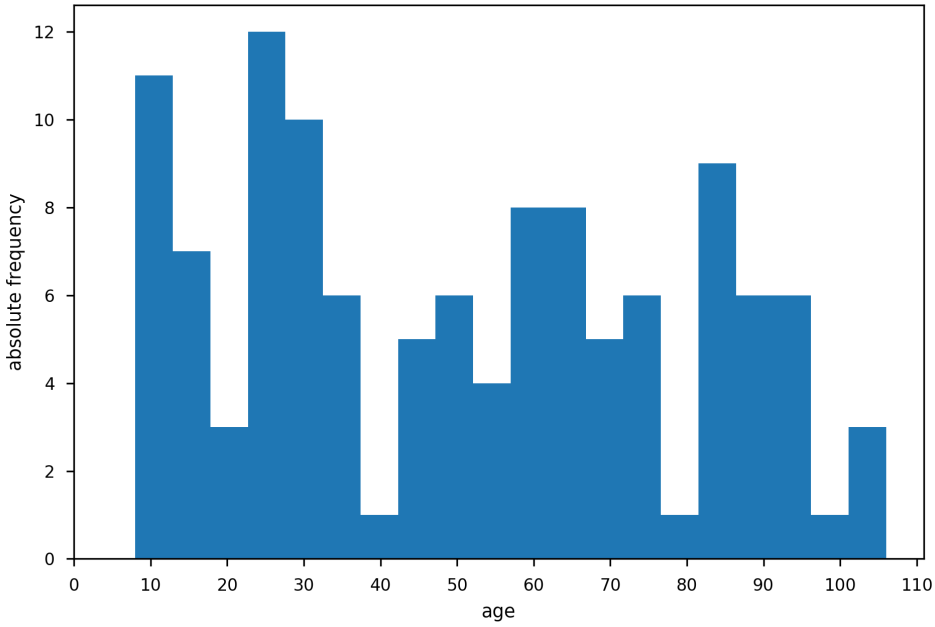


Figure 5.1: Age distribution of the 118 individuals involved in the study.

7 subjects presented missing values (in the features LDH and GENDER). These subjects were excluded from EDA.

The data collection process entirely ran on voluntary basis. So, it is interesting to observe the resulting age distribution. As we can see from the histogram in Figure 5.1, the decades of age are not equally represented. Ideally, we would have collected samples uniformly distributed with respect to their age, but this was unfortunately not possible. Therefore, in this data science challenge we shall adopt robust resampling schemes in order to ameliorate possible biases induced by this phenomenon.

Next, we aim at investigating how the distribution of the molecular biomarkers is influenced by the age of individuals. To this aim we perform a preliminary univariate analysis. We group the measures per decade and we represent their distribution with boxplots, see Figure 5.1. As we can see, most of the biomarkers are clearly influenced by the age. Let us start the visual inspection from the variables related to the mitochondrial aerobic metabolism, *i.e.* Figure 5.2a to 5.1i.

In particular, focusing our attention on the ATP/AMP ratio (Figure 5.1i), which is known to be an energy status monitor of the cells, we can see that the values decrease progressively with the decades, with a drastic drop between 40 and 50 years. Moreover, from an observation of ATP and AMP intracellular concentration (Figure 5.1g and Figure 5.1h, respectively) we can sense how the decrease of the ATP/AMP ratio is mainly due to the growth of AMP in the aging process. Similar considerations can be made for the efficiency of oxidative phosphorylation, evaluated by PO-PYR/MAL and PO-SUCCINATE. In particular, PO-PYR/MAL oscillates around its reference level of 2.5 nmol $O_2/(\text{min} \cdot \text{mg})$ [Hinkle, 2005] in subjects having from 0 to 30 years starting to decrease afterwards. Moreover, PO-PYR/MAL, similarly to ATP/AMP, is fairly stable at its lowest level for elderly (age $\geq$ 70).

Let us now focus on the activity of LDH (Figure 5.1k). As we can see, this metabolic biomarker almost monotonically increases with the aging process. This is mainly due to the fact that

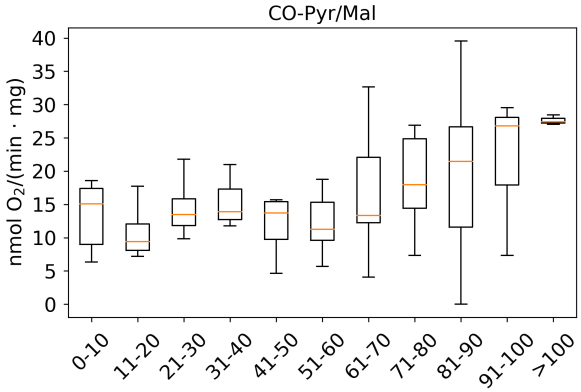
LDH is involved in the glycolysis metabolism, which is a metabolic pathway chosen by the cell to compensate its altered aerobic metabolism. Finally, let us focus on the distributions of MDA (Figure 5.1j). As expected, MDA has an opposite trend with respect to ATP/AMP and PO-PYR/MAL. In fact, it increases from 21 to 80 years. This can be due to the increased oxidative stress production induced by uncoupled oxidative phosphorylation [Dai et al., 2014]. MDA is more stable for elderly, mainly because of their physiological metabolic slowdown.

So far, we have investigated the relationship between the collected measures and the age of the individuals. Let us now focus on the relationship between the variables themselves. In order to investigate possible collinearities in the data, we can evaluate for each pair of variables the Pearson correlation coefficient

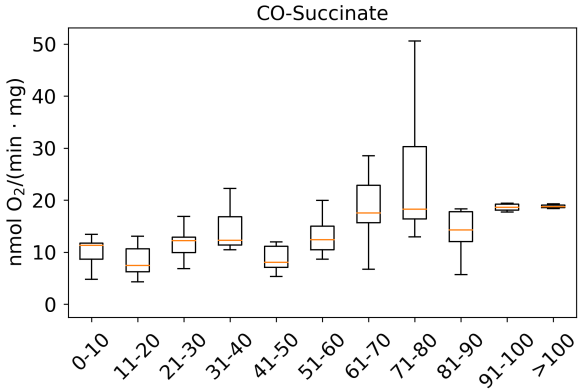
$$\rho(a,b) = \frac{\sum_{i=1}^{n}(a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=1}^{n}(a_i - \bar{a})^2 \sum_{i=1}^{n}(b_i - \bar{b})^2}}$$

where $\bar{c} = \frac{1}{n}\sum_{i=1} c_i$ is the empirical mean any variable c. To ease the visualization, we restrict this analysis to a subset of 5 molecular biomarkers, namely: PO-PYR/MAL, PO-SUCCINATE, ATP/AMP, LDH, MDA. This results in a symmetric 5×5 correlation matrix. We split the data in 5 groups, one for each two-decades, and we represent the collinearity in each group with a symmetric heatmap in which dark red cells are associated with strong positive correlation, white cells represent no correlation and dark blue cells correspond to strong negative correlation, see Figure 5.2. Thanks to this visualization, we can see that ATP/AMP and PO-PYR/MAL have positive correlation until the 6^{th} decade, while showing a negative correlation for elderly. This suggests that, up to approximately 60 years, most of the cellular energy is produced by the mitochondria, while in older subjects this contribution decreases. The same observation can be made for ATP/AMP and PO-SUCCINATE, although in this case it is less evident. On the other hand, the correlation between ATP/AMP and MDA or LDH activity is negative in young subjects, while flipping it sign after approximately 60 years. This is in line with the cellular need to increase the anaerobic metabolism that compensates the inefficiency of aerobic metabolism and the increment of oxidative stress which usually occurs for elderly.

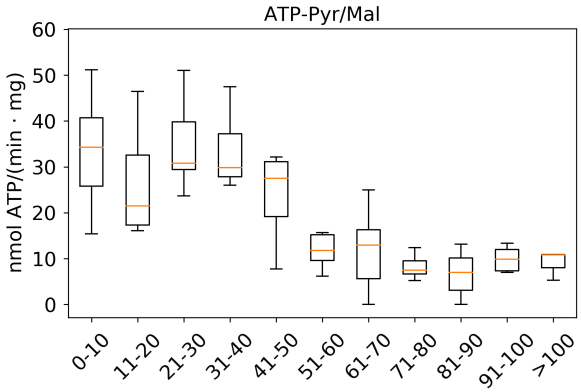
Let us now try to visualize the collected data in a scatter plot. In order to do that, we shall first reduce the dimensionality of the problem, as described in Section 3.3.2. In this EDA, most of the variables showed strong inner linear correlation and correlation with the age. We then reduce the dimensionality of the problem following a two step pipeline: (i) data standardization followed by (ii) linear PCA. Our hope is to recognize some quasi-linear temporal structure. The resulting scatter plot is presented in Figure 5.3. Each point in the scatter plot represents a subject that is color coded according to the age. Let us read the image from left to right. As we can see, it looks like the subjects are partially grouped according to their age. In the top left corner of Figure 5.3 subjects with approximately 20 years (or less) are clustered together. At the center bottom of the plot we can recognize individuals around their thirties. Then, following an approximately linear law from center bottom to top right we can see that the age increases until reaching the elderly, color coded in dark red. The insightful data visualization in Figure 5.3, lets us sense that it is possible to devise a supervised strategy to predict the age of a given individual from the collected molecular biomarkers.


CO-Pyr/Mal
nmol O2/(min · mg)
0-10
11-20
21-30
31-40
41-50
51-60
61-70
71-80
81-90
91-100
>100

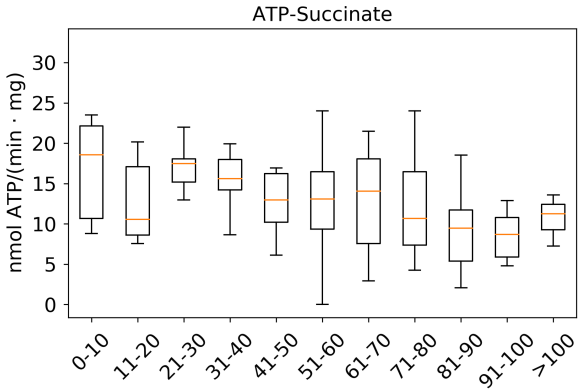
(a)


CO-Succinate
nmol O2/(min · mg)
0-10
11-20
21-30
31-40
41-50
51-60
61-70
71-80
81-90
91-100
>100

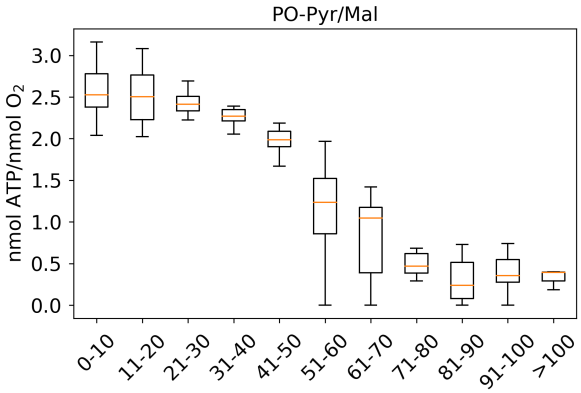
(b)


ATP-Pyr/Mal
nmol ATP/(min · mg)
0-10
11-20
21-30
31-40
41-50
51-60
61-70
71-80
81-90
91-100
>100

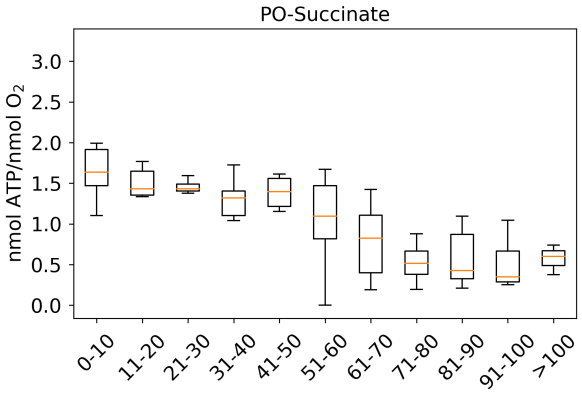
(c)


ATP-Succinate
nmol ATP/(min · mg)
0-10
11-20
21-30
31-40
41-50
51-60
61-70
71-80
81-90
91-100
>100

(d)


PO-Pyr/Mal
nmol ATP/nmol O2
0-10
11-20
21-30
31-40
41-50
51-60
61-70
71-80
81-90
91-100
>100

(e)


PO-Succinate
nmol ATP/nmol O2
0-10
11-20
21-30
31-40
41-50
51-60
61-70
71-80
81-90
91-100
>100

(f)

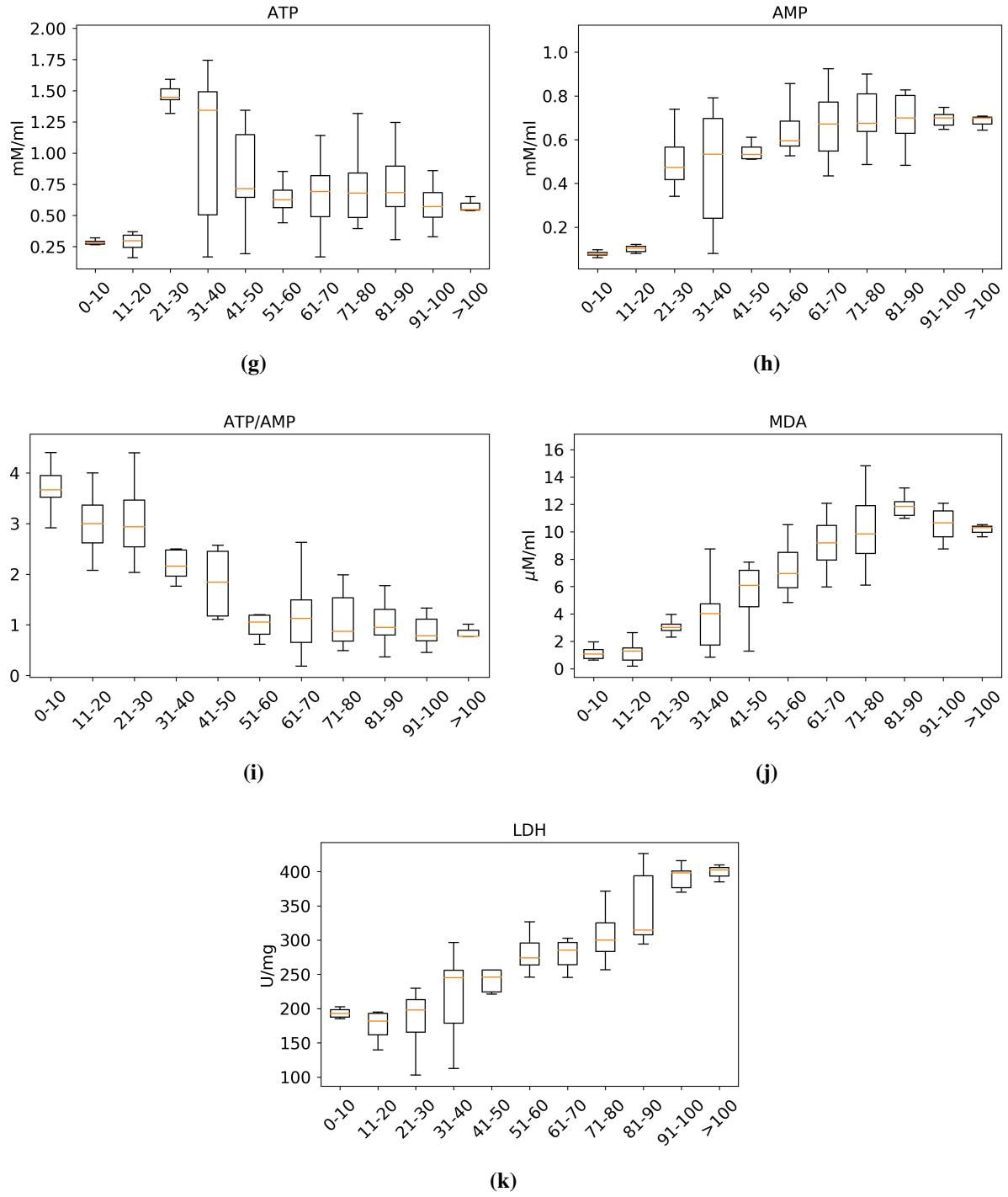


Figure 5.1: Distribution of the collected molecular biomarker values grouped per decade: CO-PYR/MAL panel (a), CO-SUCCINATE panel (b), ATP-PYR/MAL panel (c), ATP-SUCCINATE panel (d), PO-PYR/MAL panel (e), PO-SUCCINATE panel (f), ATP panel (g), AMP panel (h), ATP/AMP panel (i), MDA panel (j) and LDH panel (k).

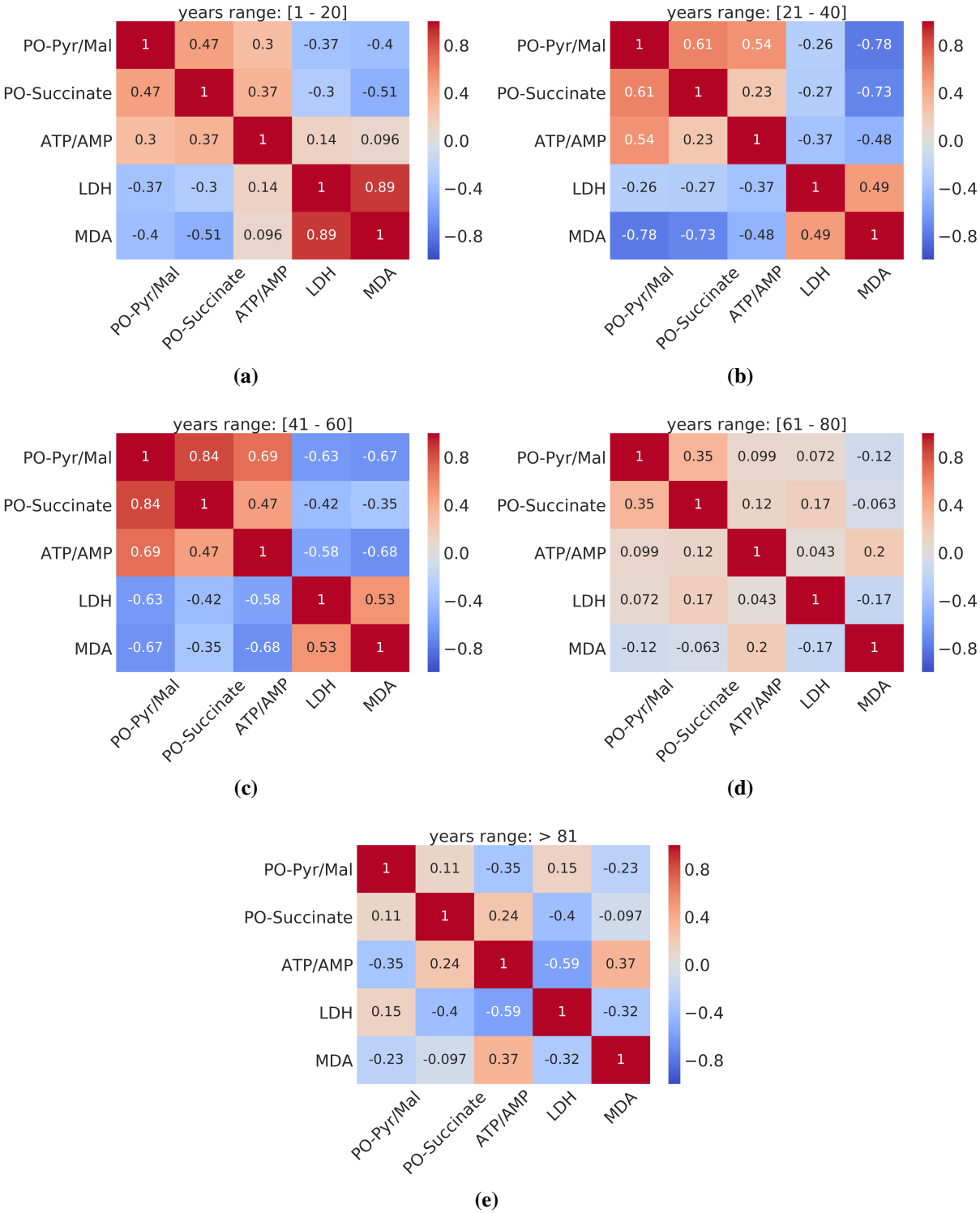


Figure 5.2: The 5×5 symmetric heatmaps representing the Pearson correlation coefficient of the collected variables in the 5 age groups: $[1-20]$ (a), $[21-40]$ (b), $[41-60]$ (c), $[61-80]$ (d), > 81 (e).

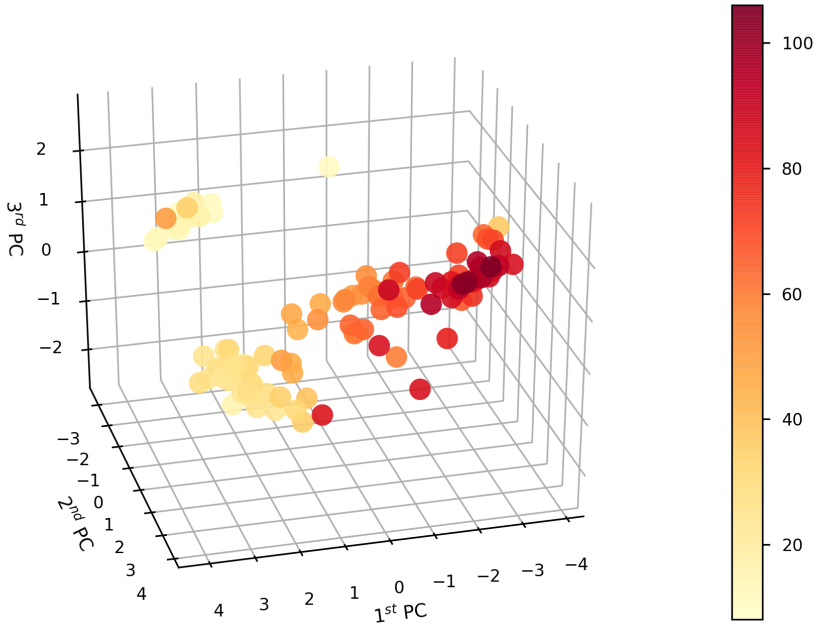


Figure 5.3: Scatter plot obtained after projecting the data in a $3D$ space via linear PCA. The color-coding represents the age of the individuals.

