University of Padova

Department of Information Engineering

Machine learning techniques for classification problems related to therapies in diabetes patients

Author:
Elena Barutta

Supervisor:
Prof. Fabio Vandin

A.Y. 2018/2019

A thesis submitted for the Master degree in

ICT for Internet and Multimedia

December 9, 2019

Acknowledgements

I would like to do a special thanks to my parents who have always helped and supported me in these years, to my siblings, my fiance and my friends who have endured me in the worst moments and to all the people I met until today because, for some strange reasons, it is thanks to all of them if I have come this far. Thank you all.

Abstract

The growing use of technology and cyber tools has been embraced by the healthcare sector in many ways. On the market there are already some application available in areas such as diagnosis assistance, precision medicine, "social" robots for therapeutic assistance in the psychiatric field and others. However, an interesting and currently not completely exploited field of application is "case management and patient engagement". This thesis is inserted in the bigger project of optimizing the Italian Electronic Health Record, with the target of keeping under control patients that are following a certain therapy by examining automatically new exams results and dates, and communicating to the patients their progresses trying to keep them involved and in contact with their therapy, their results and obviously their reference doctors. In particular, this thesis tackles the problem of classifying diabetes patients, based on the therapy they are following in: patients that are following the correct therapy and patients that are not following the therapy, or for which the therapy is not correct. This will be done with the use of different Machine learning techniques that will be presented in this text together with an analysis of results and possible improvements.

Contents

1 Introduction **7**

2 State of the art **9**
2.1 e-Health . . . 9
2.1.1 Electronic Health Record standards . . . 13
2.1.2 User engagement . . . 16
2.2 Context of the project . . . 18

3 Analysis **21**
3.1 The dataset . . . 21
3.2 Machine learning . . . 24
3.2.1 Multi-layer Perceptron . . . 26
3.2.2 Support Vector Machine . . . 28
3.2.3 Decision Tree . . . 31
3.2.4 K-means . . . 33

4 Data processing **37**
4.1 Development environment and files architecture . . . 37
4.2 Vectors creation . . . 39
4.2.1 Temporary vector . . . 39
4.2.2 Final vector . . . 40
4.3 Vectors tagging . . . 41
4.3.1 Binary tag . . . 42
4.3.2 Multi tag . . . 43

5 Development and Results **47**
5.1 Code description . . . 47
5.2 Parameters choice . . . 48
5.2.1 Clustering . . . 48
5.2.2 Support Vectror Machine . . . 49
5.2.3 Multi-layer Perceptron . . . 51
5.2.4 Decision Tree . . . 53
5.3 Numerical results . . . 55

6 Conclusions **61**

Bibliography **65**

List of Figures

2.1 eHealth categories 10
2.2 EHR benefits 14
2.3 Share of citizens using the Electronic health records 15
2.4 Machine learning usage in the platform 18

3.1 Threshold logic unit 26
3.2 Perceptron algorithm 27
3.3 Multi-Layer Perceptron 27
3.4 Hard SVM algorithm 29
3.5 Soft SVM algorithm 29
3.6 ID3 algorithm 32
3.7 K-means algorithm 35

4.1 Temporary vector 40
4.2 Final vector 41
4.3 Examples of Binary tag 44
4.4 Examples of Multi-class tag 45

5.1 SVM results changing degree 50
5.2 SVM results changing C 51
5.3 SVM results changing gamma 51
5.4 MLP results changing hidden layer size 53
5.5 SVM ROC and precision/recall curves 58
5.6 MLP ROC and precision/recall curves 58
5.7 Decision tree ROC and precision/recall curves 58
5.8 Binary decision tree representation 59

Chapter 1

Introduction

The assignment that will be presented in this thesis is inserted in the bigger project of optimizing the Italian Electronic Health Record (EHR). I studied different approaches to insert the new Machine learning strategies, together with some Experts in public health and artificial intelligence, with the target of improving patient involvement and adhesion to the EHR.

My thesis relates to the project of keeping patient involved in the therapies they are following. In fact a lot of patient with chronic diseases must take periodic exams, follow a certain life style, and obviously take periodic medicines. However some of them don't pay much attention on the respect of exams date and in the progresses they are making.

The EHR gives us the possibility to keep tract of the different exams divided by typology and dates, but also on the medicines purchased at the pharmacy. These data can be used by a patient to see progresses or regressions and by the doctors to identify patients that are following the correct therapy and patients that are not following it or for which the therapy is not giving the desired results.

The goal is to combine these two point of view, helping both patients and doctors. In fact we want to use machine learning techniques to classify patients in the two groups described above, based on the therapy followed. To do so we need a starting dataset to train our models. We dispose of a dataset of almost 12000 patients, who took different types of exams. However I concentrated on patients affected by diabetes that needs periodical exams to keep always under control their values of glycated hemoglobin.

Finally we can say that the project presented here is basically, but not only, a binary classification of patients in two categories:

- patients that are following the correct therapy,
- patients that are not following the therapy, or for which the therapy is not correct.

To do so we tested different supervised algorithms and an unsupervised strategy, useful in our case since the starting dataset presented some structural problems that will be described later in this manuscript.

We now describe the organization of the thesis. In the second chapter we will present today standards in the Electronic Health Record and the milestone reached by new technologies in the healthcare sector, and then the different possibilities we considered to introduce machine learning technologies in the EHR platform proposed, before concentrating on the idea tested. In the third chapter we will analyze our problem, describing the data we used to made the tests and making a small introduction on machine learning standards and the algorithms I used. Then, on the fourth part we move to the pre-processing on the initial dataset, useful to build the optimum set of vectors usable by machine learning strategies. In the following chapter I described the last step of this project: the application of the predicting algorithms, the parameters choice and the analysis of the results.

Chapter 2

State of the art

2.1 e-Health

With the growing use of technology and cyber tools, it is evident that the healthcare sector is also embracing this in many ways including eHealth. It is a relatively recent healthcare practice supported by electronic processes and communication, dating back to at least 1999. Usage of the term varies such that a study in 2005 [1] found 51 unique definitions. We can summarize them deifining eHealth as: the use of information and communication technologies (ICT) for health.

In terms of normal usage eHealth can be accessed via any kind of electronic device or health monitoring system that is used by physicians in the healthcare practice or by individuals to monitor or improve their health. Digital healthcare solutions will transform the whole healthcare process to become more efficient, less expensive, and of higher quality. In fact, we expect that eHealth solutions will soon experience the same advances in other industries (e.g., telecom and banking) when IT systems and networks were deployed in the past [2]. As said above eHealth may vary with respect to functions, stakeholders, contexts, etc. by encompassing a broad range of medical informatics.

To have better clarity of eHealth on the different forms, Anuja Konda [3] divided eHealth in five categories: Website, Online synchronous, Online asynchronous, Smart Devices, and Electronic Device.

"Website" is one of the earliest forms of eHealth. In the year 2001 itself, it was reported that more than 70000 websites disseminating health information; more than 50 million people searching health information online. The "Website" category of eHealth offers widespread access to health information, and the advantages of interactivity, information tailoring and anonymity.

"Online synchronous" category refers to eHealth activities that can be carried out real-time interactively. Whereas "Online asynchronous" category refers to on-line eHealth activities in which the inaction between provider and health seeker is not real-time interactive. The primary difference between Website and Online (synchronous or asynchronous) is that information on websites tends to be static, whereas information gained through online interactive mode is dynamic and close to

real-time.

"Smart Devices" is another category through which eHealth can be provided. At present times, smart devices such as mobile phones apps have become common among individuals.

"Electronic device" category of eHealth refers to the electronic devices such as wearable, implants, or any monitoring and testing devises used by individuals for tracking their health. The difference between Smart devices and Electronic devices is in terms of dynamic alerts and suggestions received from smart devices vs. information stored and tracked using electronic devices.

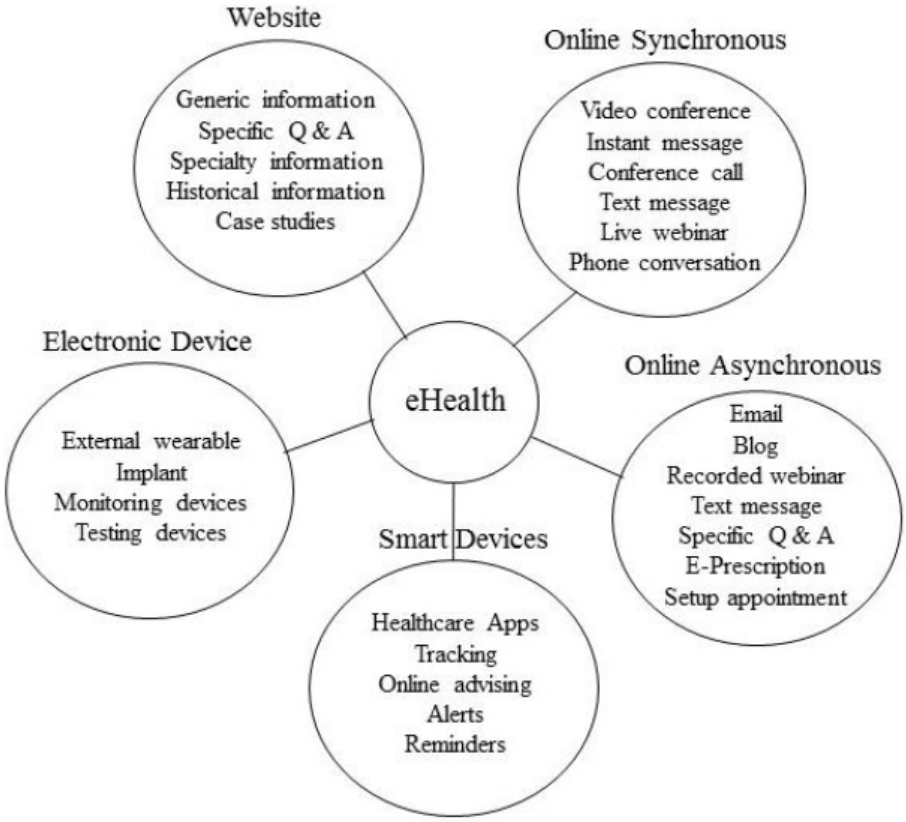


Figure 2.1: eHealth categories

For all these kind of eHealth, a successful implementation has to address a new paradigm in interconnection infrastructure: from proprietary record ownership to networked consumption of health record; from ad-hoc networking to interoperable interconnections; from separated IT solutions to orchestrated service creation and management with quality of service guarantees. All aspects of the new paradigm shall impose new challenges in the underlying consumer network and services infrastructure. W. Liu, E.K. Park and U. Krieger [2] divided the different challenges in 3 groups from the IT point of views: System Interconnection, eHealth Security and QoS and Operational Challenges.