5.4 Metabolic age prediction

This section presents the development of the regression model for the age prediction task presented in this chapter.

To this aim, we imputed the small fraction of missing values following the k-nearest neighbors (with $k = 3$) proposed in [Troyanskaya et al., 2001]. So, the development of the predictive model can take advantage of the total number of individuals ($n = 118$).

The fist question we pose is whether the collected dataset is large *enough* for our purpose, or if we should keep enrolling new volunteers. Providing a definitive answer for this question is, in general, unfeasible. Nevertheless, we can investigate toward this direction by evaluating the so-called *learning curves* [Murphy, 2012].

Such graphical insight is obtained by iteratively fitting a given regression model on increasingly large chunks of the entire dataset. Horizontal axis corresponds to the number of training samples, while vertical axis represents mean values of some cross-validated regression metric, such as MAE. The shape of the obtained learning curve provides relevant information about the prediction problem at hand. For instance, if the cross-validation error keeps decreasing we may sense that there may be *more* to learn about the input/output relationship and that the regression performance would benefit from a larger dataset. Conversely, if the cross-validation error initially decreases, eventually reaching a plateau, it may be that the number of samples is large enough and collecting more data would not or would only marginally improve the prediction performance. This may happen mainly because:

1. the available feature set only partially explains the input/output relationship and more/better variables should be observed to improve the regression performance;

2. the selected model is incapable of capturing some of the input/output relationship hidden

in the data;

3. data are too noisy.

Learning curves plot gives also information on the amount of bias and variance affecting a given predictive model. Four our purpose, we rely on the use of standard Ridge regression model (see Section 3.1.1.2) and 100-splits Monte Carlo cross-validated MAE to evaluate each point of our learning curves (see Section 5.5).

In EDA we realized that most of the input variables show a quasi-linear relationship with the age. Moreover, we also realized that many molecular biomarkers present a strong pairwise linear correlation. Therefore it seems sensible to investigate whether a polynomial feature expansion of degree D could be beneficial for the age prediction. To this aim, we devised the following experimental design.

A cross-validated estimate of the empirical distribution of four regression scores: R^2, MAE, MSE and EV (defined in Section 3.4.3) is evaluated on 500 Monte Carlo random splits. This strategy consists in iteratively extracting $n_{\text{train}} = 0.75 \cdot n$ random samples multiple times[3]. A supervised learning model is fitted on each obtained training set. Once the model is fitted, the four regression scores are evaluated on the remaining $n_{\text{test}} = n - n_{\text{train}}$ samples.

The adopted supervised model consists in a pipeline having two steps: (i) data standardization and (ii) regression model fitting. The first step simply consists in subtracting the mean from each feature and dividing them by their standard deviation. Nesting this preprocessing step inside the Monte Carlo cross-validation scheme improves the empirical estimate of the regression scores. In fact, for each cross-validation iteration, the mean and the standard deviation are estimated from the training set only. This allows to have, each time, a genuine score, estimated only on data points that were never seen before by the current supervised regression pipeline.

Moreover, for the second step of the pipeline we adopted the following regression models: (a) Ridge, (b) Lasso, (c) Elastic-Net, (d) Linear SVM, (e) RBF Kernel SVM, (f) RBF Kernel Ridge, (g) Random Forests, (h) Gradient Boosting and (i) Multilayer Perceptron. The free parameters of each model, including the degree of the polynomial expansion D, are optimized via grid-search 5-fold cross-validation.

This Monte Carlo cross-validated procedure is evaluated two times: with and without a preliminary polynomial feature expansion. We expect linear models to benefit more from the polynomial expansion than the nonlinear ones. Then, for each model and for each metric, we evaluated the p-value obtained from the one-tailed two-sample Kolmogorov–Smirnov test [Everitt and Skrondal, 2002]. This let us understand in which cases the scores obtained after a polynomial expansion are significantly better than their counterpart, obtained only with linear features.

The final goal of this section is to find the best age prediction model. Therefore, it is important to understand which are the most predictive variables. So, as a side result of the previous analysis, we ranked the features according to their selection frequency. The most important features are more likely to be selected more often. This strategy is known in literature as *selection stability* framework (see Section 3.2) and, for our purposes, it let us get a preview of which features will likely appear in the final model.

Finally, the proposed age predicting model is achieved by fitting the best regression strategy on a

[3] The fraction of data to use for training is chosen accordingly to the learning curve in Figure 5.4a, as described in Section 5.5.

training set obtained by randomly extracting 75% of the whole dataset. This model is eventually tested, by the usual metrics, on the remaining 25%.

5.5 Results

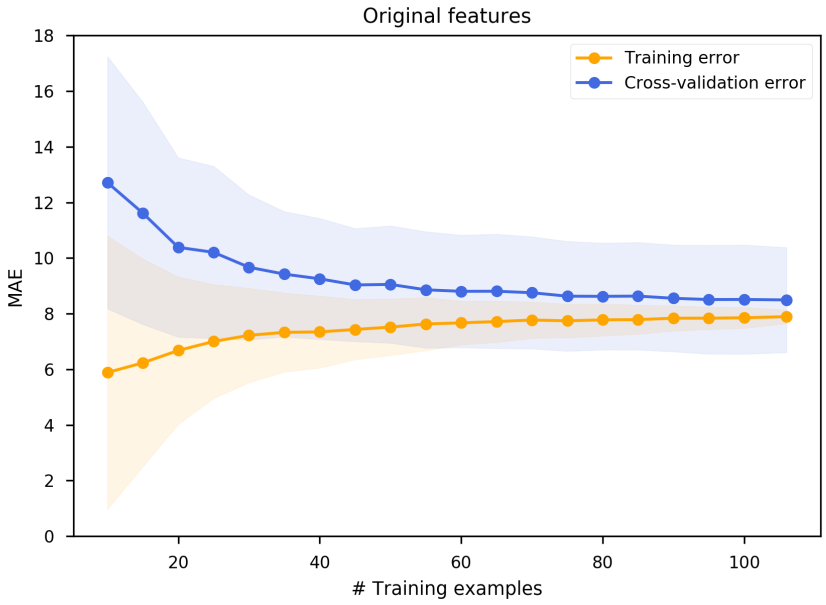
Figure 5.4a shows the obtained learning curves. Interestingly, the average cross-validation error (blue line), reaches a plateau around MAE ≈ 9 years a little bit after 80 samples. This suggests that adding more training examples would not significantly improve the predictions. This justifies the choice of dimensioning the Monte Carlo training sets as 75% of the input data, as described in Section 5.4. Moreover, we can see that the gap between training and cross-validation error becomes smaller as the number of sample increases. This suggests that the linear regression model is generalizing well. We can also notice that the training variance decreases from left to right. This suggests that, as expected, increasing the number of training examples induces a stabilization effect on the learned function.

We repeated the learning curve experiment after a degree $D = 2$ polynomial feature expansion, see Figure 5.4b. At a first glance, we can see two main differences: (i) at the right hand side of the plot, the cross-validation curve (blue line) is still, slowly, decreasing as it does not reach any plateau, (ii) the gap between training and cross-validation curves is wider. Therefore, we suspect that increasing the number of collected sample would significantly improve the regression performance in this case. For consistency with the previous case, we opted to anyway randomly extract training sets of 75% of the dataset after the polynomial expansion as well. What we describe here is a real-world case study and, at the time of writing, expanding the data collection is unfortunately not possible. Nevertheless, we strongly believe that it would be beneficial for the study, therefore we mark this as future work.

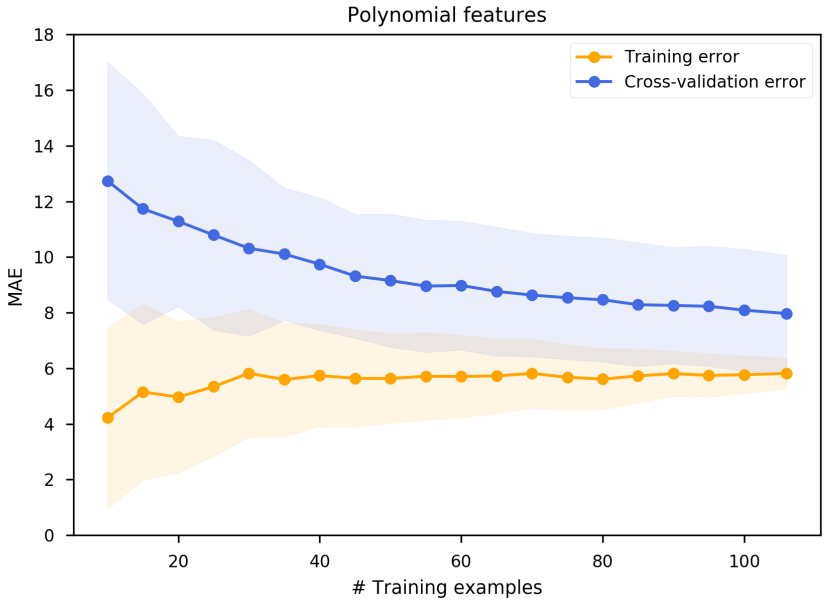
Table 5.2 shows the results of the model assessment performed as described earlier. Focusing on linear features (top half of the table), we can see that Random Forests is the top performing method (in bold). We can also notice that RF is immediately followed by SVM with RBF kernel. This suggests that some nonlinear input/output relationship is hidden in the data. So, we shall investigate whether a simple 2^{nd}-degree polynomial expansion exposes such hidden data structure.

Looking at the bottom half of the Table 5.2, we can see that all the linear models benefit from the polynomial expansion. Quite surprisingly, two linear methods, namely linear SVM and the Lasso, even outperform Random Forests, according to almost every metric.

Table 5.3 highlights the cases in which the metrics evaluated after the polynomial expansion are statistically significantly better than their counterpart evaluated with linear features only (with p-value < 0.01). As expected, the polynomial expansion is beneficial for all the linear method, while it is not for almost every nonlinear method. A separate comment can be made for MLP. In this context, given the reduced number of training samples, we opted for a shallow network with only one hidden layer. The number of hidden units in the hidden layer is considered as a free parameter of the model and it is therefore chosen via grid-search cross validation. As we can see from Table 5.2, MLP is consistently the worst performing regression method. This can partially be explained with the fact that neural networks are known to be more powerful when trained with large datasets, which is not the case. Interestingly, MLP is the only nonlinear method that statistically significantly benefits from the polynomial feature expansion. This result looks quite surprising at a first glance, but actually we can speculate that the effect of the polynomial



(a)



(b)

Figure 5.4: Learning curves obtained on the aging problem by fitting a Ridge regression model on 100 Monte Carlo random splits and evaluating the MAE on each training (orange line) and test (blue line) sets. Panel (a) shows the learning curves obtained using only the original features, whereas panel (b) shows the results achieved after a degree 2 polynomial feature expansion.

	PF	MAE	MSE	EV	R2
Ridge	✗	8.85±1.17	141.04±43.49	0.82±0.06	0.81±0.06
Lasso	✗	8.76±1.19	134.29±40.43	0.83±0.06	0.82±0.06
Elastic-Net	✗	8.82±1.19	137.74±42.66	0.82±0.06	0.82±0.06
Linear SVM	✗	8.72±1.39	149.79±55.16	0.80±0.08	0.80±0.09
RBF Kernel SVM	✗	7.86±1.35	126.29±47.62	0.84±0.06	0.83±0.07
RBF Kernel Ridge	✗	8.37±1.31	143.93±56.30	0.82±0.07	0.81±0.08
Random Forests	✗	**7.66±1.26**	**121.16±44.47**	**0.84±0.06**	**0.84±0.06**
Gradient Boosting	✗	8.26±1.31	146.31±56.86	0.81±0.08	0.80±0.08
MLP	✗	1.00±1.40	166.30±47.47	0.79±0.06	0.77±0.08
Ridge	✓	8.21±1.27	135.94±52.26	0.83±0.08	0.82±0.08
Lasso	✓	7.87±1.21	**116.61±39.34**	**0.85±0.06**	**0.84±0.06**
Elastic-Net	✓	7.87±1.30	118.27±40.51	0.85±0.06	0.84±0.06
Linear SVM	✓	**7.54±1.30**	120.03±46.97	0.86±0.07	0.84±0.07
RBF Kernel SVM	✓	7.91±1.34	131.00±49.34	0.83±0.07	0.82±0.07
RBF Kernel Ridge	✓	8.22±1.31	141.23±54.79	0.82±0.08	0.81±0.08
Random Forests	✓	7.61±1.23	118.20±43.99	0.85±0.06	0.84±0.06
Gradient Boosting	✓	8.19±1.46	142.00±58.45	0.82±0.08	0.81±0.08
MLP	✓	9.40±1.78	183.53±107.98	0.76±0.15	0.75±0.16

Table 5.2: The performance assessment of various ML methods on the problem described in this chapter. Values are expressed as: mean ± standard deviation. The PF flag is ✓ for metrics obtained after a second degree polynomial features expansion, and ✗ for linear features only. Bold digits correspond to column-wise best values.

	p-value MAE	p-value MSE	p-value EV	p-value R2
Ridge	$4.44 \cdot 10^{-14}$	$6.22 \cdot 10^{-03}$	$5.06 \cdot 10^{-03}$	$6.22 \cdot 10^{-03}$
Lasso	$1.09 \cdot 10^{-28}$	$3.39 \cdot 10^{-12}$	$8.49 \cdot 10^{-12}$	$3.27 \cdot 10^{-11}$
Elastic-Net	$1.76 \cdot 10^{-22}$	$9.92 \cdot 10^{-10}$	$9.92 \cdot 10^{-10}$	$4.96 \cdot 10^{-09}$
Linear SVM	$6.20 \cdot 10^{-26}$	$5.07 \cdot 10^{-11}$	$4.33 \cdot 10^{-10}$	$2.84 \cdot 10^{-10}$
RBF Kernel SVM	$4.65 \cdot 10^{-01}$	$9.54 \cdot 10^{-02}$	$1.96 \cdot 10^{-02}$	$8.29 \cdot 10^{-02}$
RBF Kernel Ridge	$3.28 \cdot 10^{-02}$	$1.80 \cdot 10^{-01}$	$2.75 \cdot 10^{-01}$	$2.75 \cdot 10^{-01}$
Random Forests	$2.75 \cdot 10^{-01}$	$3.01 \cdot 10^{-01}$	$3.28 \cdot 10^{-01}$	$3.81 \cdot 10^{-01}$
Gradient Boosting	$1.60 \cdot 10^{-01}$	$7.18 \cdot 10^{-02}$	$1.25 \cdot 10^{-01}$	$7.18 \cdot 10^{-02}$
MLP	$1.95 \cdot 10^{-13}$	$2.33 \cdot 10^{-08}$	$8.31 \cdot 10^{-07}$	$8.31 \cdot 10^{-07}$

Table 5.3: One-tailed p-value resulting from the two-sample Kolmogorov-Smirnov test used to investigate whether or not these ML methods perform better after a second degree polynomial expansion. The green cells indicate the cases that benefit from the polynomial expansion (p-value < 0.01).

expansion could be compared to the effect of the addition of another hidden layer. In fact, it is known that deeper architectures, even if harder to train, usually lead to better results.

From the insights above it is clear that the proposed model uses a degree 2 polynomial feature expansion. However, Table 5.2 indicates two possible linear regression models: linear SVM and the Lasso. In this context we choose to use the Lasso to accomplish the age prediction task for two main reasons: (i) it outperforms linear SVM for 3 metrics out of 4 and (ii) its sparsity enforcing penalty leads to a more compact and interpretable model.

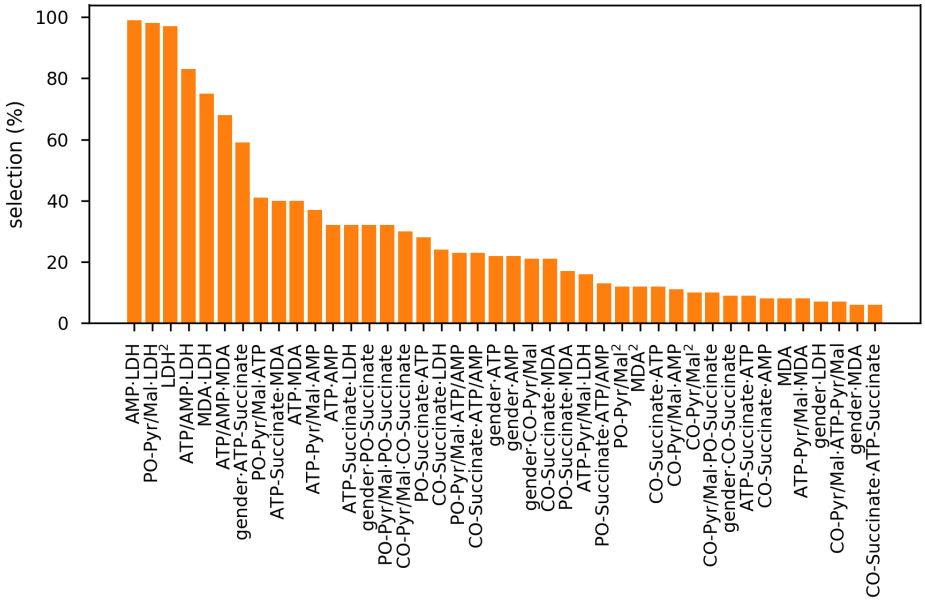


Figure 5.5: The feature ranking obtained via stability selection on the aging problem adopting the Lasso regression model.

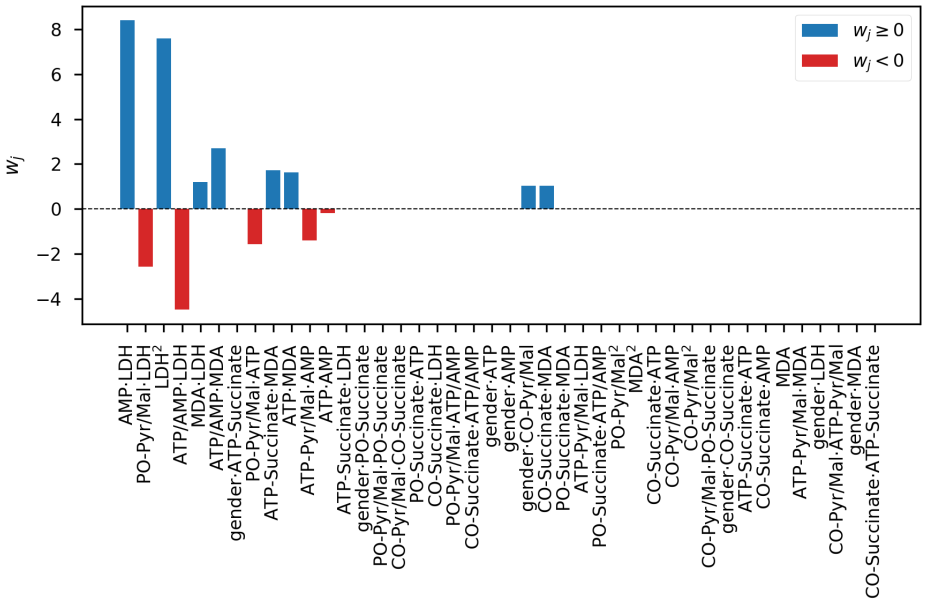


Figure 5.6: The coefficients of the proposed age prediction model.

Analyzing the feature selection frequency of the Lasso, Figure 5.5, we can see that all the top-ranked variables are features that arise from the polynomial expansion. This is a further evidence of the importance of such step.

The final Lasso model, fitted on a 75% of the dataset after the 2^{nd} degree polynomial expansion and a feature-wise standardization leads to the coefficients of Figure 5.6. Such model, evaluated on the test set, achieves MAE $= 6.40$ years, MSE $= 97.24$ years^2, $R^2 = 0.81$ explaining the 82% of the variance. Which is consistent with what we expected from Table 5.2.

Nevertheless, as the Lasso regression model is trained on preprocessed data, the coefficients in

Figure 5.6 cannot be used to predict the age from raw measures. To this purpose, we also report the prediction model, with raw data-ready coefficients, in Equation (5.1).

$$\begin{aligned}\hat{\text{age}} = {} & 1.13 \times 10^{-1} \cdot (\text{GENDER} \cdot \text{CO-PYR/MAL}) + 9.63 \times 10^{-3} \cdot (\text{CO-SUCCINATE} \cdot \text{MDA}) - \\ & 1.91 \times 10^{-1} \cdot (\text{ATP-PYR/MAL} \cdot \text{AMP}) + 3.31 \times 10^{-2} \cdot (\text{ATP-SUCCINATE} \cdot \text{MDA}) \\ & 1.17 \cdot (\text{PO-PYR/MAL} \cdot \text{ATP}) - 1.45 \times 10^{-2} \cdot (\text{PO-PYR/MAL} \cdot \text{LDH}) - \\ & 6.73 \times 10^{-1} \cdot (\text{ATP} \cdot \text{AMP}) + 5.03 \times 10^{-1} \cdot (\text{ATP} \cdot \text{MDA}) + \\ & 9.52 \times 10^{-2} \cdot (\text{AMP} \cdot \text{LDH}) + 5.30 \times 10^{-1} \cdot (\text{ATP/AMP} \cdot \text{MDA}) - \\ & 2.28 \times 10^{-2} \cdot (\text{ATP/AMP} \cdot \text{LDH}) + 8.08 \times 10^{-4} \cdot (\text{MDA} \cdot \text{LDH}) + \\ & 1.84 \times 10^{-4} \cdot \text{LDH}^2 + 29.1\end{aligned} \tag{5.1}$$

5.6 Conclusions and future works

This chapter presented the first biomedical data challenge of the thesis. The goal of this task is to investigate the changes of energy metabolism during the physiological aging by means of a set of metabolic biomarkers obtained from mononuclear cells isolated from peripheral blood.

After a preliminary, and insightful EDA, we realized that the measured variables are highly correlated and that they also show a strong trend with the age. Therefore, we investigated the use of several linear and nonlinear regression models used in combination with a degree 2 polynomial feature expansion.

We devised a Lasso-based regression model that, once trained on the 75% of the dataset, predicted the age of the remaining 25% with MAE of 6.591 years (MSE $= 82.182$, $R^2 = 0.901$ and EV $= 0.902$). This result is in line with what observed in the experiments summarized in Table 5.2.