System Interconnection Challenges: In early 2000's, healthcare IT systems were only isolate solutions that did not take the holistic view of healthcare process and outcomes. A number of initiatives have been reported to reform the healthcare IT systems, with a mix of successes and failures.The

HITECH (Health Information Technology for Economic and Clinical Health) legislation provided additional monetary incentives to interconnect to the eHealth systems.The major exchange infrastructure enables transmission of electronic health records covering patient demographics, progress notes, medication problems, prescriptions, vital signs, past medical history, immunizations, lab data and radiology reports. The major concern here is to ensure ubiquitous interconnection with the national health information network. Association of end-points has to be built upon network connections, but the association does not require dedicated connection channels. The association (sometimes called electronic bonding) authenticates the participating entities which are user or system end points. These aspects of interface requirements are missing (if not entirely) in current national health information network trials. The association setup process required provider identification, directory look-up and entity validation, subject data or functional context negotiation.

eHealth Security Challenges: New security concerns arise in transmitting and processing of electronic medical records, personal healthcare records, patient billing records, as well as public health alerts, among many parties with varying security, privacy and trust levels. As more and more healthcare providers are expected to convert internal data and transmit digital records over external infrastructure, passing multiple hops, there is a need for security guarantees with end-to-end control. When digital records can be easily shared, multiple parties are involved in eHealth transactions. While traditional security protocols govern two end-points, a new security paradigm of coordination has to be developed over healthcare network to accommodate different users while achieving scalability. Some study estimated that almost 400 people may have access to one's personal medical information throughout the typical care process. At any time, many collaborating providers may possess variable visibility/right of the data. Some parts of the record are confidential patient personal information while other fields are epidemic information for public analysis and research. And still there are other portion of the records such as billing and plan usage information for a few limited parties. A single encrypted data payload can no longer meet everyone's needs. An agile solution has to be found to allow fragmentation in diverse security settings while varying the protection levels at a different processing node. Still another challenge in eHealth security is the efficiency of the solutions. For example, some portion of the patient records may not be relevant to another eHealth party (e.g., such as the Lab processing entity thus only needs a patient identity without the detail records). With multiple parties in secure eHealth association(s), the potential growth of combinations is not scalable.

QoS and Operational Challenges: To guarantee Quality of Service in eHealth applications, we need an integrated view to combine the healthcare applications, the interconnection infrastructure support, as well as operational support with common services. Once a holistic understanding of all areas is established, we can better align our research in eHealth interconnection services and end-to-end solutions with QoS guarantee. Vital physiological data (such as body temperature, blood pressure, heart rate and cardiogram and blood sugar level that are constantly monitored by mobile devices) have to be fed into the patient records in real time. While medical records may not have to be always transmitted in real time, they have to be instantly available during a diagnosis

and consultation session with a doctor. When any change or irregularity happens for a sustained period of time, an alert has to be transmitted within a predefined time interval to the patient and his or her healthcare specialists to enable immediate actions. Moreover communication delays should be within a tolerable sub-second session setup time. Delay jitters have to be deterministic in order to avoid misunderstanding of verbal consultations. Medical image displays have to be in synchronization with voice sessions. Any degradation in service level or loss of service can impact the care given to many patients and could delay or hamper a critical surgery.

Nowadays solutions are reaching these standards and overtaking the challenges. However we can underline three key challenges also in the health and care sector, identified by the National Information Board [4]:

1. The health and wellbeing gap: If the nation fails to get serious about prevention then recent progress in healthy life expectancies will stall, health inequalities will widen, and our ability to fund beneficial new treatments will be crowded out by the need to spend billions of pounds on wholly avoidable illness.

2. The care and quality gap: Unless we reshape care delivery, harness technology and drive down variations in quality and safety of care, patients' changing needs will go unmet, people will be harmed who should have been cured, and unacceptable variations in outcomes will persist.

3. The funding and efficiency gap: If we fail to match reasonable funding levels with wide-ranging and sometimes controversial system efficiencies, the result will be some combination of poorer services, fewer staff, deficits, and restrictions on new treatments.

These gaps are exacerbated by the lack of integration across care services (hospital, community and home, clinical and social care, formal and informal settings). As financial pressures grow, and the gap between expectations, demand and resources increases, the need for the care system to make use of the best available technologies has become increasingly urgent. To ensure sustainability, health and care sector needs to move from a model of late disease management to early health. Information technology plays an essential and rapidly expanding role in empowering people to take charge of their own health, by providing information, support and control.

According to the text [4] we have to act now, with the revolution of 5G systems that will give us big advantages in systems interconnections and with the diffusion of Electronic Health Record in almost all Italian regions. Right in this period a new trial will start in Rome for new smart health and cities services, implemented by the agreement between the "Fondazione Policlinico Gemelli", "Cassa depositi e risparmi" and "Acea". The goal of the project is to improve health services such as prevention activities, diagnoses, therapies to fight diseases, strengthening health systems and improving citizens' access to care [5].
The company Almaviva, with whom I collaborated, together with the hospital company Asst Vimercate are realizing "the oracle of health": a project started last November and that sees clinicians and computer scientists working side by side to develop intelligent algorithms to take

medical decisions even more supported by the evidence, [6]. The full operativity is scheduled for 2021.

However we have to keep in mind some limitations of the application of new technologies to patient care and assistance. In fact, in January of this year 52 experts, selected from the world academic, industry and civil society, drafted a document where the European guidelines for the use of artificial intelligence were presented. As this technology is increasingly used in this area and is the center of this project it is better to report a summary of this document, [7].

The first principle contained in the EU guidelines on Artificial Intelligence (AI) states that there must always be a human control, because the goal is to improve human action and its rights, not to reduce its autonomy. The second one provides that the algorithms must be safe, reliable and resistant to errors or inconsistencies in the different phases of the life cycle of AI systems. The third is that citizens must always be informed of the use of their personal data and have full control so they are not used against them, and this must be done in line with the European rules on the protection of privacy of the General Data Protection Regulation (GDPR). The fourth principle provides transparency, guaranteeing the traceability of artificial intelligence systems. The fifth, to guarantee diversity and non-discrimination, with human beings who may be able to change decisions algorithms taking into account all the necessary factors. And with mechanisms of human appeal against the decisions of the algorithms, to ensure the responsibility of who manages the calculation systems in case of damage or accidents.

Finally, artificial intelligence will have to work in favor of social and environmental well-being, increasing ecological sustainability. "A person must always know when he is in front of a machine and not a human being - explained the EU commissioner, Maryia Gabriel, for this - AI systems must be recognizable". The EU is pushing for companies and researchers to develop artificial intelligence in an ethical and transparent way.

After this first document, it is up to the World Health Organization to launch the first guidelines for the use of digital technologies for health-related interventions. "Exploiting the power of digital technologies is essential for obtaining universal health coverage", is the premise of DR Tedros Adhanom Ghebreyesus Director General of the WHO. "Digital technologies are not ends in themselves - reiterates Ghebreyesus - they are essential tools to promote health, keep the world safe and serve vurnable people" [8].

2.1.1 Electronic Health Record standards

The Electronic Health Record (EHR) is the set of digital health and social-health data and documents generated by past and present clinical events concerning a patient [9]. The main objectives of the EHR are:

1. facilitate patient health care;
2. offer a service that can facilitate the integration of different professional skills in the health field;

3. provide a consistent clinical information base realted to the patient.

The availability of an EHR system promotes the improvement of all care and treatment activities throughout the patient's life, therefore the Record is continuously fed by the subjects who take care of the patient himself within the National Health Service (NHS) and of the socio-regional services, as well as, at the request of the citizen, with the medical data held by the same, thus determining the absence of additional charges for public finances. The EHR is established by the Regions and the autonomous provinces, in compliance with the current legislation on the protection of personal data, and concern to a wide range of activities related to the provision of health services, from prevention to the verification of quality of care. Specifically, the initiative related to the implementation of EHR systems aims at the overall improvement of the quality of services concerning:

- prevention, diagnosis, treatment and rehabilitation;
- study and scientific research in the medical, biomedical and epidemiological fields;
- health planning, verification of the quality of care and evaluation of health care.

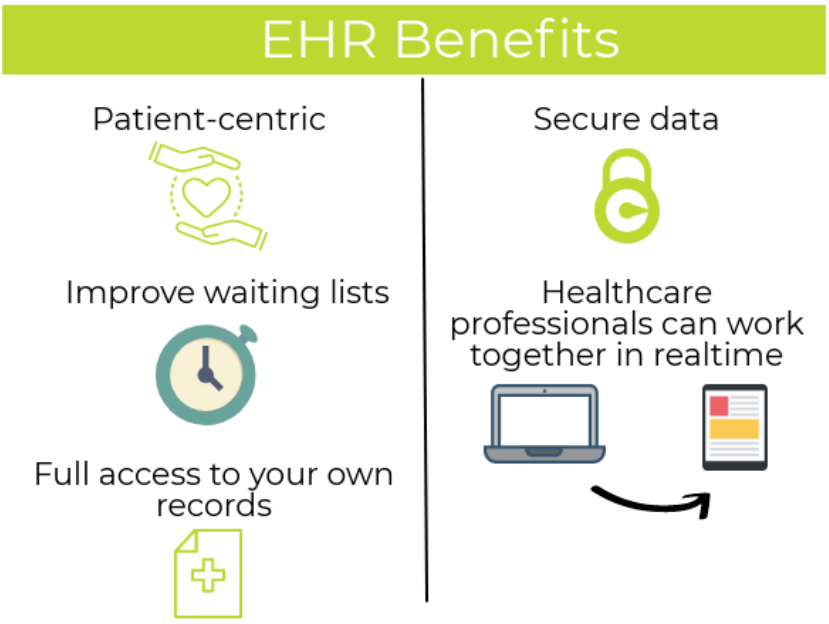


Figure 2.2: EHR benefits

Regarding the access to the EHR, data and documents consultation can be carried out only with the consent of the user and always in compliance with professional secrecy, except for specific cases of medical emergency for which particular operating procedures are envisaged. From the point of view of the contents and of the documentation made available, the Electronic Health Record is made up of a minimum set of documents that must be made available by the system and

by supplementary documents that allow to widen its sphere of use to support the different paths activated, in order to guarantee continuity of care. In particular, the minimum core consists of:

- identifying and administrative data of the client;
- reports;
- First Aid reports;
- letters of dismissal;
- synthetic health profile;
- pharmaceutical dossier;
- consent or denial of donation of organs and tissues.