To the best of our knowledge this is the first attempt to build an age predicting model that takes into account such metabolic biomarkers. Our goal here is not to devise an accurate and ready-to-use age prediction model, whereas we aimed at verifying that the energetic state of blood cells by itself is already a good predictor of the age of a subject.

In the next future we plan to extend this study by enrolling more volunteers in order to expand the number of samples in the dataset. We also plan to exploit this method to investigate possible metabolic alterations of adult patients that suffered from blood tumor in their childhood, were treated with chemotherapy and are currently considered in remission.

6 Temporal prediction of multiple sclerosis evolution from patient-centered outcomes

In this chapter, we investigate the use of patient-centered outcomes to predict the evolution of multiple sclerosis and to assess its impact on patients' lives. Multiple Sclerosis is a degenerative condition of the central nervous system that affects nearly 2.5 million of individuals in terms of their physical, cognitive, psychological and social capabilities. Despite the high variability of its clinical presentation, *relapsing* and *progressive* multiple sclerosis are considered the two main disease types, with the former possibly evolving into the latter. Recently, the attention of the medical community toward the use of patient-centered outcomes in multiple sclerosis has significantly increased. Such patient-friendly measures are devoted to the assessment of the impact of the disease on several domains of the patient life. To this aim, we build a novel temporal model based on random forests classification and multiple-output elastic-net regression. The model provides clinically interpretable results along with accurate predictions of the disease course evolution.

6.1 Introduction: the evolution of multiple sclerosis

Multiple Sclerosis (MS) is a neurodegenerative and chronic disease of the central nervous system characterized by damages to the myelin sheaths, resulting in a wide range of symptoms, such as fatigue, numbness, visual disturbances, bladder problems, mobility issues and cognitive deficits.

People with MS (PwMS) are mainly classified according to their disease course: relapsing-remitting (*RR*), secondary-progressive (*SP*), primary-progressive (*PP*), progressive-relapsing (*PR*) and benign (*B*) [Giovannoni et al., 2016] . Neurological disability in *RR* patients is mainly due to the development of multifocal inflammatory lesions and it results in relapses, that are attacks of neurological worsening (*i.e. relapses*), followed by partial or complete recovery. Disability accrues predominantly in progressive courses (*SP*, *PP*, *PR*) that are more characterized from diffuse immune mechanisms and neurodegeneration. Benign MS occurs when the patient remains fully functional in all neurologic systems for at least 15 years after the onset. Figure 6.1 shows a representative disability progression of MS patients according to their disease course.

An estimated 15% of PwMS have a *PP* or *PR* course at the onset, the remaining 85% is diagnosed with a *RR* course. About 80% of *RR* patients develop *SP* course within 15–20 years if untreated, or if the adopted pharmacological and rehabilitative protocols are not continuously adjusted according to the evolution of the disease [Scalfari et al., 2014].

For this reason, the prediction of the transition from *RR* to *SP* is one of the most important methodological gaps that MS researchers are currently addressing. The availability of a statistical model able to predict disease worsening is one of the major unmet needs that could significantly improve timeliness, personalization and, consequently, the efficacy of the treatments. Nowadays, there are no clear clinical, imaging, immunologic or pathologic criteria to foresee the transition

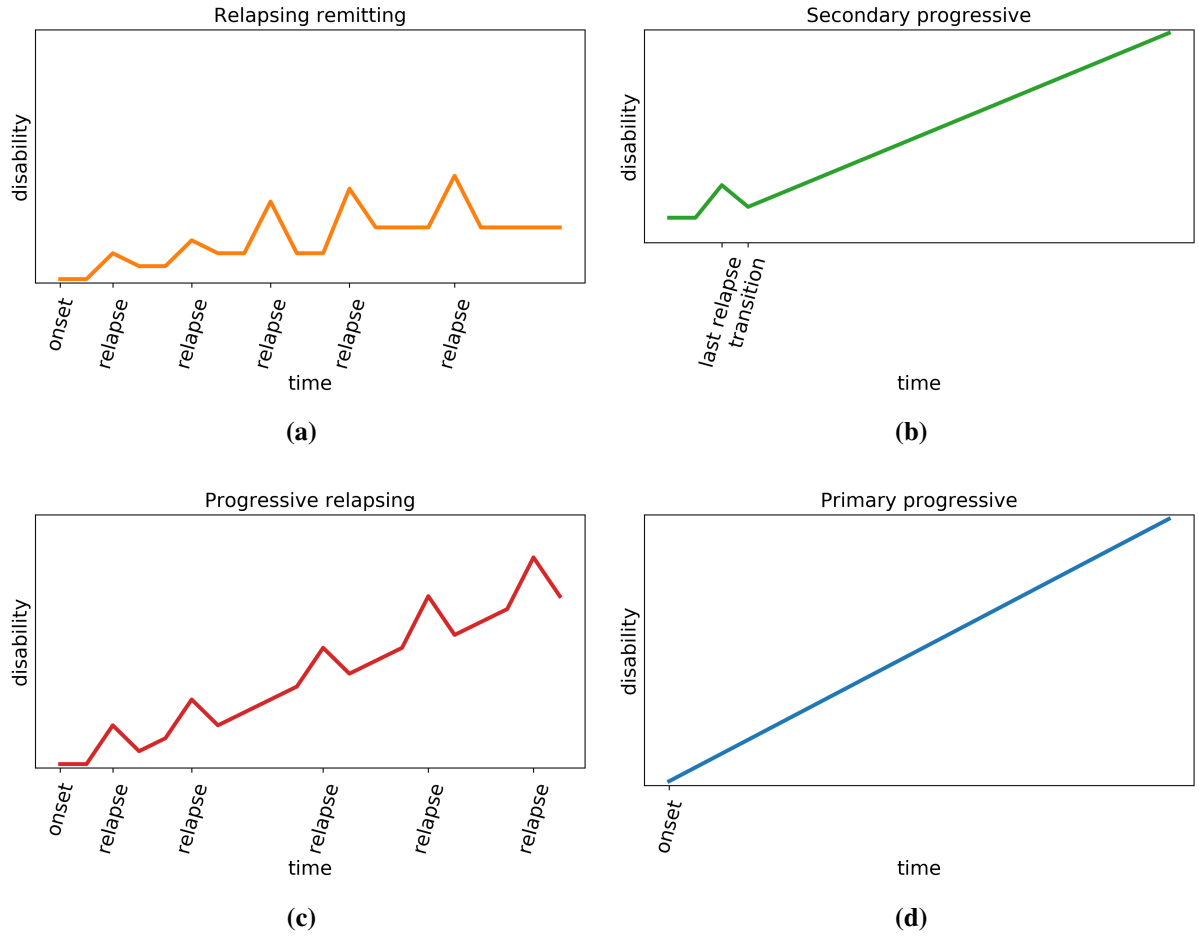


Figure 6.1: Disability evolution of the four main MS courses: panel (a) shows a prototypical *RR* patient, characterized by time-limited attacks which may or may not leave permanent deficits; panel (b) shows *SP* typical disability progression, that is steady with no more relapses; panel (c) represents a typical *PR* disability evolution, which is characterized by steady disability progression from the onset; panel (d) shows a *PR* patient; which has a steady disability progression from the onset with relapses.

from *RR* to *SP* [Lublin et al., 2014]. Several clinical factors relating to possible *SP* course predictors have been identified [Bergamaschi et al., 2015; Dickens et al., 2014]. However, as showed by [Vukusic and Confavreux, 2003], studies investigating on prognostic factors for MS course evolution generally suffer from two shortcomings: they report a higher proportion of *RR* patients not monitored enough to reach progressive course and they lead, to some extent, to contradictory results. Currently, MS research mainly focuses on developing and assessing drugs and rehabilitative protocols for *RR* patients disregarding progressive courses.

6.2 PCOs data collection

In the recent past, researchers explored the potential role of Patient-Centered Outcomes (*PCO*) to follow the progression of neurodegenerative diseases and to take timely decisions [Black, 2013]. PCOs comprise self- and physician-administered tests, questionnaires and clinical scales consisting of either ordinal or categorical scaled answers. As opposed to stressful, not frequently repeatable and expensive clinical exams, like magnetic resonance imaging or blood tests, PCOs are patient-friendly and low-cost measures that could allow to investigate the individual changes and disease impact on several aspects such as physical, cognitive, psychological, social and well-being domains [Fiorini et al., 2015]. To date, PCOs are extensively used to assess general health status, to support diagnosis and monitor progress of disease and to quantify the patients' perception of the effectiveness of a given therapy or procedure [Nelson et al., 2015]. Nevertheless, it is still unclear which are the most informative PCOs and, contextually, whether they can be used as *predictors* for disease evolution.

The biomedical data science challenge presented in this chapter is based on a PCO dataset acquired from a cohort of PwMS progressively enrolled within an ongoing funded project [1].

Each patient is evaluated every four months through the items of the PCOs reported in Table 6.1 which cover physical, cognitive and psychosocial domains. PCO data are intrinsically noisy due to the subjectivity of self-reported measures provided by the patients that can be influenced by personal feelings and opinions. In order to ameliorate this issue, 4 questionnaires out of 10 are administered by medical staff which is trained to keep a homogeneous level of evaluation. A comprehensive description of the PCOs involved in the study is presented below.

MFIS This is a 21-item self-reported questionnaire typically administered in 5 to 10 minutes. MFIS provides an assessment of the effects of fatigue in terms of physical, cognitive, psychosocial functioning and it is considered a valuable tool by clinicians.

HADS This is a 14-item self-reported questionnaire typically administered in 2 to 6 minutes which aims at detecting clinically significant symptoms of depression and anxiety in patients. HADS consists in 7 questions for depression and the remaining 7 for anxiety.

LIFE This is an 11-item self-reported questionnaire which investigates patients quality of life. LIFE can be administered in approximately 5 minutes. To each of the 11 items, the patient can assign an ordinal score $0, 1, 2$ which corresponds to "*disagree*", "*not sure*" and "*agree*" answers.

[1] Ethical review committee approval *023REG2014* was obtained for this work.

Acronym	Full name	Reference
MFIS	Modified fatigue impact scale	[Flachenecker et al., 2002]
HADS	Hospital anxiety and depression scale	[Honarmand and Feinstein, 2009]
LIFE	Life satisfaction index	[Franchignoni et al., 1999]
OAB	Overactive bladder questionnaire	[Cardozo et al., 2014]
EDINB	Edinburgh handedness inventory	[Oldfield, 1971]
ABILH	Hand ability index	[Arnould et al., 2012]
FIM	Functional independence measure	[Granger et al., 1990]
MOCA	Montreal cognitive assessment	[Dagenais et al., 2013]
PASAT	Paced auditory serial addition task	[Aupperle et al., 2002]
SDMT	Symbol digit modality test	[Parmenter et al., 2007]
EDSS	Expanded disability status scale	[Kurtzke, 1983]

Table 6.1: The set of available PCOs. The first 6 are self-reported, while the last 5 are administered by trained medical staff. In our analysis all PCOs were used, with the exception of EDSS.

OAB This is an 8-item self-reported questionnaire which investigates patients bladder control. OAB can be administered in approximately 5 to 8 minutes and it is a reliable tool to investigate possible stress or discomfort lead by unexpected urinary urgencies that patients may experience during day or night.

EDINB This 10-item self-reported questionnaire can be used to assess dominance of a person's right/left hand during daily activities. The items are very straightforward, so EDINB can be administered in 3 to 5 minutes.

ABILH This is a 23-item self-reported questionnaire which can be used to measure hand ability in adults with upper limb impairments. ABILH assesses a person's ability to manage daily activities that require the use of the upper limbs, whatever the strategies involved. ABILH is usually administered in 5 to 10 minutes.

FIM This is 19-item clinical scale assessing the amount of assistance required for the patient to carry out activities of daily living. FIM is typically administered by trained examiner in 35 to 40 minutes and it covers both motor and cognitive domains.

MOCA This is an 11-item clinical scale assessing several cognitive domains such as short-term memory recall, visuospatial abilities, phonemic fluency, attention, concentration and so on. MOCA is typically administered in less than 10 minutes.

PASAT This clinical scale is a measure of cognitive function that assesses patients' auditory processing speed and flexibility, as well as their calculation ability. It can be administered in 10 to 15 minutes and it consists in audio stimuli in which single digits are presented every 3 seconds. The patient is asked to add each new digit to the one immediately prior to it. PASAT must be administered by trained examiner.

SDMT This clinical scale is a test for organic cerebral dysfunctions. This test simply involves a substitution task: using a reference key the patients has few seconds to pair specific numbers with specific geometric measures. SDMT is typically administered by trained examiner in less than 5 minutes.

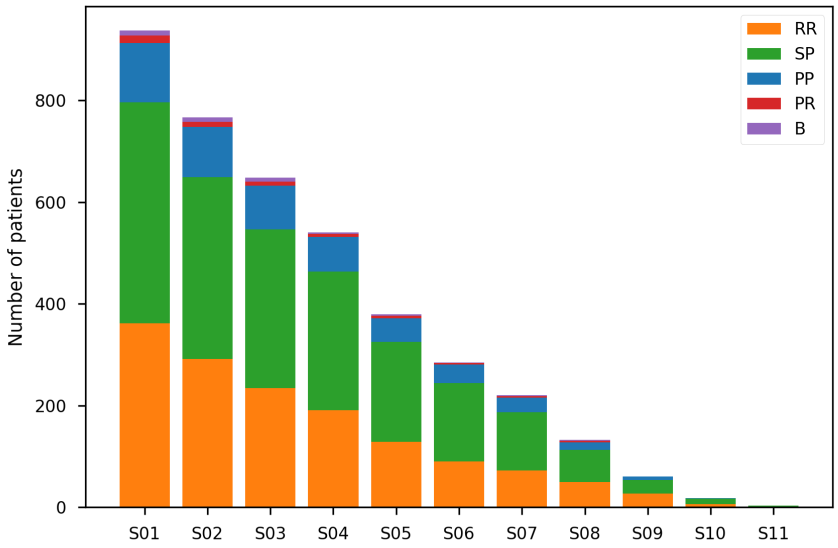


Figure 6.2: Bar chart of the number of MS patients in each disease form at different examinations.

EDSS This is probably the oldest assessment instrument for MS. EDSS is based on a neurological examination consisting of 7-items. Each item rates a different function, all the items are then combined in the final EDSS score which is an ordinal scale ranging from 0 (normal neurological examination) to 10 (death due to MS), in half-point increment. The use of EDSS in modern MS assessment is somewhat controversial. Although usually adopted as an index of the disability level, EDSS focuses mainly on deambulation disability without taking into account other aspects that could impact patient disability, such as upper limb or cognitive functions [Meyer-Moock et al., 2014; Uitdehaag, 2014].

In our analysis we considered all these PCOs except EDSS.

The collected PCO dataset comprises additional information such as: i) number of relapses in the last four months (NR), ii) educational level expressed in terms of total years of education (EDU), iii) height (H) expressed in cm and iv) weight (W) expressed in kg. Moreover, a neurologist assigns to each patient the corresponding disease course.

6.3 PCOs exploratory data analysis

In this work we analyze PCOs data acquired every four months from a cohort of MS patients enrolled in a funded study. Currently, we have collected data for 11 examinations and, as patients enrollment is still ongoing, the number of individuals for each time point is successively decreasing, as shown in Figure 6.2.

The collected dataset comprises a grand total of 3991 patients, with 1451 *RR*, 1947 *SP*, 503 *PP*, 53 *PR* and 37 benign cases, as shown in Figure 6.3.

Each sample of the dataset is represented by a vector containing the 145 predictors summarized in Table 6.1. As the missing data ratio amounts to 1.61% of the entire dataset, we resort to the

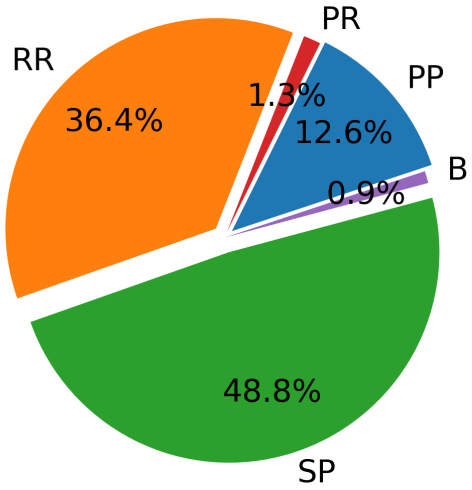


Figure 6.3: Representation of the distribution of the total amount of acquisitions, divided according to the disease form.

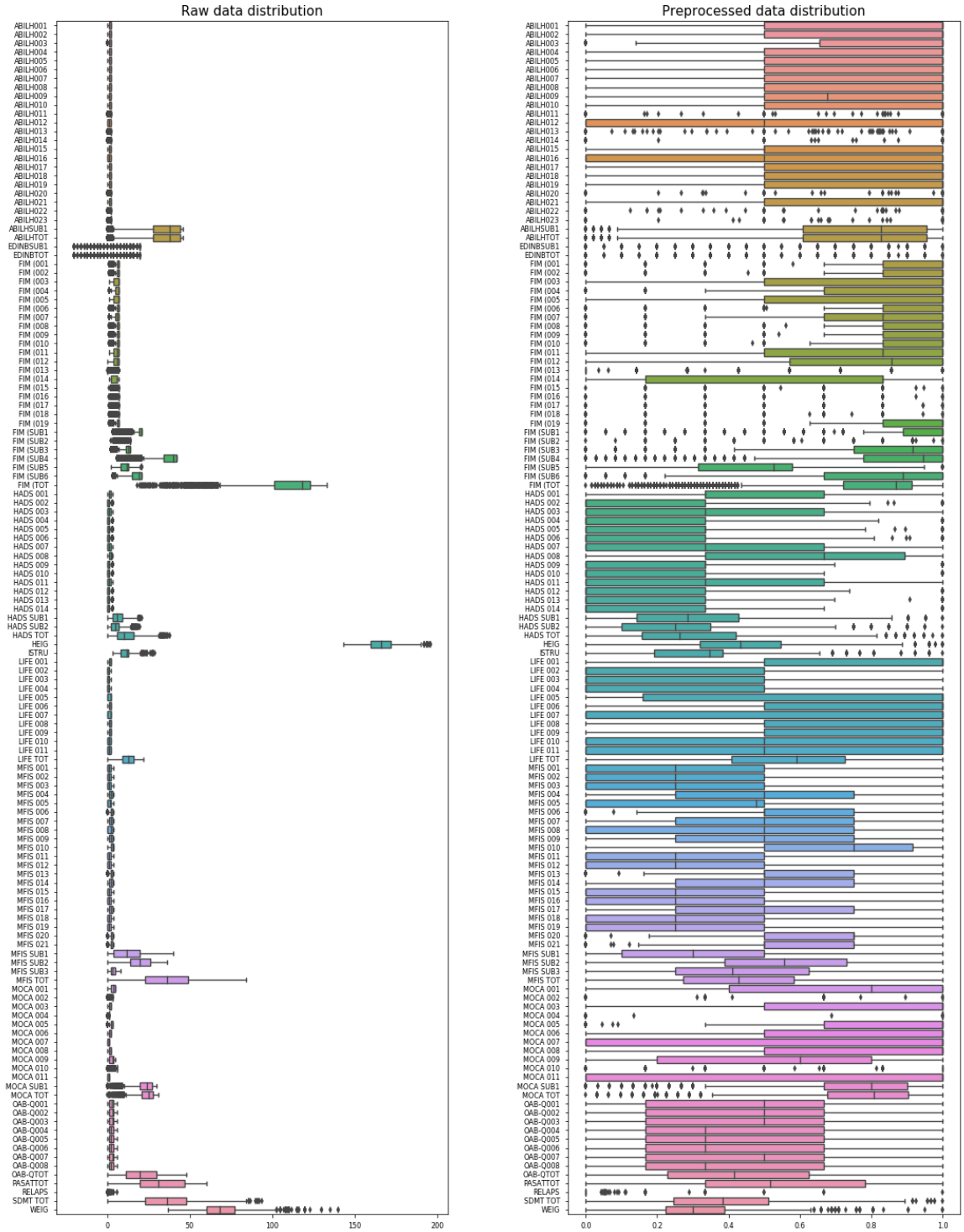
k-nearest neighbor data imputing strategy (with $k = 3$) proposed in [Troyanskaya et al., 2001].

Analyzing PCO data is challenging from several respects. For instance, items belonging to different questionnaires are encoded with numerical values in different ranges. For example, the items of the MFIS questionnaire have ordinal scale values in $[0 - 4]$, whereas the SDMT outcome is the global number of correctly answered items of the test (max 110) and the EDINB test consists in 10 categorical items measuring the dominance of right or left hand in the activities of daily living. To tackle such issues, in this EDA we opted for a preliminary data preprocessing of the ordinal answers and a binary one-hot-encoding of the categorical ones, the latter increases the dimensionality of the samples, leading to $d = 165$ variables in the dataset. The adopted data preprocessing strategy, namely min-max scaling, consists in casting each feature $\boldsymbol{x}^j$ in a fixed range, *i.e.* $\boldsymbol{x}^{j'} \in [0, 1]$. The min-max scaling is obtained by the transformation described in Equation (6.1).

$$\boldsymbol{x}^{j'} = \frac{\boldsymbol{x}^j - \min(\boldsymbol{x}^j)}{\max(\boldsymbol{x}^j) - \min(\boldsymbol{x}^j)} \tag{6.1}$$

The effect of the data preprocessing on the ordinal input variables is visually represented in Figure 6.4. As we can see, this preprocessing step allows to compare more easily the input features.