In 2008 the Italian Ministry of Health conducted a study to determine the status of adoption of the EHRs [10]. The study showed that 43% of the local health authorities, 62% of the hospitals and 19% of local clinics use the electronic health records. As far as health workers using electronic health records, a coverage of 71% was estimated for general practitioners and paediatricians, 67% for other doctors in the National health service while 29% of nurses and 5% of pharmacist adopted it. The electronic health record is in general adopted to carry out 52% of the specialist hospital services, 33% of the pharmaceutical services and 24% of the emergency care services. Moreover, 43% of the Italian regions declare to have adopted electronic health records for the management of part of their healthcare data. These data show that in 2008 the EHR was quite well developed in Italy, even without a specific national legal framework.

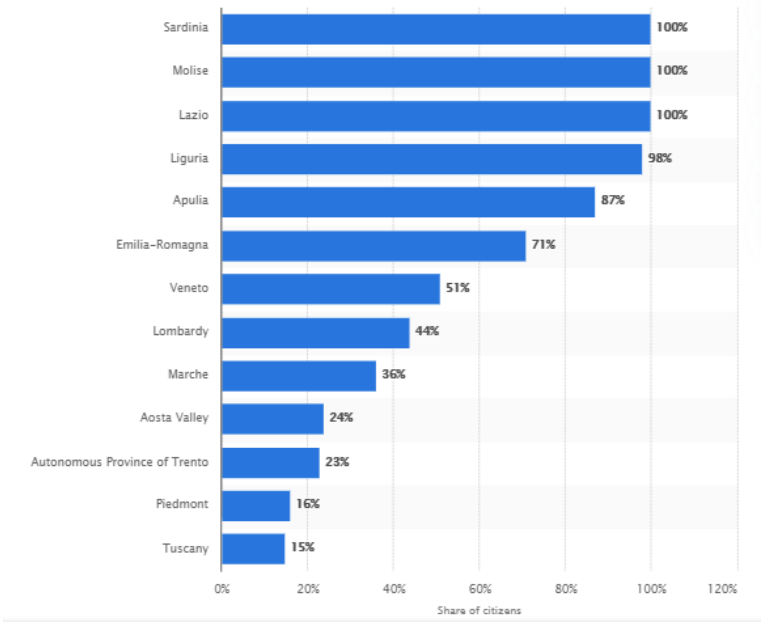


Figure 2.3: Share of citizens using the Electronic health records

Today the regions where the EHR is implemented and operative are 14 and the target for 2020 is to bring it to all the regions [11]. The citizens with active EHR are only 21% with a target of 70% for the 2020. Above I reported the data Published by Statista Research Department [12], 13 March 2019, on the percentage of citizens using the Electronic health records (EHR) in Italy in the 4th quarter 2018, divided by region.

2.1.2 User engagement

On the market there are already some application experiences in areas such as diagnosis assistance, precision medicine, "social" robots for therapeutic assistance in the psychiatric field and others. However, an interesting and currently not at completely exploited field of application is case management and patient engagement. This is an area of intervention with great potential. Unlike the applications closer to the medical theme, which present some important critical issues (lack of datasets, difficulty in documenting decision-making mechanisms, etc.), applications oriented to improving case management and making available to stakeholders useful and engaging functionalities, should be considered a priority for different a number of factors:

- Ability to enhance existing information assets;
- Time-to-market and limited effort;
- Substantial differentiation from the offer of competitors;
- Creation of a "recognizable" offer for its approach philosophy and consistent with our claim of "Digital absolute";
- Strong orientation to the "daily" needs of the citizen (simple but not trivial).

User engagement can be reached through different methods, one of them that has been exploited during these years is user interactivity. It may be defined as a software characteristic that allows human users to input and change the system output dynamically. Previous researchers have highlighted the diversity of users in terms of abilities, interests and needs. Thus an adaptive user interface that is personalized to satisfy the heterogenous needs of users may be one solution. In general, interactivity in websites is favored by users over passive information. Thus, interactive engagement with users is a central theme in eHealth exemplified by online conversational agents that mimic a healthcare professional. These software based intelligent virtual agents are often termed eCoaches.

Very few randomized trials have been conducted in this context [13]. A trial involving 200 adults evaluated user engagement for eHealth interventions (interactive website versus non-interactive website versus mobile app) to decrease dementia risk [14]. Although, there was no difference in the ease of understanding, the non-interactive information website was rated higher as being easier to navigate. Thus, paradoxically, interactivity did not show a clear benefit over a static website. The

researchers speculate that while interactivity is generally favored, in middle-aged adults traditional modes of delivery may be more appropriate.

Moreover, usage of eHealth apps is often limited by external factors such as technology access and the so-called "ceiling effect". The ceiling effect refers to the lack of benefit of using an app among patients who have mild or well-controlled disease at baseline. For instance, in a 2-year longitudinal study of a web application showed that patients with well controlled diabetes did not continue long-term [15]. A 2014 review of 656 diabetes apps found that usability was moderately good and decreased in apps with more features[16]. The question arisen whether to improve usability by limiting the number of features including interactivity? Yet, implementation of basic user interactivity is often lacking in many apps. For example, a 2016 review of 24 tuberculosis apps found minimal functionality in terms of engaging patients in self-care [17]. Inconsistencies in user interface design were also apparent. Furthermore, as the population ages, an increasing number of eHealth users will be the elderly. Standard user interface design may not be appropriate for this group of patients. This point of view is rapidly gaining support. A recent study redesigned a diabetes app for use by older patients [18]. Thus, a few small changes to the interface layout substantially improved usability for this demographic group.