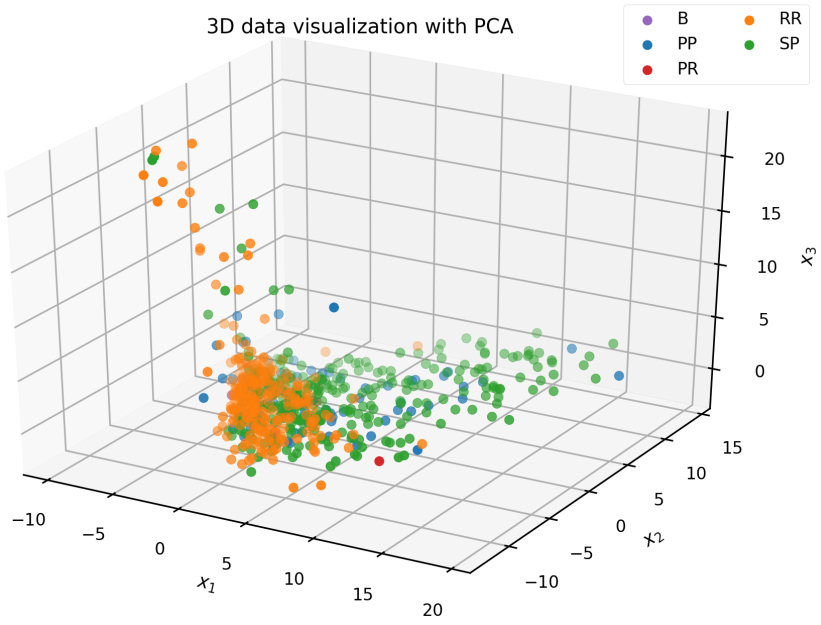
Furthermore, in order to visually inspect the data we investigate how to reduce their dimensionality. To this aim, we project the data in a 3D space with linear PCA and Isomap (see Section 3.3.2). These algorithms are sensitive to outliers, which we expect to affect our dataset. Therefore, we follow a preliminary isolation forests-based anomalies detection and removal as described in [Liu et al., 2008, 2012]. The obtained scatter plots are shown in Figure 6.5. As we can see, both the obtained projections hint suggest a class separation between *RR* and *SP* subjects, while the same conclusion cannot be drawn for the other classes. Moreover, considering only the first three principal components, PCA explains only the 37.4% of the variance of the dataset. This suggests that most of the information, which may be useful for classification purposes, is spread



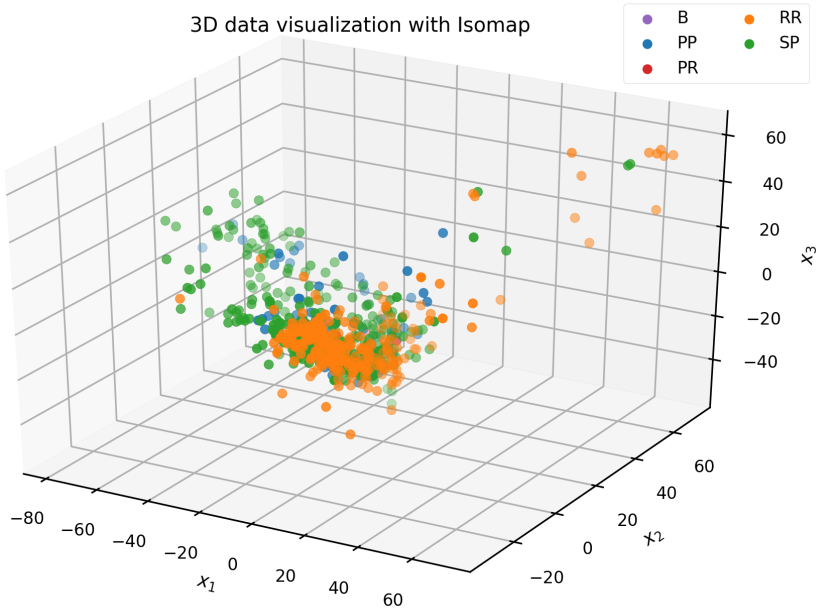
(a) **(b)**

Figure 6.4: The effect of the data preprocessing on the input PCOs. The left panel (a) shows the distribution of the raw collected variables, whereas the right panel (b) shows the distribution of the same variables after the preprocessing step.

across several input variables.



(a)



(b)

Figure 6.5: A random extraction of the 20% of the MS dataset projected on a 3D space by linear PCA, on panel (a), and Isomap, on panel (b).

This EDA raised our hopes to successfully perform a further supervised MS course classification.

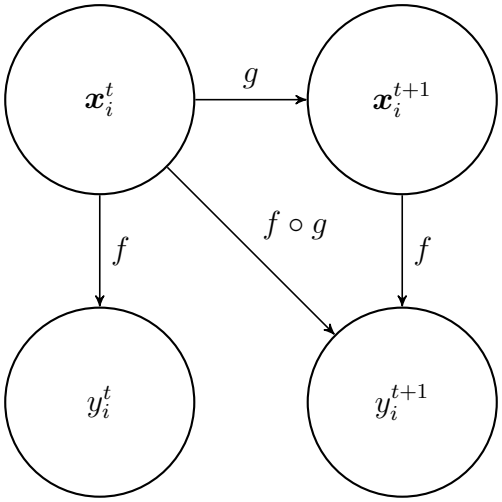


Figure 6.6: A visual representation of the temporal structure assumed in the collected data. When the two functions f (CCA) and g (PEP) are learned, the FCA model $f \circ g$ can be used to predict the evolution of the disease course for future time points y_i^{t+1}.

6.4 Supervised analysis

In order to develop a temporal model of MS evolution, we assume that our data can be modeled according to the temporal structure outlined in Figure 6.6. Therefore, predicting the MS course evolution can be split in three different related tasks, which we call: Current Course Assignment (CCA), PCOs Evolution Prediction (PEP) and Future Course Assignment (FCA).

CCA Given the 165-dimensional representation of a patient at a fixed time point $\boldsymbol{x}_i^t$, this task consists in assigning the corresponding disease course y_i^t. CCA can be translated into a binary classification problem which can be solved by learning a discriminative function $f(\boldsymbol{x}_i^t) = y_i^t$.

PEP Given the historical representation of a patient $\boldsymbol{x}_i^t$ for $t = 1, \ldots, \tau$, this task consists in predicting the patient representation $\boldsymbol{x}_i^{\tau+1}$. PEP can be seen as a multiple-output regression function and it can be solved by learning an appropriate function $g(\boldsymbol{x}_i^t) = \boldsymbol{x}_i^{t+1}$.

FCA This task can be seen as foreseeing the MS disease course $y_i^{\tau+1}$ from $\boldsymbol{x}_i^t$ for $t = 1, \ldots, \tau$. Once $\hat{f}(\boldsymbol{x})$ and $\hat{g}(\boldsymbol{x})$ are learned by training on historical PCO data, the FCA problem is finally solved by the temporal model $\hat{f} \circ \hat{g}(\boldsymbol{x}_i^t) = y_i^{t+1}$. In time-series data analysis, this is known as *one-step-ahead forecast*. Notably, the FCA model allows to foresee if the patient at the next time point is likely to experience a transition from ***RR*** to ***SP***, or not.

6.4.1 Experimental design

Our final goal is predicting MS course evolution of ***RR*** and ***SP*** patients, hence the subjects with ***PR***, ***PP*** and benign forms are not further taken into account.

We considered all the patients with a minimum of 1 time point (the most recently enrolled) up to $T = 11$ time points for a total of 3398 samples, of which 1451 ***RR*** and 1947 ***SP***. Following the

EDA, we opted for a preliminary feature-wise min-max scaling. However, to promote unbiasedness of the results, this preprocessing phase is not performed on the entire data collection, but it is embedded into the model fitting procedure. This trick, jointly applied with cross-validation techniques, guarantees that even the feature-wise range used for data preprocessing is *learned* from training set and it is applied on previously unseen validation sets. However, the experimental design adopted to learn $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ is slightly different. Therefore, they are separately discussed in the remainder of this section.

CCA The CCA model $f(\boldsymbol{x})$ solves a binary classification problem: to each input $\boldsymbol{x}_i^t$ is associated an output y_i^t that encodes the corresponding MS disease course (*RR* or *SP*) with a binary label. We split the dataset in three temporal chunks, namely *training*, *validation* and *test* sets, consisting of all samples collected at time points $t = 1, 2, 3$, $t = 4$ and $t = 5, 6, 7, 8, 9, 10, 11$, respectively. Accordingly, we used 1993 samples for training $f(\boldsymbol{x})$, 463 for validation leaving the remaining 942 for test.

Seven candidate models, see Section 6.4.2, for $f(\boldsymbol{x})$ are fitted on 100 Monte Carlo random sampling of the training set each time keeping $\frac{1}{4}$ of the samples aside, see Section 3.4.1. For each Monte Carlo sampling the fitting procedure is performed on the remaining $\frac{3}{4}$ of the samples and it includes an inner parameter optimization via grid-search cross-validation, as described in Section 3.4.2. In particular, we require the MS course prediction to be based on a reduced number of variables, therefore we enforce sparsity in each candidate model. Leveraging on this Monte Carlo-based stability selection strategy, we rank the variables according to their selection frequency, see Section 3.2. Once a variable ranking is achieved for each candidate model, the list of selected variables is identified by thresholding the corresponding ranking with the threshold that maximizes the MCC on the validation set. Finally, the last training step consists in fitting each candidate model on the union of training and validation sets taking only into account the corresponding reduced subset of selected variables. The final CCA model $\hat{f}(\boldsymbol{x})$ is chosen as the one that performs better on the previously unseen test set in terms of accuracy, MCC, precision, recall and F_1 score. These performance metrics for classification are defined in Section 3.4.3.

PEP On the other hand, learning the PEP model $g(\boldsymbol{x})$ implies solving a multiple-output regression problem as to each input $\boldsymbol{x}_i^t$ is associated the output vector $\boldsymbol{x}_i^{t+1}$. Therefore, we can only consider samples at time point t with an available follow-up at the next time point $t + 1$, which reduces the overall number of available samples. The dataset splitting is consistent with the one followed for learning $f(\boldsymbol{x})$, although in this case there is no need for a separate validation set, as learning $g(\boldsymbol{x})$ does not require any variable selection process. We used the samples collected at time points $t = 1, 2, 3, 4$ for training and those at $t = 5, 6, 7, 8, 9, 10, 11$ for test, resulting in 1946 and 714 samples, respectively. The fitting procedure includes an inner parameter optimization via grid-search cross-validation. Each candidate model is a function $g : \mathbb{R}^{165} \to \mathbb{R}^k$ where k is the number of variables selected by the best CCA model. Three candidate models are evaluated to solve this problem. The final PEP model $\hat{g}(\boldsymbol{x})$ is chosen as the candidate model that performs better on the previously unseen test set in terms of MAE, defined in Section 3.4.3.

FCA The predictive capability of the FCA model $\hat{f} \circ \hat{g}(\boldsymbol{x})$ is finally evaluated on the test set. The CCA model $\hat{f}(\boldsymbol{x}_i^t)$ predicts the MS course $\hat{y}_i^t$ from the PCO data vector $\hat{\boldsymbol{x}}_i^t$ that, in turn, is predicted by the PEP model $\hat{g}(\boldsymbol{x}_i^{t-1})$. We shall notice here that the predictions $\hat{f} \circ \hat{g}(\boldsymbol{x}_i^t) = y_i^{t+1}$ for $t = 11$ are foreseeing possible *RR* to *SP* transitions that are beyond

our data observation, hence predictions at the last time point cannot be used to assess the FCA model performance. Therefore, its performance is evaluated on 616 test samples.

6.4.2 Learning $f(x)$

We imposed $f(\boldsymbol{x})$ to be sparse. This requirement is helpful from two distinct respects: (i) predictive model performance increases thanks to a reduced effect of the course of dimensionality [Hastie et al., 2015] and (ii) the identification of a reduced subset of meaningful PCOs provides easily interpretable results for the clinicians.

In order to achieve a sparse model, we take advantage of two variable selection strategies: embedded and wrapper methods, see Section 3.2. When using embedded methods, we exploited the sparsity inducing penalties of three models:

- Elastic-Net, that presents square loss and a penalty which is a convex combination of ℓ_1- and ℓ_2-norm of the variable weights (see Section 3.1.1.4),
- sparse logistic regression, which combines the logistic loss with a Lasso penalty on the variable weights (see Section 3.1.1.6),
- ℓ_1-penalized SVM (see Section 3.1.1.7).

We expect Elastic-Net to exploit correlations between PCOs, which we assume in our dataset. On the other hand, the use of sparse logistic regression and ℓ_1-penalized SVM can benefit from the renowned classification capability of their loss function. Moreover, we applied the RFE wrapper method to four other methods:

- logistic regression, that achieves smooth solutions thanks to its ℓ_2-norm penalty;
- SVM, which also has an ℓ_2-norm penalty;
- Random Forests (RF), a tree-based bagging ensemble that: (i) can capture nonlinear relationship between PCOs and disease course, (ii) it is well suited to deal with categorical/ordinal variables and (iii) thanks to the inner bagging strategy, it is less prone to overfitting;
- Gradient Boosting (GB), a tree-based boosting ensemble that has recently demonstrated to be an excellent nonlinear model for structured data [Chollet, 2018], but it is more prone to overfitting.

As shown in Section 3.2, using embedded methods it is possible to achieve a list of selected variable for each Monte Carlo iteration. Conversely, RFE produces a variable ranking. The list of selected variables, in this case, is obtained by a further nested K-fold cross-validation optimization (with $K = 3$).

6.4.3 Learning $g(x)$

As no prior information on the relationship between PCOs evaluated at different time points was available, to learn $g(\boldsymbol{x})$ we investigated the use of both linear and nonlinear models.

	MCC	Accuracy	Precision	Recall	F_1-score
GB	0.688 ± 0.035	0.846 ± 0.017	0.863 ± 0.021	0.858 ± 0.023	0.860 ± 0.016
RF	0.687 ± 0.032	0.845 ± 0.016	0.865 ± 0.019	0.853 ± 0.021	0.859 ± 0.014
Elastic-Net	0.612 ± 0.029	0.804 ± 0.015	0.850 ± 0.031	0.789 ± 0.047	0.817 ± 0.018
ℓ_2 LR	0.620 ± 0.032	0.809 ± 0.016	0.857 ± 0.020	0.787 ± 0.027	0.820 ± 0.016
ℓ_1 LR	0.632 ± 0.031	0.814 ± 0.016	0.867 ± 0.019	0.786 ± 0.026	0.824 ± 0.016
ℓ_2 SVM	0.621 ± 0.032	0.808 ± 0.016	0.863 ± 0.019	0.779 ± 0.026	0.819 ± 0.017
ℓ_1 SVM	0.632 ± 0.030	0.814 ± 0.015	0.872 ± 0.020	0.779 ± 0.025	0.823 ± 0.016

Table 6.2: Classification performance scores of the seven candidate models (for the CCA problem) achieved on 100 Monte Carlo cross-validation iterations and expressed in terms of average $\pm$ standard deviation. GB and RF outperform linear models, while performing almost identically.

Concerning the linear models, we explored two different solutions: Nuclear Norm Minimization (NNM) and Multi-task Elastic-Net (MTEN), see Section 3.1.1.5 and 3.1.1.4, respectively. The first imposes a low-rank prior on the result. The second is a natural multiple-output extension of Elastic-Net, hence it induces a row-structured sparsity pattern on the solution where collinear variables are more likely to be included in the model together. For nonlinear prediction, we resorted to a Multi-layer Perceptron (MLP) approach, see Section 3.1.5.1.

6.5 Results and discussion

We shall separately discuss the results achieved in terms of CCA, PEP and FCA models.

CCA The first step toward the definition of this model consisted in the Monte Carlo-based stability selection assessment of the seven candidate models. Table 6.2 summarizes the performance scores obtained by the seven models across the 100 cross-validation iterations, expressed in terms of average $\pm$ standard deviation. The two tree-based methods perform quite similarly and consistently outperform the linear models.

The stability selection procedure leads to the definition of a variable ranking for each of these models (results not shown). Such variable rankings are then cut according to the threshold of the selection frequency that gives best MCC on the validation set. The number of variables selected by each model is then represented in Figure 6.7. Interestingly, RF and GB achieve the most parsimonious representation (only 33 variables out of 165) and the top cross-validation scores. This confirms that the relationship between PCOs and disease course is better represented by a nonlinear estimator and that the CCA problem can be solved by observing a reduced number of PCO variables.

The second step consisted in refitting each model on the union of training and validation sets always considering only the corresponding relevant features. The seven candidate models on the test set reach the performance summarized in Table 6.3 and visually represented in Figure 6.8. In this case, the RF method outperforms the other candidate models, therefore we chose it as CCA model $\hat{f}(\boldsymbol{x})$.

Insights on the use of PCOs for MS assessment are provided by the sparsity of the CCA model induced by the RFE schema with RF. The 33 selected variables are reported in Table 6.4. Comparing the full list of PCO questionnaires of Table 6.1 with Table 6.4, we

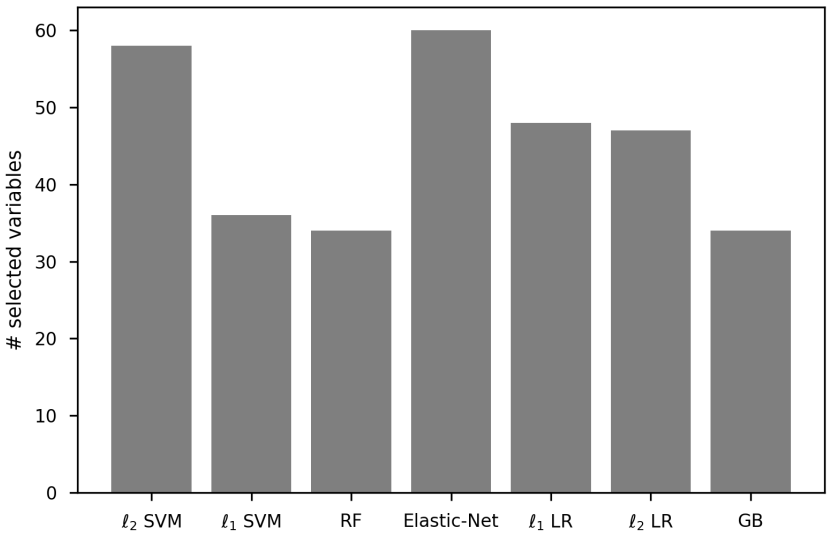


Figure 6.7: The number of variables selected by each of the seven candidate models.

	MCC	Accuracy	Precision	Recall	F_1-score
GB	0.664	0.839	0.873	0.858	0.865
RF	**0.712**	**0.860**	**0.904**	**0.860**	**0.881**
Elastic-Net	0.588	0.785	0.900	0.725	0.803
ℓ_2 LR	0.596	0.792	0.895	0.744	0.812
ℓ_1 LR	0.594	0.794	0.885	0.758	0.817
ℓ_2 SVM	0.600	0.789	0.892	0.740	0.809
ℓ_1 SVM	0.594	0.792	0.891	0.747	0.813

Table 6.3: Classification performance scores of the seven candidate models (for the CCA problem) achieved on the test set.

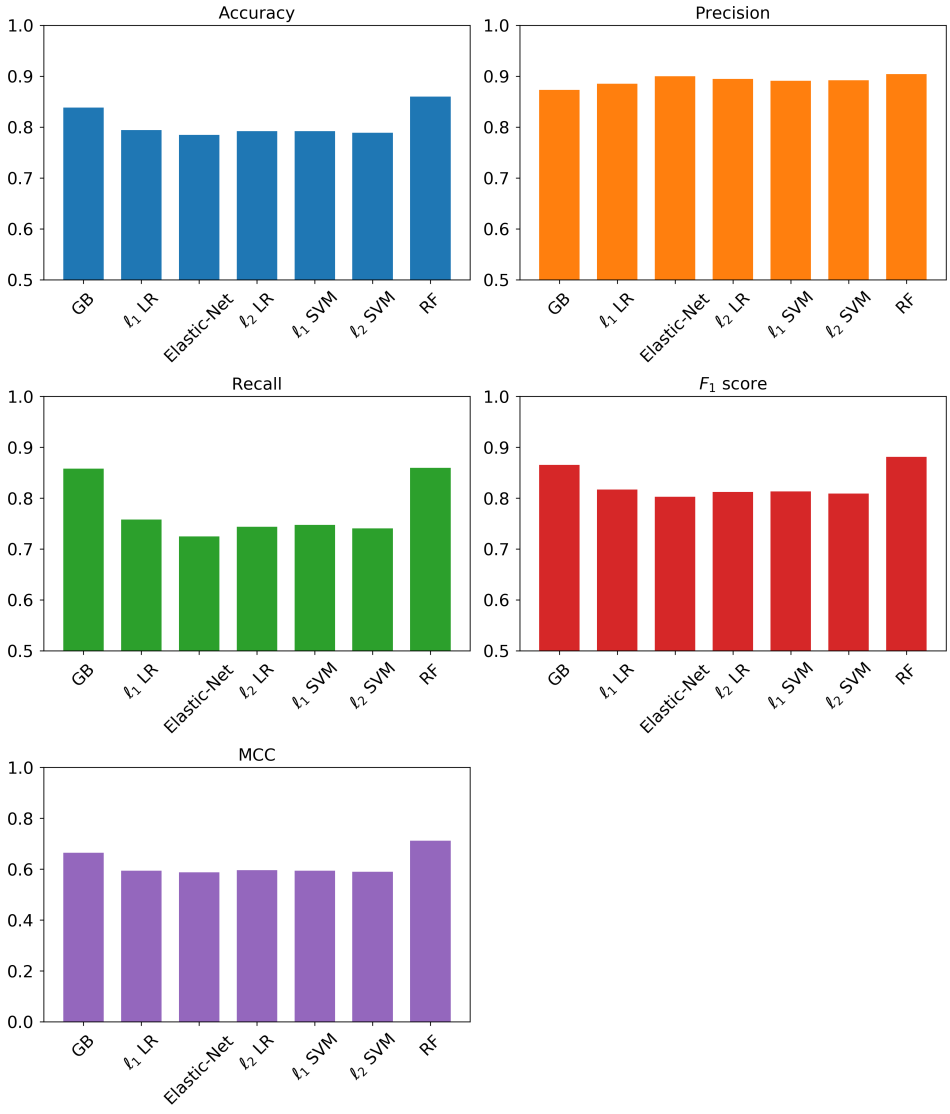


Figure 6.8: Classification performance scores of the seven candidate models (for the CCA problem) achieved on the test set.