Concluding, interactivity can make a great difference but we have to adapt it to patients in terms of age, pathology and technological skills.

Today, in Italy, there are some mobile applications that allow you to request assistance at home to help people with motor and/or cognitive difficulties at various levels [19]. An example is "Ugo", now active in Turin, Milan and Genoa, which offers paid personalized assistance, but there are also similar free services, which count on volunteering. In this regard, in Europe we are working on the Care4Dem web platform to develop an advanced model digital welfare aid (should be ready for use within a year). In Emilia Romagna, Lepida fielded the eCare service, carried out for the Bologna and Ferrara Local Health Authorities, and the package of online services available in the field of digital welfare thanks to the use of the credentials of the Public digital identity system or Spid. The "Policlinico di Bari" has developed a mixed reality technology, which allows you to consult at distance, for patients with heart attack or stroke, for example. It gives the possibility to communicate in real-time, data and information exchange and to take advantage of more holopresence technology health workers who thanks to virtual and augmented glasses can remotely assess the conditions of the patient at home. In this way the Policlino di Bari has already seen hospitalizations decrease by 40%, with over 506 thousand consultations online in 2017 and savings of 30 million euros.

All of this can be improved and enhanced thanks to the online sharing of data and information between patient and doctor/operator/assistant, and to the medical records that have passed from the traditional physical support to the online one, with reports that can be read on the PC and on the smartphone.

2.2 Context of the project

The project that will be presented in this thesis is be placed in the context of integrating Campania EHR in a new platform. The objective of this new platform is to create an environment where producing solutions for the total dematerialization of care processes. It is not as a "finished" solution but rather a context on which, following an analysis of individual clients' requirements, it is possible to reach the definition and activation of a single definitive ecosystem, with a wide re-use of the experiences matured and appropriately consolidated.

In the platform the application of Machine Learning techniques was hypothesized in the information enhancement phase for extracting meaning from parts of unstructured documents and for reconstructing the links between different documents, both on the basis of content that on a temporal basis (contemporaneity, consequentiality, classification of the continuity of a therapy). Thanks to these applications, a browsing experience between "a graph" documents is offered to the consumer. Moreover it can provide to secondary stakeholders, such as ASLs, treatments, information on adherence to diagnostic therapeutic pathways, possible interactions between drugs, respect for therapeutic indications of drugs, coherent use of first line and second line drugs, medicine of initiative and many others.

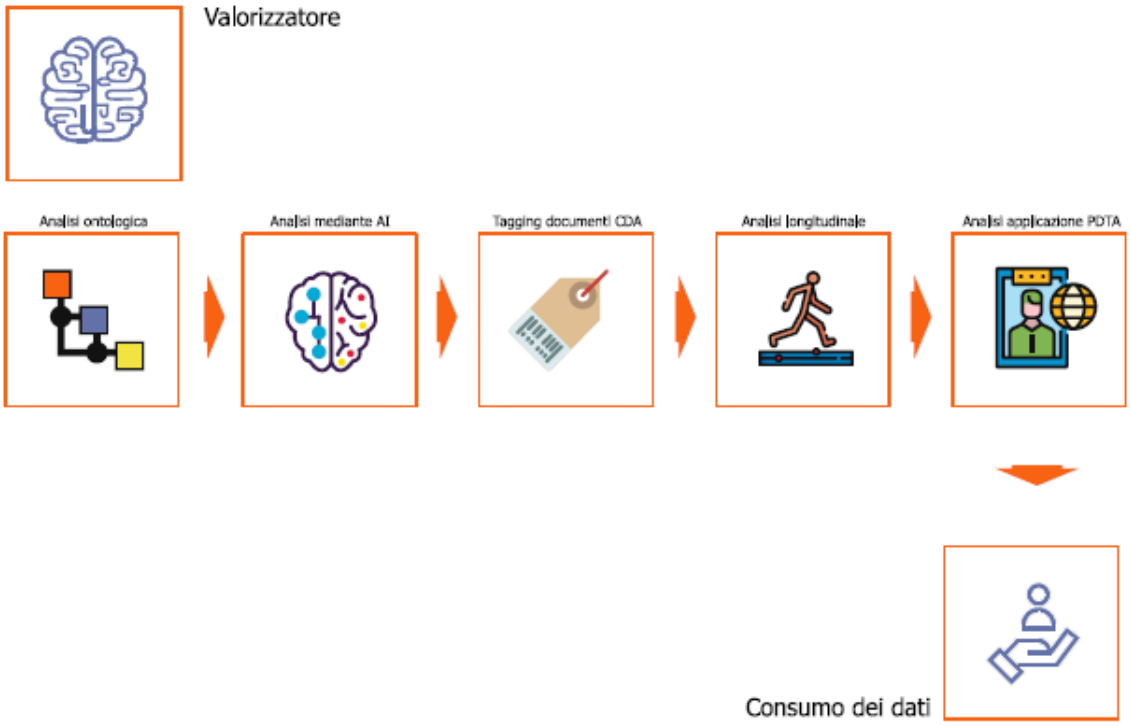


Figure 2.4: Machine learning usage in the platform

In the area of information consumption, Machine learning applications can lead to a platform activity based on the continuous profiling of user needs, both in individual and class terms (for example, patients affected by a specific pathology, general practitioners, orthopedic specialists, gynecologists, etc.), while allowing a sharing of the application functionalities between the two

main protagonists of the clinical process: patient and doctor. Also in this area, the use of Machine learning techniques can produce substantial advantages for secondary stakeholders, above all thanks to the ability to produce advanced data analysis on the use of the platform. Above I report a graph showing the process that a new document will pass through entering the platform, and so where the new technologies will be used to increase the value of input data.