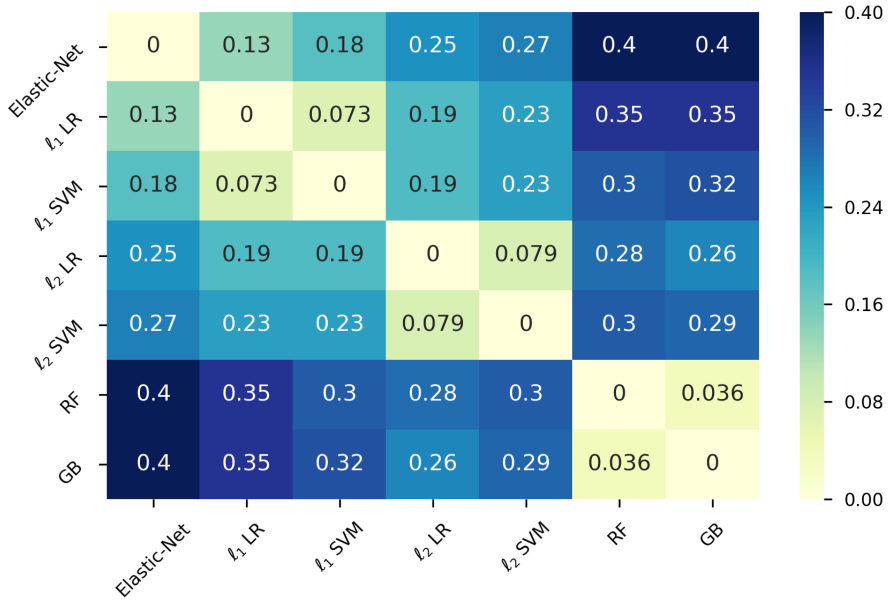


Figure 6.9: Heatmap displaying the distance between the lists of variables selected by each model in terms of their hamming distance.

observe that each PCO used in this study is represented at least once, except EDINB, and the most represented is FIM. We also see that, whenever possible, the model tends to select aggregate scores (total and subtotal) rather that single items. This is consistent with the clinical practice, where neurologists are more likely to assess patient's health status by using the aggregate scores, rather than the single questions. Quite surprisingly, the recent number of relapses is the only additional information not selected by the model. Finally, we note that all the domains that are known to be affected by the disease are well covered: mobility (upper and lower limbs), cognition, emotional, fatigue, bladder and psychosocial.

The heatmap in Figure 6.9 shows the Hamming distance of the lists of variables selected by the seven CCA candidate models. Observing this figure, a block structure clearly emerges. Interestingly, at a higher level, we can see that linear methods are more prone to select similar variables with respect to tree-based methods. Moreover, across linear methods, ℓ_1-based and RFE-based methods select more similar variables, as their Hamming distance is lower. Whereas, as expected, Elastic-Net selects collinear variables and its Hamming distance from the other linear models is higher.

PEP MTEN outperforms the other candidate models in terms of MAE ($\text{MAE}_{\text{MTEN}} = 0.099$, $\text{MAE}_{\text{NNM}} = 0.101$, $\text{MAE}_{\text{MLP}} = 0.107$), hence we select it as our PEP model $\hat{g}(\boldsymbol{x})$.

FCA Finally, the FCA model $\hat{f} \circ \hat{g}(\boldsymbol{x})$, obtained by combining MTEN and RF achieves the following performance scores on the 616 test samples: accuracy 0.826, precision 0.882, recall 0.828, F_1-score score 0.854 and MCC score of 0.642. Predicting the MS course at the next time point is of course a harder task than the predictions at the current time. However, the performance loss of FCA with respect to CCA, mainly due to the accumulation of errors introduced by PEP, are relatively small.

Selected item	Description
ABILH (12)	Hammering a nail
ABILH (TOT)	Sum of all the ABILH subscores
EDU	Total years of formal education
FIM (10)	How much assistance is required for toilet transfer
FIM (11)	How much assistance is required for shower transfer
FIM (12)	How much assistance is required for locomotion (ambulatory)
FIM (14)	How much assistance is required for locomotion (wheelchair)
FIM (SUB_3)	FIM subtotal measuring global sphincter control
FIM (SUB_4)	FIM subtotal measuring global personal care
FIM (SUB_5)	FIM subtotal measuring global locomotion
FIM (SUB_6)	FIM subtotal measuring global mobility
FIM (TOT)	FIM total score
HADS (7)	I can sit at ease and feel relaxed
HADS (SUB_1)	HADS subtotal measuring global level of anxiety
HADS (SUB_2)	HADS subtotal measuring global level of depression
HADS (TOT)	HADS total score
H	Height of the individual in cm
LIFE (TOT)	LIFE total score
MFIS (2)	I have had difficulty paying attention for long periods of time
MFIS (SUB_1)	MFIS subtotal measuring global cognitive level
MFIS (SUB_2)	MFIS subtotal measuring global physical level
MFIS (SUB_3)	MFIS subtotal measuring global psychosocial level
MFIS (TOT)	MFIS total score
MOCA (1)	MOCA visuoconstructional skill test
MOCA (9)	MOCA memory test
MOCA (SUB_1)	Sum of all the MOCA subscores
MOCA (TOT)	MOCA score corrected for individuals with less than 12 years of formal education
OAB (1)	Frequent urination during the daytime hours
OAB (4)	Accidental loss of small amounts of urine
OAB (TOT)	OAB total score
PASAT	PASAT score
SDMT	SDMT score
W	Weight of the individual in kg

Table 6.4: The list of PCOs items selected by GB with RFE.

6.6 Conclusions and future works

This chapter describes a temporal model based on PCOs and ML for disease form prediction in MS. In particular, we address the tasks of current course assignment, PCOs evolution prediction and future course assignment. The model is built on a collection of PCOs acquired on a cohort of individuals enrolled in an ongoing funded study (*DETECT-MS PRO*).

The proposed temporal model was able to correctly assign the current MS form and to foresee future ones with accuracy of $\approx 86.0\%$ and $\approx 82.6\%$, respectively. This demonstrates that PCOs can effectively be used to build an effective ML-based MS disease course predictor.

In the next future, we plan to further investigate the predictive capabilities of the proposed model with longer temporal horizons and to compare it with different approaches, such as probabilistic graphical models.

Given the achieved promising results, the proposed model is soon going to be validated in clinical practice, where it will assist the clinicians involved in this study to foresee possible disease course transition and to take important decisions concerning treatment and therapies that can substantially improve the quality of life of their patients.

In the context of neurodegenerative diseases, clinicians typically use PCOs data to corroborate evidences coming from standard quantitative exams [Black, 2013]. Interestingly, in our case the absence of clear *SP* predictors makes the information extracted from PCOs data the only available resource.

In the era of precision medicine, the problem of predicting MS course evolution still relies on stressful exams and clinical judgment. To the best of our knowledge, this is the first work that aims at solving this delicate task leveraging only on patient-friendly measures and ML.

7 Data-driven strategies for robust forecast of continuous glucose monitoring time-series

Over the past decade, continuous glucose monitoring (CGM) has proven to be a very resourceful tool for diabetes management. To date, CGM devices are employed for both retrospective and online applications. Their use allows to better describe the patients' pathology as well as to achieve a better control of patients' level of glycemia. The analysis of CGM sensor data makes possible to observe a wide range of metrics, such as the glycemic variability during the day or the amount of time spent below or above certain glycemic thresholds. However, due to the high variability of the glycemic signals among sensors and individuals, CGM data analysis is a non-trivial task. Standard signal filtering solutions fall short when an appropriate model personalization is not applied. State of the art data-driven strategies for online CGM forecasting rely upon the use of recursive filters. Each time a new sample is collected, such models need to adjust their parameters in order to predict the next glycemic level. In this chapter we will see that the problem of online CGM forecasting can be successfully tackled by personalized machine learning models, that do not need to recursively update their parameters.

7.1 Introduction: modern diabetes care

Diabetes is a chronic metabolic disorder affecting nearly 400 million of individuals worldwide. The number of diabetic patients is increasing and it is expected to reach almost 600 million in the next future [Guariguata et al., 2014]. According to the *World Health Organization* [Organization et al., 2016], the global prevalence of diabetes among adults has nearly doubled in the last few decades, rising from 4.7% in 1980 to 8.5% in 2014. If not treated correctly, diabetes may cause several permanent complications, such as visual impairment and kidney failure.

Hypoglycemia is a severe risk in diabetes therapy. The mean incidence of hypoglycemia in patients with type 1 diabetes (T1D) is 1-2 events per week, while severe hypoglicemia occurs 0.1-1.5 episodes per year [van Beers and DeVries, 2016]. Moreover, hypoglycemia interferes with the quality of life and increases the risks of cardiovascular events in type 2 diabetes (T2D) patients [van Beers and DeVries, 2016]. On the other hand, hyperglycemia associates with an increased risk of diabetes complication as well.

The most common glucose monitoring solutions are self blood glucose meters and Continuous Glucose Monitoring systems (CGM). CGM devices are minimally-invasive, and can be used in daily life for retrospective or online applications [Vigersky and Shrivastav, 2017]. CGM systems measure interstitial glucose concentration at fixed time intervals, enabling an accurate observation of glycemic variability during the day as well as the ratio of time spent in hypo/hyperglycemia. When CGM is employed online, an improvement of the therapy can be achieved by embedding in the system a tool that foresees glucose levels using suitable time-series models trained on past CGM data [Sparacino et al., 2007]. In this case, alarms can be generated when

the glucose concentration exceeds the normal range [Vigersky and Shrivastav, 2017].

To model patient-specific Blood Glucose (BG) levels, several approaches, integrating various information, were proposed [Bunescu et al., 2013; Zecchin et al., 2011]. However, the problem of glycemic level prediction is still challenging, due to the high CGM signal variability among patients and acquisition devices.

This chapter describes the third, and last, biomedical data science challenge of the thesis. Throughout this chapter we assume that CGM signals have structural information on time-changes of BG concentration [Sparacino et al., 2007; Bunescu et al., 2013] and we perform glycemic level forecasting exploring the performance of a set of purely data-driven techniques for time-series analysis.

Data-driven forecasting approaches, as opposed to models driven by an *a-priori* description of the underlying phenomena, are capable of modeling input/output relationships without requiring prior knowledge on the field of use. This family of approaches can be successfully employed to obtain reliable predictions without modeling complex, and possibly not completely understood, environments. Data-driven forecasting models are widely applied in real-world scenarios often employing a moving-window paradigm, *i.e.* the system keeps track of the last w acquisitions using them to forecast future values.

In this chapter, we focus on two main groups of data-driven methods for online time-series forecasting: *recursive filters* and, of course, ML models. The first group includes linear stationary models, such as Autoregressive Moving Average (ARMA), non stationary models, such as Autoregressive Integrated Moving Average (ARIMA) and adaptive filtering techniques, such as the Kalman Filter (KF). These methods are well-established and extensively used, but they require recursive parameters adjustment for every new collected sample [Box et al., 2015]. As these methods are note covered in Chapter 3, a quick overview is provided in Section 7.3.

The second group comprises regularized kernel methods, such as Kernel Ridge Regression (KRR) (see Sections 3.1.1.2 and 3.1.2) and deep learning methods, such as Long Short-Term Memory networks (LSTM), see Section 3.1.5.2.

As we have seen in the previous chapters, ML methods showed very promising results in several applications, also including time-series forecasting [Bunescu et al., 2013; Schmidhuber et al., 2005]. To achieve a predictive model, they need to learn the model parameters from a training set. This learning process is done *only once* and it conveys a model that does not require further parameter adjustments to predict future values. This remarkable advantage makes machine learning models more suitable to be delivered on embedded portable systems.

This chapter aims at understanding whether purely data-driven machine learning methods can be successfully employed to forecast glucose values of diabetic patients.

7.2 Temporal forecasting problem setting

In the previous chapters we fixed the notation for regression and classification problems. The temporal forecasting can be seen as a special regression case, therefore some additional notation is needed.

Given a set of data $\mathcal{S}$, we refer to data-driven models with hyperparameters $\boldsymbol{\theta}$ as $\mathcal{M}_{\mathcal{S}}(\boldsymbol{\theta})$. Given the time-series $y(t)$ we aim at predicting $y(t + \Delta T)$, where ΔT is some *prediction horizon*.

A well-known issue of CGM sensor data analysis is that signal properties, such as the signal-to-noise ratio, may vary among devices and individuals [Facchinetti et al., 2010]. In order to achieve an accurate forecast of the glucose level of a given individual from past samples, any prediction model $\mathcal{M}_{\mathcal{S}}(\boldsymbol{\theta})$ must be *personalized*. Such model personalization procedure is two-fold: *hyperparameters* ($\boldsymbol{\theta}$) *optimization*, also known as model selection (see Section 3.4), and *parameter estimation*, or model fitting. In this work we fixed a common strategy for hyperparameters optimization whilst the actual model fitting procedure is defined differently according to the model.

7.3 Recursive filters overview

Recursive filters are the most widely adopted class of temporal forecasting strategies. In this chapter we make CGM temporal prediction by exploiting two different recursive filters approaches: ARIMA and KF. Providing a comprehensive overview of temporal forecasting strategies is beyond the scope of this thesis, therefore Chapter 3 does not cover this topic. This section sketches the main ideas behind this two strategies, providing the information that are necessary to understand this last biomedical data science challenge.

7.3.1 Autoregressive Integrated Moving Average

ARIMA methods can be used to perform linear forecasting of non stationary time-series assuming that its d-th difference is a stationary ARMA process [Box et al., 2015]. The output of an ARMA process, with white input noise $u(t) \sim \mathcal{N}(0, \sigma^2)$, can be expressed as in Equation (7.1).

$$y(t) = -\sum_{k=1}^{p} a_k y(t-k) + \sum_{k=0}^{q} b_k u(t-k) \tag{7.1}$$

The output of an ARMA(p, q) model can be seen as the sum of p autoregressive and q moving average terms. This strategy requires the modeled processes to be stationary. On the other hand, when a time-series can be considered stationary after d differentiations, ARIMA(p, d, q) models can be used. This is the case for non stationary time-series that exhibit local stationary behavior. In general, (p, d, q) are unknown and we will consider them as hyperparameters of the ARIMA model.

The cross-validation index (see Section 7.6) for ARIMA models can be defined as $J(\boldsymbol{\theta}) = \text{AIC}(\mathcal{M}_{\mathcal{S}}(\boldsymbol{\theta})) + \bar{\varepsilon}_{\text{cv}}$, where AIC stands for Akaike Information Criterion [Box et al., 2015] and $\bar{\varepsilon}_{\text{cv}}$ is the MSE evaluated via cross-validation, see Section 3.4.3. We refer to [Box et al., 2015] for a detailed description of ARIMA model fitting.

The application of this class of models to predict CGM sensor data was also explored in [Sparacino et al., 2007; Bunescu et al., 2013].

7.3.2 Kalman Filter

The KF addresses the problem of estimating the state $\boldsymbol{x} \in \mathbb{R}^d$ of a discrete-time process governed by the linear stochastic difference equation $\boldsymbol{x}(t+1) = F\boldsymbol{x}(t) + \boldsymbol{w}(t)$ with measurements $\boldsymbol{y} \in \mathbb{R}^k$,

$\boldsymbol{y}(t) = H\boldsymbol{x}(t) + \boldsymbol{v}(t)$, where $\boldsymbol{w}(t)$ and $\boldsymbol{v}(t)$ are independent random variables representing state and measurement noise, respectively [Welch and Bishop, 1995]. It is usually assumed that $\boldsymbol{w}(t) \sim \mathcal{N}(\mathbf{0}, Q)$ and $\boldsymbol{v}(t) \sim \mathcal{N}(\mathbf{0}, R)$, with both Q and R unknown. In the context of CGM sensor data prediction, we can safely assume that the state space is two-dimensional, hence $x_1(t) = u(t)$, $x_2(t) = u(t-1)$ where the unknown signal $u(t)$ is described *a-priori* as an integrated random-walk $u(t) = 2u(t-1) - u(t-2) + w(t)$ as in [Facchinetti et al., 2010]. Consequently, the state space transition matrix F can be written as

$$F = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix} \tag{7.2}$$

while the measurement vector is

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

the process noise covariance is

$$Q = \begin{bmatrix} \lambda^2 & 0 \\ 0 & 0 \end{bmatrix} \tag{7.3}$$

and the measurement noise covariance is $R = \sigma^2$, as in [Facchinetti et al., 2010]. Both λ^2 and σ^2 are unknown and we will consider them as hyperparameters of the KF forecasting strategy.

In this case, the cross-validation index (see Section 7.6) can be defined as $J(\boldsymbol{\theta}) = \bar{\varepsilon}_{\text{cv}}$.

The application of KF to predict CGM sensor data was also explored in [Facchinetti et al., 2010; Knobbe and Buckingham, 2005].

7.4 CGM data collection

We acquired CGM samples from a group of 148 T1D and T2D patients wearing the iPro®2 Professional CGM sensor (Medtronic), which reported glucose values every 5 minutes. Patients were monitored for up to 7 days in free living conditions, keeping track of their treatments. From this initial cohort, we excluded the 18 individuals which acquisitions lasted for less than 3.5 days as well as the 24 time-series that presented artifacts due to incorrect use of the CGM acquisition device. Hence, our final dataset comprises 106 subjects of which 72 T1D and 34 T2D. On average, glycemic variability is relatively high, with 170.7 ± 70.0 mg/dL for T1D and 158.4 ± 43.6 mg/dL for T2D. Figure 7.2 shows two examples, one for each diabetes type.

7.5 CGM exploratory data analysis

For each patient, the risk of hypo/hyperglycemia is determined by computing the Low Blood Glucose Index (LBGI) and the High Blood Glucose Index (HBGI), defined as in [Fabris et al., 2016]. LBGI and HBGI are summary statistics, extracted from a series of CGM data, that increases when the frequency and/or the extent of low CGM or high CGM readings increases. These two indices are heavily influenced by the frequency and extent of hypo- and hyperglycemic episodes. To estimate LBGI and HBGI from CGM data we followed [Kovatchev et al., 1997]. Figure 7.1 shows the LBGI and HBGI distributions for T1D and T2D. The green areas at the bottom of each boxplot represents the low risk area for hypo/hyperglycaemia, as reported in literature [Kovatchev et al., 1997]. As expected, the fraction of T1D patients experiencing risks of hyperglycaemia episodes is higher than T2D patients.

Figure 7.1: Plots reporting the distributions of LBGI (left) and HBGI (right) for T1D and T2D. Green areas at the bottom of each plot represents low risk of hypo/hyperglycemia events.

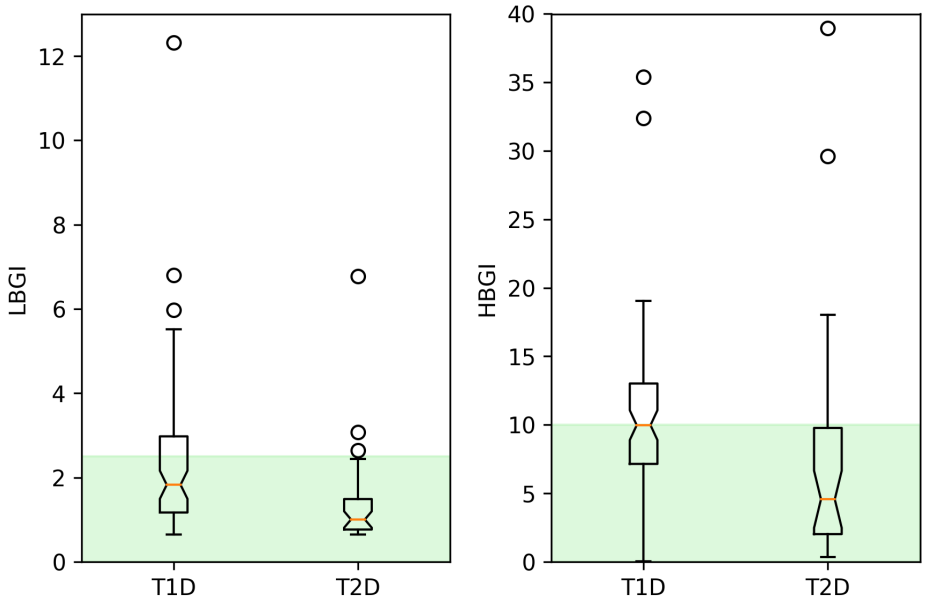
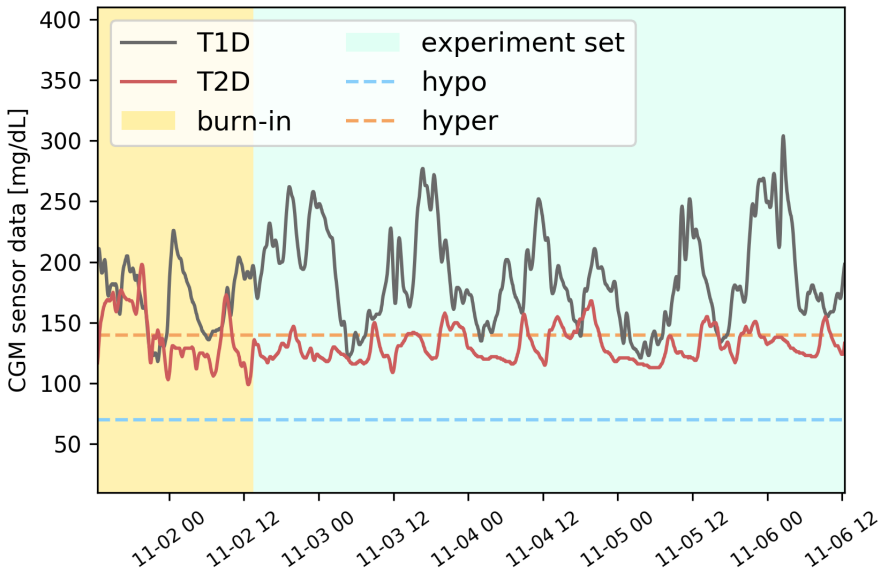


Figure 7.2: An example of two glycemic profiles obtained from T1D and T2D patients. The glucose target range is set between 70 mg/dL and 140 mg/dL (dashed lines). The yellow area at the left hand side of the plot is the initial *burn-in* interval used for model *personalization*.



7.6 Experimental design

We shall now focus on the hyperparameters optimization strategy. Given $y(t)$ for time points $t = 1, \dots, T$, we split the time-series in two chunks: an initial *burn-in* of $T' = 300$ CGM observations and the *experiment set* made of the subsequent $T - T'$ samples (see Figure 7.2). For each model $\mathcal{M}_{\mathcal{S}}(\boldsymbol{\theta})$, and for each subject, we use the burn-in samples to optimize the hyperparameters $\boldsymbol{\theta}$ via grid-search cross-validation. In other words, the best hyperparameters $\boldsymbol{\theta}^*$ are chosen as the ones that minimize an index $J(\boldsymbol{\theta})$ estimated on cross-validation splits and defined differently according to the model, as anticipated in the previous sections.

In the context of time-series, the cross-validation training splits consist only of observations occurred before the corresponding validation samples; such cross-validation flavor is sometimes referred to as *rolling forecasting origin* [Tashman, 2000]. The grid-search cross-validation scheme promotes the identification of models capable of returning robust predictions.

Once the best hyperparameters $\boldsymbol{\theta}^*$ are identified, the model $\mathcal{M}_{\mathcal{S}}(\boldsymbol{\theta}^*)$ is fitted on the whole burn-in and it is used to perform online forecasting on the data points of the experiment set (*i.e.* for $t = T' + 1, \dots, T$).

For each personalized model $\mathcal{M}_{\mathcal{S}}(\boldsymbol{\theta}^*)$ we calculate the accuracy for prediction horizons ΔT of $30, 60,$ and 90 minutes. The performance is estimated by RMSE and MAE, defined in Section 3.4.3, that in the context of temporal forecasting, and assuming n samples in the experiment set, can be rewritten as in Equation (7.4) and (7.5).

$$\text{RMSE} = \sqrt{\frac{\sum_t (y(t + \Delta T) - \hat{y}(t + \Delta T))^2}{n}} \tag{7.4}$$

$$\text{MAE} = \frac{\sum_t |y(t + \Delta T) - \hat{y}(t + \Delta T)|}{n} \tag{7.5}$$

We use $w = 36$ as the size of the window in the moving-window paradigm and we indicate with $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$ the input/output pairs for machine learning model. Therefore, each input $\boldsymbol{x}_i$ is a w-dimensional vector made of the CGM samples acquired at times $t = t_{i-w}, \dots, t_i$, while the corresponding output y_i is the CGM acquisition at time t_{i+1}.

7.7 CGM forecasting results

Taking into account the available information on treatments, we divided the dataset into four groups, namely T1D with microinfusion pump (32 subjects), T1D with insulin injection (40 subjects), T2D with rapid acting insulin (10 subjects) and T2D with other therapies (24 subjects).

For each group, we applied all forecasting models whose performance, expressed in term of MAE and RMSE, is presented in Table 7.1. The forecasting errors clearly increase with the prediction horizon as the errors accumulate at each predicted time step.

KRR achieves the most accurate prediction overall and for almost every group of patients, although ARIMA results are comparable. Moreover, KRR loss of reliability from the first to the last prediction horizon is lower than for ARIMA.

Figure 7.3: Online time-series forecasting obtained by KRR model. *One-step-ahead prediction* (left): the green solid line shows the available samples; the dashed line represents one-step-ahead predictions that are obtained by applying the model on a moving window of 36 time-points. *Open-loop forecast* (right): with passing time, the moving-window incorporates an increasing number of predicted points, accumulating errors; the dashed line represents forecast with a prediction horizon of 90'. Absolute prediction errors are evaluated with respect to future measures (red solid line).

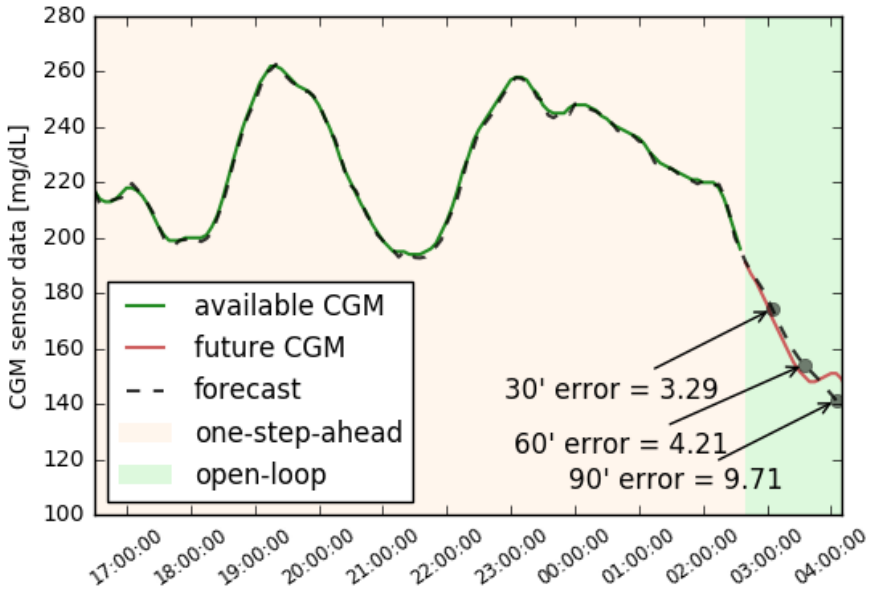


Figure 7.3 displays an example of time-series forecast obtained with KRR showing the three different prediction errors at 30, 60 and 90 minutes.

7.8 Conclusions and future works

This chapter presented the last biomedical data science challenge of the thesis.

We compared the performance of two types of data-driven strategies for online CGM forecasting: recursive filters and ML models. We also illustrate a general procedure for model personalization based on cross-validation with hyperparameters optimization via grid-search.

Finally, we showed that reliable CGM predictions can be obtained with ML models that do not need to recursively adjust their parameters along time.

In the future, to improve the performance on long prediction horizon, we plan to investigate sparse data representations with deeper architectures or regularized dictionary learning approaches.

Table 7.1: Prediction errors (standard deviation) of the forecasting models for increasing prediction horizons, overall and on the four groups. *MP* = Microinfusion Pump, *II* = Insulin Injection, *RAI* = Rapid Acting Insulin, *Other* = Other therapies. Bold numbers indicate best result.

ARIMA

	MAE mg/dL			RMSE mg/dL		
	30	60	90	30	60	90
Overall	11.5 (5.2)	25.6 (12.5)	37.2 (20.3)	16.7 (7.9)	37.1 (19.3)	53.8 (31.8)
T1D - *MP*	13.3 (3.6)	30.3 (7.4)	45.3 (12.5)	19.5 (5.4)	43.9 (10.9)	64.9 (18.1)
T1D - *II*	13.4 (6.3)	31.0 (16.2)	46.1 (27.1)	19.7 (10.6)	45.1 (27.0)	66.6 (45.7)
T2D - *RAI*	**11.0 (3.3)**	**23.8 (6.5)**	33.2 (8.5)	16.2 (4.8)	34.1 (8.3)	46.9 (10.9)
T2D - *Other*	10.0 (5.1)	21.4 (10.4)	29.2 (12.5)	13.6 (6.0)	29.6 (12.7)	41.3 (16.5)

KF

	MAE mg/dL			RMSE mg/dL		
	30	60	90	30	60	90
Overall	46.8 (23.2)	50.8(25.4)	159.6 (43.5)	58.3 (27.7)	63.1 (30.3)	169.9 (47.1)
T1D - *MP*	56.6 (15.9)	59.5 (15.2)	163.5 (26.7)	70.5 (19.0)	74.3 (18.4)	177.5 (29.9)
T1D - *II*	59.9 (24.5)	67.9 (26.7)	179.2 (38.9)	74.4 (28.7)	83.5 (31.3)	193.8 (40.4)
T2D - *RAI*	48.6 (20.4)	50.9 (20.8)	195.2 (63.1)	60.4 (22.9)	63.6 (24.1)	204.4 (63.8)
T2D - *Other*	33.0 (12.1)	35.6 (15.8)	145.0 (30.4)	41.3 (14.2)	44.3 (17.7)	150.1 (31.8)

KRR

	MAE mg/dL			RMSE mg/dL		
	30	60	90	30	60	90
Overall	**11.1 (4.3)**	**25.1 (10.4)**	**35.2 (15.4)**	**15.5 (6.7)**	**33.6 (13.8)**	**45.8 (19.7)**
T1D - *MP*	**12.9 (3.4)**	**30.2 (8.4)**	**43.1 (12.6)**	**17.9 (5.1)**	**40.2 (10.9)**	**56.0 (16.1)**
T1D - *II*	**13.1 (4.9)**	**30.3 (10.6)**	**43.1 (14.5)**	**18.6 (8.4)**	**40.5 (14.5)**	**55.8 (18.6)**
T2D - *RAI*	11.2 (2.9)	24.1 (5.5)	**33.2 (7.1)**	**16.0 (4.7)**	**32.7 (7.5)**	**43.7 (9.5)**
T2D - *Other*	**8.7 (2.3)**	**19.8 (6.9)**	**27.7 (11.8)**	**11.9 (3.4)**	**26.5 (8.9)**	**36.2 (14.4)**

LSTM

	MAE mg/dL			RMSE mg/dL		
	30	60	90	30	60	90
Overall	19.9 (10.6)	44.0 (25.9)	61.3 (36.5)	26.2 (13.4)	55.0 (30.0)	74.7 (41.3)
T1D -*MP*	22.9 (9.1)	51.6 (21.5)	71.6 (27.0)	30.6 (11.6)	65.7 (27.0)	89.8 (33.5)
T1D - *II*	24.5 (12.9)	56.5 (33.1)	81.4 (45.8)	31.2 (14.9)	68.8 (35.9)	97.0 (48.4)
T2D - *RAI*	18.3 (6.7)	39.4 (14.8)	54.1 (21.0)	25.7 (10.1)	51.6 (21.7)	67.7 (29.1)
T2D - *Other*	16.8 (8.9)	35.0 (17.7)	47.0 (26.5)	22.2 (12.1)	43.4 (19.0)	56.5 (27.3)

8 Conclusions

Data science is an evolving cross-disciplinary field that has recently gained the attention of both industry and academia. The main goal of data science is to analyze and understand arbitrarily large data collections in order to devise data-driven and statistically sound decision making strategies. The main characteristics of data scientists is to be are domain experts joining a strong mathematical background with advanced computer science programming skills.

In this context, machine learning is undoubtedly one of the most important data science tools. In fact, machine learning models and algorithms are capable of uncovering hidden structure in the data with the final aim of devising some data-driven prediction strategy. Thanks to this approach, it is possible to tackle complex real-world tasks with little/no human supervision.

In machine learning and data science applications everything revolves around analysis and interpretation of a data collection. However, as it often happens, reaching the desired insights or predictive power is not straightforward. This can happen for several reasons: *e.g.* data can be too large-scale or too high-dimensional, measures can be noisy, different unstratified populations may be represented, *etc*. In this context, exploratory data analysis is a fundamental and insightful procedure that should be carried out in order to guide further predictive model developments.

Chapter 4 is dedicated to the description of ADENINE: an open-source data exploration tool developed for large-scale structured data that can seamlessly run on a single workstation as well as on HPC cluster facility. With ADENINE it becomes easy to run exploratory data analysis on large datasets and to generate publication-ready plots and reports.

Moreover, in this thesis we have discussed three different biomedical data science challenges.

First, we have seen that it is possible to predict the age of healthy individuals by exploiting a sparsity-enforced linear model fitted on a polynomial expansion of a set of molecular biomarkers that are measured from peripheral blood mononuclear cells. Accurate data exploration, model development and assessment for this problem are described in Chapter 5.

Then, we have explored the use of patient centered outcomes for the assessment of quality of life in multiple sclerosis patients. These tests consist in a set of self-reported questionnaires and clinical scales administered by trained clinical staff. In Chapter 6 we presented a temporal model that can predict the disease evolution from the initial relapsing-remitting course, to the secondary-progressive form, from the answers provided by an individual to a number of patient centered outcomes. The use of this temporal model will soon be validated in clinical practice. where it can be used to take timely decisions aimed at improving management and treatment of the disease, hence increasing patients' quality of life.

Finally, in Chapter 7 we have investigated the use of continuous glucose monitoring systems and data-driven forecasting models of the glycemic level in type 1 and type 2 diabetic patients. We have empirically shown that it is possible to achieve reliable predictions, at increasing prediction horizons, exploiting personalized kernel-based machine learning model that, as opposed to recursive filters strategies, do not need to recursively adjust their parameters in time.

Throughout the chapters of this thesis, particular attention is paid towards providing a rigorous

cross-validation-based performance assessment of the proposed data-driven models. This approach is fundamental in order to tackle biomedical data science challenges with actionable solutions, that can be effectively applied in clinical practice.

A Appendix

As already pointed out at the beginning of Chapter 3, ML is a cross-disciplinary field and the statistical tools used in literature to describe models and algorithms heavily depend on the academic background of the author. This can make the approach to ML fascinating and somewhat cumbersome at the same time.

The goal of this appendix is to shed light on some of the statistical tools and definitions that are typically left unsaid, or given for granted, by most of the authors. In particular, in the following sections insightful statistical details on the formulation of the supervised learning problem expressed in Equation (3.2) will be provided.

A.1 Useful theorems and definitions

This first section lists the theorems and the definitions that are useful for the comprehension of the following sections [Keener, 2011; Everitt and Skrondal, 2002].

Theorem 1 (Law of the unconscious statistician) *Given two continuous random variables $(a, b) \in A \times B$ with joint probability distribution $p(a, b)$, the expected value of the function $g(a, b)$ can be stated as follows.*

$$\mathbb{E}[g(a,b)] = \iint_{A\times B} g(a,b)\, p(a,b)\, dadb$$

Definition 1 (Conditional probability) *Given two events A and B, the conditional probability of A given B is defined as*

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

where B is not an impossible event, i.e. $P(B) > 0$.

Theorem 2 (Bayes rule) *Given A and B two events with probability $P(A)$ and $P(B) \neq 0$, the conditional probability of observing A given that B is true is*

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

where $P(B|A)$ is the probability of observing B given that A is true.

Definition 2 (Well-posed problem) *A problem is **well-posed** if its solution: (i) exists, (ii) is unique, (iii) depends continuously on the data (e.g. it is stable).*

Definition 3 (Ill-posed problem) *A problem is **ill-posed** if it is not well-posed.*

Definition 4 (Likelihood function) *Let $\boldsymbol{a}$ be a continuous random variable with probability distribution $p(\boldsymbol{a}, \boldsymbol{\phi})$ depending on the parameter $\boldsymbol{\phi}$; then the function $\mathcal{L}(\boldsymbol{\phi}|\bar{\boldsymbol{a}}) = p(\bar{\boldsymbol{a}}, \boldsymbol{\phi})$ is the likelihood function of $\boldsymbol{\phi}$ given that $\bar{\boldsymbol{a}}$ is the outcome of $\boldsymbol{a}$.*

A.2 Empirical risk minimization

In Section 3.1 we introduced the concept of supervised learning as the branch of ML in which predictive models are trained on labeled data. The final goal of supervised learning is to find a function of the input variables $f : \mathcal{X} \to \mathcal{Y}$ that provides a *good* approximation of the output y. In order to measure the adherence between predictions $\hat{y} = f(\boldsymbol{x})$ and actual output y, we introduced the concept of *loss function* $L(\hat{y}, y)$, see Table 3.1. For a fixed choice of the loss, the ideal estimator, also known as the *target* function, f^* is the minimizer of the (true) expected risk $\mathcal{E}(f)$ *i.e.*

$$f^* = \underset{f \in \mathcal{F}_0}{\operatorname{argmin}} \, \mathcal{E}(f) \tag{A.1}$$

where $\mathcal{F}_0$ is the (huge) class of functions $f : \mathcal{X} \to \mathcal{Y}$ such that $\mathcal{E}(f)$ is finite.

Applying the law of the unconscious statistician, stated in Theorem 1, the expected risk $\mathcal{E}(f)$ can be written as in Equation (A.2), where $(\boldsymbol{x}, y)$ are two random variables with joint probability distribution $p(\boldsymbol{x}, y)$.

$$\mathcal{E}(f) = \mathbb{E}[L(f(\boldsymbol{x}), y)] = \iint_{\mathcal{X} \times \mathcal{Y}} L(f(\boldsymbol{x}), y) \; p(\boldsymbol{x}, y) \; d\boldsymbol{x}dy \tag{A.2}$$

In real situations, a direct computation of $\mathcal{E}(f)$ is unfeasible as the joint probability distribution $p(\boldsymbol{x}, y)$ is unknown. Although, we assume to be provided with a collection of input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ that are supposed to be sampled *i.i.d.* from $\mathcal{X} \times \mathcal{Y}$ according to $p(\boldsymbol{x}, y)$. In statistical learning theory, as introduced by Vapnik [Vapnik, 2013], the dataset $\mathcal{D}$ can be used to build a stochastic approximation of $\mathcal{E}(f)$ called *empirical risk* $\mathcal{E}_{\mathcal{D}}(f)$ and defined in Equation (A.3).

$$\mathcal{E}_{\mathcal{D}}(f) = \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \tag{A.3}$$

As $\mathcal{D}$ is drawn according to the probability distribution $p(\boldsymbol{x}, y)$, our hope is that the empirical risk can be used as a proxy for the expected risk, hence $\mathcal{E}_{\mathcal{D}}(f) \approx \mathcal{E}(f)$. The solution of the supervised learning problem is then found by *Empirical Risk Minimization* (ERM), defined in Equation (A.4)

$$\hat{f}(\boldsymbol{x}) = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \, \mathcal{E}(f_{\mathcal{D}}) = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \, \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \tag{A.4}$$

Where $\mathcal{F}$ is a suitable small subset of $\mathcal{F}_0$. In practice, minimizing $\mathcal{E}(f_{\mathcal{D}})$ instead of $\mathcal{E}(f)$ comes at a price. The central problem is whether the first is a good approximation of the second. For instance, when $p(\boldsymbol{x}, y)$ is too *complex*, the number of examples is too small and/or the class of functions $\mathcal{F}$ is *too large*, $\hat{f}(\boldsymbol{x})$ will be far from the target function $f^*(\boldsymbol{x})$, even when its empirical error is 0. In real circumstances, it is impossible to control the true probability distribution and it is often extremely difficult to collect a very large number of examples. The only element we can control is the class of functions $\mathcal{F}$ and, in particular, its *size*. Since Tikhonov [Tikhonov, 1963] it is known that, for an arbitrary function space $\mathcal{F}$, the ERM problem is ill-posed (see Definition 3). A possible way to ensure well-posedness is to impose a constrain that restricts

the function space. Hence, the constrained ERM problem assumes the form in Equation (A.5), where $\lambda \neq 0$.

$$\begin{aligned} &\underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \\ &\text{subject to } \mathcal{R}(f) < \frac{1}{\lambda} \end{aligned} \tag{A.5}$$

Applying the *Lagrange multipliers technique*[1] Equation (A.5) can be finally written as Equation (A.6).

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) + \lambda \mathcal{R}(f) \tag{A.6}$$

The penalty term $\mathcal{R}(f)$ acts as regularizer [2] and, according to its definition, it can ensure well-posedness of the problem and it can enforce different interesting properties on the achieved solution, see Section 3.1.1. It can also be shown that the use of appropriate regularizes promotes generalization, hence increases our chance to find a $\hat{f}(\boldsymbol{x})$ *close* to $f^*(\boldsymbol{x})$.

According to the choice made for $L(\cdot)$ and $\mathcal{R}(\cdot)$, the minimization problem posed in Equation (A.6) can have very different properties; it can be convex or non-convex, it can include differentiable as well as non-differentiable terms. A rigorous review of the most common optimization methods for ML is beyond the scope of this thesis and can be found here [Boyd and Vandenberghe, 2004; Vito et al., 2005; Bach et al., 2012; Sra et al., 2012; Nesterov, 2013].

A.3 Maximum likelihood estimation

In this section we will see a different approach to tackle the supervised learning problem. Once again, let the training data be made of input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, with $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}, \forall\, i = 1, \ldots, n$. This approach relies on the expression of the uncertainty over the value of y with a probability distribution $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta} \in \Theta$. Applying Definition 4, and assuming that the samples are drawn *i.i.d.*, we can write the likelihood function for $\boldsymbol{\theta}$ as in Equation (A.7).

$$\mathcal{L}(y|\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) \tag{A.7}$$

Equation (A.7) can be considered as the probability of observing the output y_i, given the input $\boldsymbol{x}_i$ and the parameters $\boldsymbol{\theta}$ ($\forall i = 1, \ldots, n$). This statistical setup suggests a strategy to obtain an estimate for $\boldsymbol{\theta}$ known as *Maximum Likelihood Estimation* (MLE), see Equation (A.8).

[1] For a thorough description of this technique, see Appendix E of Bishop's book [Bishop, 2006].

[2] $\mathcal{R}(f)$, in general, can be thought as $\mathcal{R}(f) = \|f\|_K^2$ where $\|\cdot\|_K^2$ is the norm defined by the kernel K in a *Reproducing Kernel Hilbert Space* $\mathcal{H}$ [Evgeniou et al., 2000; Vito et al., 2005].

$$\hat{\boldsymbol{\theta}}_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmax}} \mathcal{L}(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \underset{\theta \in \Theta}{\operatorname{argmax}} \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) \tag{A.8}$$

Instead of maximizing the likelihood it is often convenient to minimize the *negative log-likelihood*[3]. Equation (A.8) can then be rewritten as Equation (A.9).

$$\hat{\boldsymbol{\theta}}_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmin}} - \sum_{i=1}^{n} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) \tag{A.9}$$

Moreover, if some prior knowledge on $\boldsymbol{\theta}$ is available, it is possible to incorporate it in the form of a prior distribution $p(\boldsymbol{\theta})$. Applying the Bayes rule (Theorem 2) it is possible to write the *posterior distribution* $p(\boldsymbol{\theta}|y, \boldsymbol{x})$ as in Equation (A.10).

$$p(\boldsymbol{\theta}|y, \boldsymbol{x}) = \frac{p(y|\boldsymbol{x}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})}{p(y|\boldsymbol{x})} \tag{A.10}$$

The normalizing constant in Equation (A.10) $p(y|\boldsymbol{x})$ is independent from $\boldsymbol{\theta}$. It is known as the marginal likelihood and it can be estimated as in Equation (A.11).

$$p(y|\boldsymbol{x}) = \int p(y|\boldsymbol{x}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}) \, d\boldsymbol{\theta} \tag{A.11}$$

Equation (A.10) suggest a new strategy to achieve an estimate for $\boldsymbol{\theta}$ that takes into account the prior distribution. This criterion is stated in Equation (A.12) and it is known as *Maximum A Posteriori* (MAP).

$$\hat{\boldsymbol{\theta}}_{MAP} = \underset{\theta \in \Theta}{\operatorname{argmax}} \, p(\boldsymbol{\theta}|y, \boldsymbol{x}) = \underset{\theta \in \Theta}{\operatorname{argmax}} \, p(y|\boldsymbol{x}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}) \tag{A.12}$$

Finally, given that $p(y|\boldsymbol{x}, \boldsymbol{\theta})$ is the likelihood of $\boldsymbol{\theta}$ (see Definition 4), we can assume *i.i.d.* samples and apply the negative log-likelihood trick to rewrite Equation (A.12) as in Equation (A.13).

$$\hat{\boldsymbol{\theta}}_{MAP} = \underset{\theta \in \Theta}{\operatorname{argmin}} - \left[\sum_{i=1}^{n} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \right] \tag{A.13}$$

Fixing the two distributions, the predictive model can be achieved solving the minimization problem in Equation (A.13). For the solution of this minimization problem the same observations provided at the end of the last section for Equation (A.6) hold.