In this thesis we will focus on the use of predictive techniques for a classification of patient with a specific disease. Specifically, I will exploit the field of patient engagement with the target of keeping under control patients that are following a certain therapy by examining automatically new exams results and dates, and communicating them their progresses trying to keep them involved and in contact with their therapy, results and obviously also their reference doctors.

In the following chapter I will present the available data, the procedure used to analyse them and the results of my tests.

Chapter 3

Analysis

In this chapter I will describe the dataset used for the tests and explain the process I needed to perform on it in order to understand the final vector created. I will then present the basis of Machine Learning and describe the techniques used in this project and their basic functioning.

3.1 The dataset

As explained before, the main purpose of this project is to analyze the use of machine learning techniques to classify patients that are following a certain therapy according to their exams results and clinical data. The techniques that will be presented in this thesis could be applied and adapted to different diseases and could be inserted in the project of a more complete version of the EHR with a focus on patient involvement and awareness. The first step was to consider only patients with diagnosed diabetes and try to classify them in patients with a correct therapy and that are following it or patients that are not following the therapy or for which the therapy is not correct. I decided to aggregate two types of patients in the second category because, from the data, it was not possible to distinguish between them; this will be better explained later. Then we can analyze the results and the possible applications and improvements.

I started from a quite large anonymized dataset, produced by private laboratories in the southern Italy accredited with NHS (National Health Service), that reports the results of 11835 patients taken during four years and seven months: from 01/01/2013 to 01/08/2017. This dataset contains the results for different types of exams (for Cholesterolemia, Diabetes and Nephropathy) and so for different type of patient's pathologies.
The initial structure of the dataset was composed of vectors of twelve columns, with the following content:

1. Patient's identification code: a number from 1 to 131445 (not consecutive);
2. Exam's date: as said above dates are distributed during four years, from 2014 and 2017;
3. Mnemonic code of the exam: short string identifying the exams, COL, HDL, LDL for Cholesterol exams, HBA1C for Diabetes exams and MICROA for Nephropathy;

4. Exam tag: cholesterolemia, diabetes and nephropathy;
5. Element under examination: Cholesterol, Cholesterol hdl, Cholesterol ldl, Glycated haemoglobin, Glycated haemoglobin index and Microalbuminuria;
6. Exam identification code: numerical code different for each element under analysis;
7. Numeric result of the exam;
8. Pathological examination result: 0 if the numerical results is in the limits defining "normality", 1 if not;
9. Sex of the patient: if necessary for results identification;
10. Age of the patient: if necessary for results identification;
11. Minimum value in which the numerical result is considered to be "normal" (sometimes dependant on sex and age);
12. Maximum value in which the numerical result is considered to be "normal" (sometimes dependant on sex and age);
13. Diagnosis: results evaluation (sometimes empty);
14. Pathology code present: small string identifying an already diagnosed pathology or certifying a disability class;
15. Pathology description: description of the code above, the most present are 'diabetes mellitus' and 'subject with less than 6 years or over 65 and low income'.

As each line presents all these categories, we have more than one line for each patient: one for each exam taken, repeating the pathology, or one for each pathology already present, repeating data and exam's result. This type of construction will lead to a lot of repetitions and useless data. For our purposes we are interested in glycated haemoglobin results that indicates the average plasma glucose concentration during the previous three months, and we want to analyze the progresses of this value during the years. So, from the staring number of patients I selected only those who took this exam, the new value of subjects then became 3396.

Moreover, we are interested only in patient with diagnosed diabetes (diabetes tag present in pathology), with this specification the number decreases again to 763 patients with different number of exams. It, in fact, varies from 1 to 30 exams. As machine learning techniques that are able to handle variable inputs do not exist, I decided to consider only patients with at least three exams; for those who present a greater number of analysis we took into consideration only the last three. With this last consideration we arrive to the final number of vectors: 310, which is not elevated but enough for a preliminary analysis. As said above, the glycated haemoglobin results report an average of the last three months of cure so we choose to consider patient with at least three exams

as we can analyze the trend for a sufficiently large amount of time: from 9 months to three years. In fact, patient must take these exams with minimum 3 months of distance and maximum a year.

Now, we have to pre-process the dataset to create a vector easy-to-use for machine learning algorithms. From the staring dataset described above we kept only:

- Patient identification code;
- Exams' dates;
- Exams' results.

So, we finally obtain a seven columns vector with the patient code, the date of its last three exams and their results. The code is the originally one, as well as the numerical results, while for the dates we had to change their format to be able to process it. After some trial I decided to use the difference in years (decimal number) from two consecutive exams, and for the last one I considered the difference with the last possible date of analysis. With this choice we must consider the right period of time between two exams to be between 0.24 and 1 (with some variance).