A.4 ERM vs MLE/MAP

The goal of this last section is to show that the approaches described in Section A.2 and in Section A.3 look very different, but they actually are two sides of the same coin.

[3] Approaching information theory or deep learning literature, the negative log-likelihood is often referred to as *cross-entropy*.

Assuming that we have the usual collection of *i.i.d.* samples $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} \; \forall \; i = 1, \ldots, n$, our aim here is to learn a good input-output relationship $f : \mathcal{X} \to \mathcal{Y}$. The function f may depend from some parameters that, for ease of writing, will be temporary omitted. If we decide to proceed by MLE, we can find $\hat{f}_{MLE}$ solving the optimization problem in Equation (A.14).

$$\hat{f}_{MLE} = \underset{f \in \mathcal{F}}{\operatorname{argmax}} \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, f) \tag{A.14}$$

Applying the negative log-likelihood trick, Equation (A.14) can be rewritten as Equation (A.15).

$$\hat{f}_{MLE} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} -\frac{1}{n} \sum_{i=1}^{n} \log p(y_i|\boldsymbol{x}_i, f) \tag{A.15}$$

As we can see here, we are naming *negative log-likelihood* what in Section A.2 was called *loss function*. In fact, using $L(f(\boldsymbol{x}), y) = -\log p(y|\boldsymbol{x}, f)$ Equation (A.15) can be rewritten as in Equation (A.16), which is the ERM problem.

$$\hat{f}_{ERM} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) \tag{A.16}$$

In Section A.2 we have seen that introducing $\mathcal{R}(f)$ reduces the space of functions $\mathcal{F}$ and prevents the achieved solution from overfitting. Intuitively, the same effect can be achieved by introducing a prior $p(f)$ as in the MAP estimate. Following Equation (A.13) we can write Equation (A.17).

$$\hat{f}_{MAP} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} -\frac{1}{n} \left[\sum_{i=1}^{n} \log p(y_i|\boldsymbol{x}_i, f) + \lambda \log p(f) \right] \tag{A.17}$$

Finally, as for Equation (A.16), we can express the penalty as $R(f) = -\frac{\lambda}{n} \log p(f)$ and Equation (A.17) becomes Equation (A.18), which is the classical $Loss + Penalty$ formulation of the ERM problem.

$$\hat{f}_{ERM} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i) + \lambda R(f) \tag{A.18}$$

In this section an intuitive explanation of the connection between two popular supervised learning approaches is provided. For a more rigorous overview on ERM and MLE/MAP we refer to [Hastie et al., 2009] and to [Rasmussen and Williams, 2006], respectively.

A.4.1 Linear regression revisited

To clarify the connection between ERM and MLE/MAP we can revisit the simple linear regression problem.

Once again, we have a collection of *i.i.d.* samples $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n = (X, \boldsymbol{y})$, where $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} \; \forall \, i = 1, \dots, n$ and our aim is to learn an input-output relationship $f : \mathcal{X} \to \mathcal{Y}$. Moreover, we assume that the outputs y_i is affected by additive Gaussian noise, hence $y_i = f(\boldsymbol{x}_i) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$. Interestingly, this corresponds to the assumption that f is modeling the mean of the outputs y_i, while its standard deviation σ_n remains unknown, therefore $y_i \sim \mathcal{N}(f(\boldsymbol{x}_i), \sigma_n^2)$ ($\forall \; i = 1, \dots, n$). In Section A.2 we have seen that the ERM solution can be estimated as in Equation A.16. For the sake of simplicity we can restrict to the Ridge Regression case (see Section 3.1.1), *i.e.* we look for a model that can be written as $\hat{y} = f(\boldsymbol{x}) = \boldsymbol{x}^T \hat{\boldsymbol{w}}$ minimizing the square loss $L(\hat{y}, y) = \frac{1}{n} \left\| y - X\boldsymbol{w} \right\|_2^2$ penalized by the ℓ_2-norm $\mathcal{R}(\boldsymbol{w}) = \|w\|_2^2$. Therefore, the minimization problem is stated in Equation (A.19).

$$\hat{\boldsymbol{w}}_{\ell_2} = \underset{\hat{\boldsymbol{w}} \in \mathbb{R}^d}{\operatorname{argmin}} \, J(\boldsymbol{y}, X, \boldsymbol{w}) = \underset{\hat{\boldsymbol{w}} \in \mathbb{R}^d}{\operatorname{argmin}} \, \frac{1}{2n} \left\| \boldsymbol{y} - X\boldsymbol{w} \right\|_2^2 + \frac{\lambda}{2} \left\| \boldsymbol{w} \right\|_2^2 \tag{A.19}$$

In Section A.4 we have seen that the regularized minimization problem corresponds to a MAP estimate with an appropriate choice for negative log-likelihood and prior distribution (on $\boldsymbol{w}$) which correspond to loss function and regularization penalty, respectively. So, considering $J(\boldsymbol{y}, X, \boldsymbol{w})$ as a negative log-posterior and factoring out λ we can write

$$\exp\left[- J(\boldsymbol{y}, X, \boldsymbol{w}) \right] \propto \exp\left[- \frac{1}{2n\lambda} \left\| \boldsymbol{y} - X\boldsymbol{w} \right\|_2^2 \right] \cdot \exp\left[- \frac{1}{2} \left\| w \right\|_2^2 \right]$$

which can be seen as

$$p(\boldsymbol{w} | \boldsymbol{y}, X) = p(\boldsymbol{y} | X, \boldsymbol{w}) \cdot p(\boldsymbol{w})$$

where $p(\boldsymbol{y} | X, \boldsymbol{w}) = \mathcal{N}(X\boldsymbol{w}, n\lambda I)$ and $p(\boldsymbol{w}) = \mathcal{N}(0, I)$. So, in this probabilistic interpretation, the variance of the noise affecting the output $\boldsymbol{y}$ plays the role of the regularization parameter $\lambda \approx \sigma_n^2$.

A.4.2 Logistic regression revisited

In this section we revisit binary classification via logistic regression from a probabilistic perspective.

In binary classification problems we are provided with a collection of input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n = (X, \boldsymbol{y})$, where $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \{+1, -1\}$, $\forall i = 1, \dots, n$. Once again we are looking for a model $f : \mathcal{X} \to \mathcal{Y}$ that associates each input sample with its corresponding class. For the sake of simplicity we restrict to the case of linear functions $\hat{y} = f(\boldsymbol{x}) = \boldsymbol{x}^T \hat{\boldsymbol{w}}$.

The main idea behind logistic regression is to use a loss function having a $[0, 1]$ range to estimate the probability that a sample $\boldsymbol{x}_i$ belongs to one of the two classes. As suggested by its name, the function of choice is the logistic $\sigma(z) = [1 + \exp(-z)]^{-1}$.

In this context, we model our outputs y_i as Bernoulli random variables, which implies that

$$P(y = 1 | \boldsymbol{x}) = \sigma(-f(\boldsymbol{x})) = \frac{1}{1 + \exp(-f(\boldsymbol{x}))}$$

and

$$P(y = -1 | \boldsymbol{x}) = \sigma(f(\boldsymbol{x})) = \frac{1}{1 + \exp(f(\boldsymbol{x}))}$$

consequently, we can write the general form as in Equation (A.20).

$$P(y = \pm 1|\boldsymbol{x}) = \frac{1}{1 + \exp\left(-yf(\boldsymbol{x})\right)} \tag{A.20}$$

Therefore, assuming a Gaussian prior on the weights $\boldsymbol{w} \sim \mathcal{N}(0, I)$, we can perform a MAP estimate of $\hat{f}$ solving the problem in Equation (A.21).

$$\hat{\boldsymbol{w}}_{\text{LR}} = \underset{\hat{\boldsymbol{w}} \in \mathbb{R}^d}{\operatorname{argmax}} \prod_{i=1}^{n} p(\boldsymbol{w}|y_i, \boldsymbol{x}_i) = \underset{\hat{\boldsymbol{w}} \in \mathbb{R}^d}{\operatorname{argmax}} \prod_{i=1}^{n} \frac{1}{1 + \exp\left(-y_i\, \boldsymbol{x}_i^T \boldsymbol{w}\right)} \cdot \exp\left(-\frac{1}{2} \left\|\boldsymbol{w}\right\|_2^2\right) \tag{A.21}$$

Log-transforming Equation (A.21), and applying some elementary linear algebra, we can write Equation (A.22).

$$\hat{\boldsymbol{w}}_{\text{LR}} = \underset{\hat{\boldsymbol{w}} \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2n\lambda} \sum_{i=1}^{n} \log\left[1 + \exp\left(-y_i \boldsymbol{x}_i^T \boldsymbol{w}\right)\right] + \frac{\lambda}{2} \left\|\boldsymbol{w}\right\|_2^2 \tag{A.22}$$

The minimization problem expressed in Equation (A.22) is known as Regularized Logistic Regression. It can be casted in the regularization framework of Equation (A.18) where the logistic loss function $L(f(\boldsymbol{x}), y) = \log\left[1 + \exp\left(-yf(\boldsymbol{x})\right)\right]$ is penalized by the ℓ_2-norm.

List of Figures

List of Tables

List of Abbreviations

AIC Akaike Information Crieria

AMI Adjusted Mutual Information

API Application Program Interface

ARI Adjusted Rand Index

ARIMA Autoregressive Integrated Moving Average

ARMA Autoregressive Moving Average

AUC Area Under the Curve

B Benign Multiple Sclerosis

BIC Bayesian Information Crieria

BLAS Basic Linear Algebra Subprograms

CART Classification And Regression Trees

CE Cross-Entropy

CGM Continuous Glucose Monitoring

CNN Convolutional Neural Network

CPU Central Processing Unit

CT Computerized Tomography

DNA DeoxyriboNucleic Acid

DSL Domain Specific Language

EDA Exploratory-Data Analysis

ERM Empirical Risk Minimization

EV Explained Variance

FISTA Fast Iterative Shrinkage-Thresholding Algorithm

FPR False Positive Rate

GB Gradient Boosting

GPU Graphics Processing Unit

GUI Graphical User Interface

GWAS Genome-wide Association Studies

HBGI High Blood Glucose Index

i.i.d. Independent and Identically Distributed

KF Kalman Filter

KRR Kernel Ridge Regression

LBGI Low Blood Glucose Index

LDH Lactate dehydrogenase

LSTM Long-Short Term Memory Network

MAE Mean Absolute Error

MAP Maximum A Posteriori

MCC Matthews correlation coefficient

MDA Malondialdehyde

MDS Multi-Dimensional Scaling

MKL Math Kernel Library

ML Machine Learning

MLE Maximum Likelihood Estimation

MLP Multi-Layer Perceptron

MRI Magnetic Resonance Imaging

MSE Mean Squared Error

MTEN Multi-task Elastic-Net

NNM Nuclear Norm Minimization

OLS Ordinary Least Squares

OVA *One-vs-All*

OVO *One-vs-One*

PCA Principal Components Analysis

PCO Patient Centered Outcomes

PET Positron Emission Tomography

PP Primary-Progressive Multiple Sclerosis

PR Progressive-Relapsing Multiple Sclerosis

$\mathbf{R}^2$ Coefficient of determination

RF Random Forests

RMSE Root Mean Squared Error

RNN Recurrent Neural Network

ROC Receiver Operating Curve

RR Relapsing-Remitting Multiple Sclerosis

RSS Residual Sum of Squares

SP Secondary-Progressive Multiple Sclerosis

SPECT Single-Photon Emission Computed Tomography

SVC Support Vector Classification

SVM Support Vector Machine

SVR Support Vector Regression

TPR True Positive Rate

Bibliography

Aben, N., Vis, D. J., Michaut, M., and Wessels, L. F. (2016). Tandem: a two-stage approach to maximize interpretability of drug response models based on multiple molecular data types. *Bioinformatics*, 32(17):i413–i420. [Not cited.]

Abraham, G., Kowalczyk, A., Zobel, J., and Inouye, M. (2013). Performance and robustness of penalized and unpenalized methods for genetic prediction of complex human disease. *Genetic Epidemiology*, 37(2):184–195. [Not cited.]

Alexandrov, L. B., Nik-Zainal, S., Wedge, D. C., Aparicio, S. A., Behjati, S., Biankin, A. V., Bignell, G. R., Bolli, N., Borg, A., Børresen-Dale, A.-L., et al. (2013). Signatures of mutational processes in human cancer. *Nature*, 500(7463):415–421. [Not cited.]

Altmann, A., Toloşi, L., Sander, O., and Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347. [Not cited.]

Ambroise, C. and McLachlan, G. J. (2002). Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the national academy of sciences*, 99(10):6562–6566. [Not cited.]

Ancona, N., Maglietta, R., D'Addabbo, A., Liuni, S., and Pesole, G. (2005). Regularized least squares cancer classifiers from dna microarray data. *BMC bioinformatics*, 6(4):S2. [Not cited.]

Angermueller, C., Pärnamaa, T., Parts, L., and Stegle, O. (2016). Deep learning for computational biology. *Molecular systems biology*, 12(7):878. [Not cited.]

Appenzeller, T. (2017). The ai revolution in science. *Science*. [Not cited.]

Argyriou, A., Evgeniou, T., and Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, 73(3):243–272. [Not cited.]

Arnould, C., Vandervelde, L., Batcho, C. S., Penta, M., and Thonnard, J.-L. (2012). Can manual ability be measured with a generic abilhand scale? a cross-sectional study conducted on six diagnostic groups. *BMJ open*, 2(6):e001807. [Not cited.]

Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics. [Not cited.]

Aupperle, R. L., Beatty, W. W., Shelton, F. d. N., and Gontkovsky, S. T. (2002). Three screening batteries to detect cognitive impairment in multiple sclerosis. *Multiple Sclerosis*, 8(5):382–389. [Not cited.]

Bach, F., Jenatton, R., Mairal, J., Obozinski, G., et al. (2012). Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106. [Not cited.]

Baldassarre, L., Rosasco, L., Barla, A., and Verri, A. (2012). Multi-output learning via spectral filtering. *Machine learning*, 87(3):259–301. [Not cited.]

Ball, G. H. and Hall, D. J. (1967). A clustering technique for summarizing multivariate data. *Systems Research and Behavioral Science*, 12(2):153–155. [Not cited.]

Barbieri, M., Fiorini, S., Tomasi, F., and Barla, A. (2016). PALLADIO: a parallel framework for robust variable selection in high-dimensional data. *PyHPC2016 conference, IEEE proceedings*. [Not cited.]

Barrett, T., Wilhite, S. E., Ledoux, P., Evangelista, C., Kim, I. F., Tomashevsky, M., Marshall, K. A., Phillippy, K. H., Sherman, P. M., Holko, M., et al. (2013). Ncbi geo: archive for functional genomics data sets—update. *Nucleic acids research*, 41(D1):D991–D995. [Not cited.]

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202. [Not cited.]

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* [Not cited.]

Bergamaschi, R., Montomoli, C., Mallucci, G., Lugaresi, A., Izquierdo, G., Grand'Maison, F., Duquette, P., Shaygannejad, V., Alroughani, R., Grammond, P., et al. (2015). Bremso: A simple score to predict early the natural course of multiple sclerosis. *European journal of neurology*, 22(6):981–989. [Not cited.]

Bishop, C. M. (2006). Pattern recognition. *Machine Learning*. [Not cited.]

Black, N. (2013). Patient reported outcome measures could help transform healthcare. *BMJ (Clinical research ed)*, 346:f167. [Not cited.]

Borg, I. and Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media. [Not cited.]

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons. [Not cited.]

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press. [Not cited.]

Bratic, I. and Trifunovic, A. (2010). Mitochondrial energy metabolism and ageing. *Biochimica et Biophysica Acta (BBA)-Bioenergetics*, 1797(6):961–967. [Not cited.]

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32. [Not cited.]

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press. [Not cited.]

Brichetto, G., Fiorini, S., Ponzio, M., Barla, A., Verri, A., and Tacchino, A. (2016). Predicting multiple sclerosis disease course with patient centred outcomes (pcos): a machine learning approach. In *Multiple sclerosis journal*, volume 22, pages 698–698. Sage publications LTD 1 Olivers Yard, 55 City Road, London EC1Y 1sp, England. [Not cited.]

Brichetto, G., Fiorini, S., Tacchino, A., Ponzio, M., Barla, A., and Verri, A. (2015). Improving disease course detection in multiple sclerosis: an alternative patient-reported outcomes-based strategy. In *Multiple sclerosis journal*, volume 21, pages 513–514. Sage publications LTD 1 Olivers Yard, 55 City Road, London EC1Y 1sp, England. [Not cited.]

Buehlmann, P. (2006). Boosting for high-dimensional linear models. *The Annals of Statistics*, pages 559–583. [Not cited.]

Bunescu, R., Struble, N., Marling, C., Shubrook, J., and Schwartz, F. (2013). Blood glucose level prediction using physiological models and support vector regression. In *Proc. of IEEE ICMLA*. [Not cited.]

Cadenas, E. and Davies, K. J. (2000). Mitochondrial free radical generation, oxidative stress, and aging. *Free Radical Biology and Medicine*, 29(3):222–230. [Not cited.]

Cai, T. T. and Wang, L. (2011). Orthogonal matching pursuit for sparse signal recovery with noise. *Information Theory, IEEE Transactions on*, 57(7):4680–4688. [Not cited.]

Campisi, J. (2013). Aging, cellular senescence, and cancer. *Annual review of physiology*, 75:685–705. [Not cited.]

Candès, E. J. and Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717. [Not cited.]

Cappelli, E., Cuccarolo, P., Stroppiana, G., Miano, M., Bottega, R., Cossu, V., Degan, P., and Ravera, S. (2017). Defects in mitochondrial energetic function compels fanconi anaemia cells to glycolytic metabolism. *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, 1863(6):1214–1221. [Not cited.]

Cardozo, L., Staskin, D., Currie, B., Wiklund, I., Globe, D., Signori, M., Dmochowski, R., MacDiarmid, S., Nitti, V. W., and Noblett, K. (2014). Validation of a bladder symptom screening tool in women with incontinence due to overactive bladder. *International urogynecology journal*, 25(12):1655–1663. [Not cited.]

Chen, X., He, J., Lawrence, R., and Carbonell, J. G. (2012). Adaptive multi-task sparse learning with an application to fmri study. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 212–223. SIAM. [Not cited.]

Chen, Y., Li, Y., Narayan, R., Subramanian, A., and Xie, X. (2016). Gene expression inference with deep learning. *Bioinformatics*, 32 12:1832–9. [Not cited.]

Chollet, F. (2018). *Deep learning with Python*. Manning Publications. [Not cited.]

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619. [Not cited.]

Dagenais, E., Rouleau, I., Demers, M., Jobin, C., Roger, É., Chamelian, L., and Duquette, P. (2013). Value of the moca test as a screening instrument in multiple sclerosis. *The Canadian Journal of Neurological Sciences*, 40(03):410–415. [Not cited.]

Dai, D.-F., Marcinek, D. J., Szeto, H. H., Rabinovitch, P. S., and Chiao, Y. A. (2014). Mitochondrial oxidative stress in aging and healthspan. *Longevity & healthspan*, 3(1):6. [Not cited.]

De Mol, C., De Vito, E., and Rosasco, L. (2009a). Elastic-net regularization in learning theory. *Journal of Complexity*, 25(2):201–230. [Not cited.]

De Mol, C., Mosci, S., Traskine, M., and Verri, A. (2009b). A regularized method for selecting nested groups of relevant genes from microarray data. *Journal of Computational Biology*, 16(5):677–690. [Not cited.]

De Souto, M. C., Jaskowiak, P. A., and Costa, I. G. (2015). Impact of missing data imputation methods on gene expression clustering and classification. *BMC bioinformatics*, 16(1):64. [Not cited.]

Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., et al. (2013). Orange: data mining toolbox in python. *The Journal of Machine Learning Research*, 14(1):2349–2353. [Not cited.]

Deng, H. and Runger, G. (2013). Gene selection with guided regularized random forest. *Pattern Recognition*, 46(12):3483–3489. [Not cited.]

Dickens, A. M., Larkin, J. R., Griffin, J. L., Cavey, A., Matthews, L., Turner, M. R., Wilcock, G. K., Davis, B. G., Claridge, T. D., Palace, J., et al. (2014). A type 2 biomarker separates relapsing-remitting from secondary progressive multiple sclerosis. *Neurology*, 83(17):1492–1499. [Not cited.]

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231. [Not cited.]

Everitt, B. and Skrondal, A. (2002). *The Cambridge dictionary of statistics*, volume 106. Cambridge University Press Cambridge. [Not cited.]

Everitt, B. S. (2006). *The Cambridge dictionary of statistics*. Cambridge University Press. [Not cited.]

Evgeniou, T., Pontil, M., and Poggio, T. (2000). Regularization networks and support vector machines. *Advances in computational mathematics*, 13(1):1–50. [Not cited.]

Fabris, C., Patek, S. D., and Breton, M. D. (2016). Are risk indices derived from cgm interchangeable with smbg-based indices? *Journal of diabetes science and technology*. [Not cited.]

Facchinetti, A., Sparacino, G., and Cobelli, C. (2010). An online self-tunable method to denoise cgm sensor data. *IEEE Trans. Biomed. Eng.* [Not cited.]

Fiorini, S., Martini, C., Malpassi, D., Cordera, R., Maggi, D., Verri, A., and Barla, A. (2017a). Data-driven strategies for robust forecast of continuous glucose monitoring time-series. In *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE*, pages 1680–1683. IEEE. [Not cited.]

Fiorini, S., Tacchino, A., Brichetto, G., Verri, A., and Barla, A. (2016). A temporal model for multiple sclerosis course evolution. *arXiv preprint arXiv:1612.00615*. [Not cited.]

Fiorini, S., Tomasi, F., Squillario, M., and Barla, A. (2017b). Adenine: a hpc-oriented tool for biological data exploration. In *Computational Intelligence methods for Bioinformatics and Biostatistics, CIBB*. [Not cited.]

Fiorini, S., Verri, A., Barla, A., Tacchino, A., and Brichetto, G. (2017c). Temporal prediction of multiple sclerosis evolution from patient-centered outcomes. In Doshi-Velez, F., Fackler, J., Kale, D., Ranganath, R., Wallace, B., and Wiens, J., editors, *Proceedings of the 2nd Machine Learning for Healthcare Conference*, volume 68 of *Proceedings of Machine Learning Research*, pages 112–125, Boston, Massachusetts. PMLR. [Not cited.]

Fiorini, S., Verri, A., Tacchino, A., Ponzio, M., Brichetto, G., and Barla, A. (2015). A machine learning pipeline for multiple sclerosis course detection from clinical scales and patient reported outcomes. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pages 4443–4446. IEEE. [Not cited.]

Flachenecker, P., Kümpfel, T., Kallmann, B., Gottschalk, M., Grauer, O., Rieckmann, P., Trenkwalder, C., and Toyka, K. (2002). Fatigue in multiple sclerosis: a comparison of different rating scales and correlation to clinical parameters. *Multiple sclerosis*, 8(6):523–526. [Not cited.]

Franchignoni, F., Tesio, L., Ottonello, M., and Benevolo, E. (1999). Life satisfaction index: Italian version and validation of a short form1. *American journal of physical medicine & rehabilitation*, 78(6):509–515. [Not cited.]

Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *science*, 315(5814):972–976. [Not cited.]

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232. [Not cited.]

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378. [Not cited.]

Genova, M. L., Pich, M. M., Bernacchia, A., Bianchi, C., Biondi, A., Bovina, C., Falasca, A. I., Formiggini, G., Castelli, G. P., and Lenaz, G. (2004). The mitochondrial production of reactive oxygen species in relation to aging and pathology. *Annals of the New York Academy of Sciences*, 1011(1):86–100. [Not cited.]

Giovannoni, G., Butzkueven, H., Dhib-Jalbut, S., Hobart, J., Kobelt, G., Pepper, G., Sormani, M. P., Thalheim, C., Traboulsee, A., and Vollmer, T. (2016). Brain health: time matters in multiple sclerosis. *Multiple Sclerosis and Related Disorders*, 9:S5–S48. [Not cited.]

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press. [Not cited.]

Gramfort, A., Kowalski, M., and Hämäläinen, M. (2012). Mixed-norm estimates for the m/eeg inverse problem using accelerated gradient methods. *Physics in medicine and biology*, 57(7):1937. [Not cited.]

Granger, C., Cotter, A., Hamilton, B., Fiedler, R., and Hens, M. (1990). Functional assessment scales: a study of persons with multiple sclerosis. *Archives of physical medicine and rehabilitation*, 71(11):870–875. [Not cited.]

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*. [Not cited.]

Guariguata, L., Whiting, D., Hambleton, I., Beagley, J., Linnenkamp, U., and Shaw, J. (2014). Global estimates of diabetes prevalence for 2013 and projections for 2035. *Diabetes research and clinical practice*. [Not cited.]

Gui, J. and Li, H. (2005). Penalized cox regression analysis in the high-dimensional and low-sample size settings, with applications to microarray gene expression data. *Bioinformatics*, 21(13):3001–3008. [Not cited.]

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182. [Not cited.]

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422. [Not cited.]

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288. [Not cited.]

Harman, D. (1972). The biologic clock: the mitochondria? *Journal of the American Geriatrics Society*, 20(4):145–147. [Not cited.]

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning*, volume 2. Springer. [Not cited.]

Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical learning with sparsity: the lasso and generalizations*. CRC Press. [Not cited.]

He, D., Kuhn, D., and Parida, L. (2016). Novel applications of multitask learning and multiple output regression to multiple genetic trait prediction. *Bioinformatics*, 32(12):i37–i43. [Not cited.]

Helmstaedter, M., Briggman, K. L., Turaga, S. C., Jain, V., Seung, H. S., and Denk, W. (2013). Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168–174. [Not cited.]

Hinkle, P. C. (2005). P/o ratios of mitochondrial oxidative phosphorylation. *Biochimica et Biophysica Acta (BBA)-Bioenergetics*, 1706(1):1–11. [Not cited.]

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67. [Not cited.]

Hofner, B., Boccuto, L., and Göker, M. (2015). Controlling false discoveries in high-dimensional situations: boosting with stability selection. *BMC bioinformatics*, 16(1):144. [Not cited.]

Hoggart, C. J., Whittaker, J. C., De Iorio, M., and Balding, D. J. (2008). Simultaneous analysis of all snps in genome-wide and re-sequencing association studies. *PLoS genetics*, 4(7):e1000130. [Not cited.]

Honarmand, K. and Feinstein, A. (2009). Validation of the hospital anxiety and depression scale for use with multiple sclerosis patients. *Multiple Sclerosis*. [Not cited.]

Horvath, S. (2013). Dna methylation age of human tissues and cell types. *Genome biology*, 14(10):3156. [Not cited.]

Hughey, J. J. and Butte, A. J. (2015). Robust meta-analysis of gene expression using the elastic net. *Nucleic Acids Research*, 43(12):e79. [Not cited.]

Ishwaran, H., Kogalur, U. B., Blackstone, E. H., and Lauer, M. S. (2008). Random Survival Forests. *The Annals of Applied Statistics*, 2(3):841–860. [Not cited.]

Jacob, L., Obozinski, G., and Vert, J.-P. (2009). Group lasso with overlap and graph lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 433–440, New York, NY, USA. ACM. [Not cited.]

Jaiswal, S., Fontanillas, P., Flannick, J., Manning, A., Grauman, P. V., Mar, B. G., Lindsley, R. C., Mermel, C. H., Burtt, N., Chavez, A., et al. (2014). Age-related clonal hematopoiesis associated with adverse outcomes. *New England Journal of Medicine*, 371(26):2488–2498. [Not cited.]

Jolliffe, I. (2002). *Principal component analysis*. Wiley Online Library. [Not cited.]

Joly, A., Schnitzler, F., Geurts, P., and Wehenkel, L. (2012). L1-based compression of random forest models. In *20th European Symposium on Artificial Neural Networks*. [Not cited.]

Jung, K., Dihazi, H., Bibi, A., Dihazi, G. H., and Beißbarth, T. (2014). Adaption of the global test idea to proteomics data with missing values. *Bioinformatics*, 30(10):1424–1430. [Not cited.]

Keener, R. W. (2011). *Theoretical statistics: Topics for a core course*. Springer. [Not cited.]

Knobbe, E. J. and Buckingham, B. (2005). The extended kalman filter for continuous glucose monitoring. *Diabetes technology & therapeutics*. [Not cited.]

Kovatchev, B. P., Cox, D. J., Gonder-Frederick, L. A., and Clarke, W. (1997). Symmetrization of the blood glucose measurement scale and its applications. *Diabetes Care*. [Not cited.]

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. [Not cited.]

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957. [Not cited.]

Kulkarni, V. Y. and Sinha, P. K. (2012). Pruning of Random Forest classifiers: A survey and future directions. In *2012 International Conference on Data Science Engineering (ICDSE)*, pages 64–68. [Not cited.]

Kuncheva, L. I. (2007). A Stability Index for Feature Selection. In *Proceedings of the 25th Conference on Proceedings of the 25th IASTED International Multi-Conference: Artificial Intelligence and Applications*, AIAP'07, Anaheim, CA, USA. ACTA Press. [Not cited.]

Kursa, M. B. (2014). Robustness of Random Forest-based gene selection methods. *BMC Bioinformatics*, 15:8. [Not cited.]

Kurtzke, J. F. (1983). Rating neurologic impairment in multiple sclerosis an expanded disability status scale (edss). *Neurology*, 33(11):1444–1444. [Not cited.]

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444. [Not cited.]

LeCun, Y., Cortes, C., and Burges, C. J. (2010). Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2. [Not cited.]

Lee, S., Zhu, J., and Xing, E. P. (2010). Adaptive multi-task lasso: with application to eqtl detection. In *Advances in neural information processing systems*, pages 1306–1314. [Not cited.]

Leung, M. K., Xiong, H. Y., Lee, L. J., and Frey, B. J. (2014). Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129. [Not cited.]

Lewis, J. M., De Sa, V. R., and Van Der Maaten, L. (2013). Divvy: fast and intuitive exploratory data analysis. *The Journal of Machine Learning Research*, 14(1):3159–3163. [Not cited.]

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 413–422. IEEE. [Not cited.]

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2012). Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):3. [Not cited.]

Liu, S., Dissanayake, S., Patel, S., Dang, X., Mlsna, T., Chen, Y., and Wilkins, D. (2014). Learning accurate and interpretable models based on regularized random forests regression. *BMC Systems Biology*, 8(3):S5. [Not cited.]

López-Otín, C., Blasco, M. A., Partridge, L., Serrano, M., and Kroemer, G. (2013). The hallmarks of aging. *Cell*, 153(6):1194–1217. [Not cited.]

Lublin, F. D., Reingold, S. C., Cohen, J. A., Cutter, G. R., Sørensen, P. S., Thompson, A. J., Wolinsky, J. S., Balcer, L. J., Banwell, B., Barkhof, F., et al. (2014). Defining the clinical course of multiple sclerosis the 2013 revisions. *Neurology*, 83(3):278–286. [Not cited.]

Lusa, L. et al. (2015). Boosting for high-dimensional two-class prediction. *BMC bioinformatics*, 16(1):300. [Not cited.]

Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E., and Svetnik, V. (2015). Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274. [Not cited.]

Ma, S. and Huang, J. (2007). Additive risk survival model with microarray data. *BMC bioinformatics*, 8(1):192. [Not cited.]

Mamoshina, P., Vieira, A., Putin, E., and Zhavoronkov, A. (2016). Applications of deep learning in biomedicine. *Molecular pharmaceutics*, 13(5):1445–1454. [Not cited.]

Marx, V. (2013). Biology: The big challenges of big data. *Nature*, 498(7453):255–260. [Not cited.]

Masecchia, S., Coco, S., Barla, A., Verri, A., and Tonini, G. P. (2015). Genome iny model of metastatic neuroblastoma tumorigenesis by a dictionary learning algorithm. *BMC medical genomics*, 8(1):57. [Not cited.]

Mayr, A., Binder, H., Gefeller, O., Schmid, M., et al. (2014). The evolution of boosting algorithms. *Methods of Information in Medicine*, 53(6):419–427. [Not cited.]

McKerrell, T., Park, N., Moreno, T., Grove, C. S., Ponstingl, H., Stephens, J., Crawley, C., Craig, J., Scott, M. A., Hodkinson, C., et al. (2015). Leukemia-associated somatic mutations drive distinct patterns of age-related clonal hemopoiesis. *Cell reports*, 10(8):1239–1245. [Not cited.]

Meier, L., Van De Geer, S., and Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71. [Not cited.]

Meinshausen, N. and Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473. [Not cited.]

Meyer-Moock, S., Feng, Y.-S., Maeurer, M., Dippel, F.-W., and Kohlmann, T. (2014). Systematic literature review and validity evaluation of the expanded disability status scale (edss) and the multiple sclerosis functional composite (msfc) in patients with multiple sclerosis. *BMC neurology*, 14(1):58. [Not cited.]

Min, S., Lee, B., and Yoon, S. (2016). Deep learning in bioinformatics. *arXiv preprint arXiv:1603.06430*. [Not cited.]

Molinaro, A. M., Simon, R., and Pfeiffer, R. M. (2005). Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307. [Not cited.]

Muhr, M. and Granitzer, M. (2009). Automatic cluster number selection using a split and merge k-means approach. In *Database and Expert Systems Application, 2009. DEXA'09. 20th International Workshop on*, pages 363–367. IEEE. [Not cited.]

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press. [Not cited.]

Nelson, E. C., Eftimovska, E., Lind, C., Hager, A., Wasson, J. H., and Lindblad, S. (2015). Patient reported outcome measures in practice. *Bmj*, 350:g7818. [Not cited.]

Nesterov, Y. (2013). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media. [Not cited.]

Ng, A. Y., Jordan, M. I., Weiss, Y., et al. (2002). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856. [Not cited.]

Nogueira, S. and Brown, G. (2016). Measuring the stability of feature selection. In Frasconi, P., Landwehr, N., Manco, G., and Vreeken, J., editors, *Machine Learning and Knowledge Discovery in Databases*, number 9852 in Lecture Notes in Computer Science. Springer International Publishing. [Not cited.]

Nowak, G., Hastie, T., Pollack, J. R., and Tibshirani, R. (2011). A fused lasso latent feature model for analyzing multi-sample acgh data. *Biostatistics*, page kxr012. [Not cited.]

Okser, S., Pahikkala, T., Airola, A., Salakoski, T., Ripatti, S., and Aittokallio, T. (2014). Regularized machine learning in the genetic prediction of complex traits. *PLoS Genet*, 10(11):e1004754. [Not cited.]

Oldfield, R. C. (1971). The assessment and analysis of handedness: the edinburgh inventory. *Neuropsychologia*, 9(1):97–113. [Not cited.]

Organization, W. H. et al. (2016). Global report on diabetes. geneva: World health organization; 2016. [Not cited.]

Parmenter, B., Weinstock-Guttman, B., Garg, N., Munschauer, F., and Benedict, R. H. (2007). Screening for cognitive impairment in multiple sclerosis using the symbol digit modalities test. *Multiple Sclerosis*, 13(1):52–57. [Not cited.]

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. [Not cited.]

Pelleg, D., Moore, A. W., et al. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, volume 1, pages 727–734. [Not cited.]

Peng, B., Wang, L., and Wu, Y. (2016). An error bound for l1-norm support vector machine coefficients in ultra-high dimension. *The Journal of Machine Learning Research*, 17(1):8279–8304. [Not cited.]

Prechelt, L. (1998). Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer. [Not cited.]

Qi, Y. (2012). Random forest for bioinformatics. In *Ensemble machine learning*, pages 307–323. Springer. [Not cited.]

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, volume 1. MIT press Cambridge. [Not cited.]

Ravera, S., Bartolucci, M., Calzia, D., Aluigi, M. G., Ramoino, P., Morelli, A., and Panfoli, I. (2013). Tricarboxylic acid cycle-sustained oxidative phosphorylation in isolated myelin vesicles. *Biochimie*, 95(11):1991–1998. [Not cited.]

Ravera, S., Bartolucci, M., Cuccarolo, P., Litamè, E., Illarcio, M., Calzia, D., Degan, P., Morelli, A., and Panfoli, I. (2015). Oxidative stress in myelin sheath: The other face of the extramitochondrial oxidative phosphorylation ability. *Free radical research*, 49(9):1156–1164. [Not cited.]

Riera, C. E., Merkwirth, C., De Magalhaes Filho, C. D., and Dillin, A. (2016). Signaling networks determining life span. *Annual review of biochemistry*, 85:35–64. [Not cited.]

Ross, D. A., Lim, J., Lin, R.-S., and Yang, M.-H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141. [Not cited.]

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65. [Not cited.]

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326. [Not cited.]

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*. [Not cited.]

Sastre, J., Pallardó, F. V., and Vina, J. (2000). Mitochondrial oxidative stress plays a key role in aging and apoptosis. *IUBMB life*, 49(5):427–435. [Not cited.]

Scalfari, A., Neuhaus, A., Daumer, M., Muraro, P. A., and Ebers, G. C. (2014). Onset of secondary progressive phase and long-term evolution of multiple sclerosis. *Journal of Neurology, Neurosurgery & Psychiatry*, 85(1):67–75. [Not cited.]

Schmidhuber, J., Wierstra, D., and Gomez, F. (2005). Evolino: Hybrid neuroevolution/optimal linear search for sequence learning. In *Proc. of IJCAI*. [Not cited.]

Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel principal component analysis. In *Artificial Neural Networks—ICANN'97*, pages 583–588. Springer. [Not cited.]

Schulz, W. (2005). *Molecular biology of human cancers: an advanced student's textbook*. Springer Science & Business Media. [Not cited.]

Service, R. F. (2017). Ai is changing how we do science. get a glimpse. *Science*. [Not cited.]

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press. [Not cited.]

Shawe-Taylor, J. and Sun, S. (2011). A review of optimization methodologies in support vector machines. *Neurocomputing*, 74(17):3609–3618. [Not cited.]

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905. [Not cited.]

Short, K. R., Bigelow, M. L., Kahl, J., Singh, R., Coenen-Schimke, J., Raghavakaimal, S., and Nair, K. S. (2005). Decline in skeletal muscle mitochondrial function with aging in humans. *Proceedings of the National Academy of Sciences of the United States of America*, 102(15):5618–5623. [Not cited.]

Smola, A. J. and Schölkopf, B. (1998). *Learning with kernels*. GMD-Forschungszentrum Informationstechnik. [Not cited.]

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222. [Not cited.]

Sparacino, G., Zanderigo, F., Corazza, S., Maran, A., Facchinetti, A., and Cobelli, C. (2007). Glucose concentration can be predicted ahead in time from continuous glucose monitoring sensor time-series. *IEEE Transactions on biomedical engineering*. [Not cited.]

Sra, S., Nowozin, S., and Wright, S. J. (2012). *Optimization for machine learning*. Mit Press. [Not cited.]

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958. [Not cited.]

Stekhoven, D. J. and Bühlmann, P. (2011). Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118. [Not cited.]

Tacchino, A., Fiorini, S., Ponzio, M., Barla, A., Verri, A., Battaglia, M., and Brichetto, G. (2017). Multiple sclerosis disease course prediction: a machine learning model based on patient reported and clinician assessed outcomes. In *Multiple sclerosis journal*, volume 23, pages 58–59. Sage publications LTD 1 Olivers Yard, 55 City Road, London EC1Y 1sp, England. [Not cited.]

Tang, Z., Shen, Y., Zhang, X., and Yi, N. (2017). The spike-and-slab lasso cox model for survival prediction and associated genes detection. *Bioinformatics*, page btx300. [Not cited.]

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. *International journal of forecasting*. [Not cited.]

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323. [Not cited.]

Thompson, M. J. et al. (2017). An epigenetic aging clock for dogs and wolves. *Aging (Albany NY)*, 9(3):1055. [Not cited.]

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288. [Not cited.]

Tibshirani, R. et al. (1997). The lasso method for variable selection in the cox model. *Statistics in medicine*, 16(4):385–395. [Not cited.]

Tikhonov, A. (1963). Solution of incorrectly formulated problems and the regularization method. In *Soviet Math. Dokl.*, volume 5, pages 1035–1038. [Not cited.]

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., and Altman, R. B. (2001). Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525. [Not cited.]

Turrens, J. F. (2003). Mitochondrial formation of reactive oxygen species. *The Journal of physiology*, 552(2):335–344. [Not cited.]

Tutz, G. and Binder, H. (2006). Generalized additive modeling with implicit variable selection by likelihood-based boosting. *Biometrics*, 62(4):961–971. [Not cited.]

Tutz, G. and Binder, H. (2007). Boosting ridge regression. *Computational Statistics & Data Analysis*, 51(12):6044–6059. [Not cited.]

Uitdehaag, B. (2014). Clinical outcome measures in multiple sclerosis. *Handb Clin Neurol*, 122:393–404. [Not cited.]

van Beers, C. A. and DeVries, J. H. (2016). Continuous glucose monitoring impact on hypoglycemia. *J Diabetes Sci Technol*. [Not cited.]

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85. [Not cited.]

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media. [Not cited.]

Vigersky, R. and Shrivastav, M. (2017). Role of continuous glucose monitoring for type 2 in diabetes management and research. *Journal of Diabetes and its Complications*. [Not cited.]

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [Not cited.]

Vito, E. D., Rosasco, L., Caponnetto, A., Giovannini, U. D., and Odone, F. (2005). Learning from examples as an inverse problem. *Journal of Machine Learning Research*, 6(May):883–904. [Not cited.]

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416. [Not cited.]

Vukusic, S. and Confavreux, C. (2003). Prognostic factors for progression of disability in the secondary progressive phase of multiple sclerosis. *Journal of the neurological sciences*, 206(2):135–137. [Not cited.]

Waldmann, P., Mészáros, G., Gredler, B., Fuerst, C., and Sölkner, J. (2013). Evaluation of the lasso and the elastic net in genome-wide association studies. *Frontiers in genetics*, 4:270. [Not cited.]

Wallace, D. C. (2010). Mitochondrial dna mutations in disease and aging. *Environmental and molecular mutagenesis*, 51(5):440–450. [Not cited.]

Welch, G. and Bishop, G. (1995). An introduction to the kalman filter. [Not cited.]

Witten, D. M. and Tibshirani, R. (2009). Covariance-regularized regression and classification for high dimensional problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):615–636. [Not cited.]

Wu, T. T., Chen, Y. F., Hastie, T., Sobel, E., and Lange, K. (2009). Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics*, 25(6):714–721. [Not cited.]

Wu, T. T. and Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, pages 224–244. [Not cited.]

Yuan, Y., Shi, Y., Li, C., Kim, J., Cai, T. W., Han, Z., and Feng, D. D. (2016). Deepgene: an advanced cancer type classifier based on deep learning and somatic point mutations. In *BMC Bioinformatics*. [Not cited.]

Zecchin, C., Facchinetti, A., Sparacino, G., De Nicolao, G., and Cobelli, C. (2011). A new neural network approach for short-term glucose prediction using continuous glucose monitoring time-series and meal information. In *Proc. of IEEE EMBC*. [Not cited.]

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC press. [Not cited.]

Zhu, J., Rosset, S., Tibshirani, R., and Hastie, T. J. (2004). 1-norm support vector machines. In *Advances in neural information processing systems*, pages 49–56. [Not cited.]

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429. [Not cited.]

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320. [Not cited.]