However, the main problem of the dataset is that it does not present any initial tag for the classification. It is strange to base a project on a classification target through machine learning without a proper dataset but I underlined the tagging rules for every patient with an expert and reproduced the tag process with a linear algorithm, [20, 21, 22]. After this I applied some known machine learning techniques, analyzing accuracy and execution time, focusing on the use of Decision Trees which gave me the possibility to compare the decision criteria of classification with the rules applied in the liner code.

Considering that this project is thought as an additional application for the EHR, the individuation of rules done with the expert was initially based on all the possible information I can retrieve from the platform. Therefore, taking into consideration a single patient with diagnosed diabetes, the available data are those described above, plus:

- Code and description of prescribed medicines;
- Prescription date;
- Collection date.

These elements will be retrievable in future and will contribute more to patient classification. Now, considering the data available for each patient, it is possible to create a list of patterns that highlight problems in the effectiveness of the therapy followed but also not attention to it. So, we can consider two classification groups:

1. Inadequate therapy or not followed by the patient:
    - Last analysis present values outside the limits while the previous exams had stable values;
    - The analysis show values that are above the limit, stable over time;

- There are repeated patterns in the time of growth and decrease of values, exiting and re-entering the limits several times;
- Failure to repeat the analysis in the minimum interval of three months and a maximum of a year;
- Failure to withdraw prescribed medicines.

2. Adequate therapy followed by the patient:
   - Values constantly within the limits;
   - Last analyses present internal limits while previous exams have external values (but constant controls are suggested);
   - Repetition of the analysis in the minimum interval of three months and a maximum of a year;
   - Withdrawal of prescribed medicines.

The non-adherence to treatments could lead to analysis with values outside the limits and therefore to the identification of the therapy as inadequate when, actually, it is not followed properly by the patient. Another factor that can lead to out-of-limits results is the patient's lifestyle: physical activity, diet and the same stress can compromise the analyses performed. So, we decided to divide the classification into: "Adequate therapy followed by the patient" and "Inadequate therapy or not followed by the patient". It is fair to specify that in the case of identification of a non-adherent patient or a patient with inadequate therapy, the platform will simply alert the interested parties, patient and doctor, in order to facilitate communication between them.

Previous considerations were made before the construction of the final vector and so before knowing that we had the possibility to elaborate only three exams. After this and after some trial, we establish some different and more precise rules for patients' classification. Moreover, we decided to test two classification divisions: the first divide the patients in the two classes described above, while the second one introduces a third class for patient that present exams in the limit but does not respect the minimum or maximum date difference or for patient that present a decreasing trend for all the three exams and will be better classified with more exams. We will see better in detail the different rules and this division in the next chapter. However, this second type of classification with three classes seems to better fit the dataset and the patients, also according to the expert, but we will compare them in the following chapters.

3.2 Machine learning

The main part of this project focuses on the application of machine learning techniques to the tagged dataset and the analysis of their performances. The term machine learning refers to the automated detection of meaningful patterns in data. In the past couple of decades, it has become a common tool in almost any task that requires information extraction from large datasets [23]. One

common feature of all applications is that, in contrast to more traditional uses of computers, in these cases, due to the complexity of the patterns that need to be detected, a human programmer cannot provide an explicit, fine-detailed specification of how such tasks should be executed.

Learning is, of course, a very wide domain. Consequently, the field of machine learning has branched into several sub-fields dealing with different types of learning tasks. The main division to make is Supervised vs Unsupervised learning algorithms.

Supervised learning algorithms build a mathematical model using a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and a desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Supervised learning algorithms include classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, discrete output, while regression algorithms are used when the outputs may have any numerical value within a range, continuous domain.

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms therefore learn from test data that has not been labelled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data.

Another distinction we can made is between classification and regression. Classification defines the algorithms that starting from a features vector return a predicted label. In practice the output is in a discrete set, it can be binary, so {0,1}, or multi-class. Regression instead defines algorithms which outputs are in the continuous domain. This thesis is centered on the classification problem, principally binary but also multi-class. We can divide in two phases the generation and evaluation of a classifier:

1. Training phase: first step for the algorithm that receives a series of samples and their labels; this values train the algorithm that modifies its parameter trying to understand the laws of the classification problem.

2. Test phase: the trained algorithm receives new data and predict the labels.

For this project I used three different supervised algorithms: Multi-layer Perceptron, Support Vector Machine, and Decision Tree classifiers. And I tested also an unsupervised algorithm, K-means, to test if my preliminary classification is approximately correct. Below we will describe the

theory behind these techniques and later in this manuscript we will go deeper in their practical functioning.

3.2.1 Multi-layer Perceptron

The Perceptron is one of the simplest ANN (Artificial Neural Network) architectures, invented in 1957 by Frank Rosenblatt. It is based on a slightly different artificial neuron called a threshold logic unit (TLU), or sometimes a linear threshold unit (LTU): the inputs and output are numbers (instead of binary on/off values) and each input connection is associated with a weight. The TLU computes a weighted sum of its inputs ($z = w_1 x_1 + w_1 x_1 + \ ... + \ w_1 x_1 = \mathbf{x}^T \mathbf{w}$)), then applies a step function to that sum and outputs the result: $h_w(\mathbf{x}) = \ step(z)$, where $z = \mathbf{x}^T \mathbf{w}$.
The most common step function used in Perceptron is the Heaviside step function, sometimes the sign function is used instead:

$$heaviside(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}; \quad sign(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}.$$

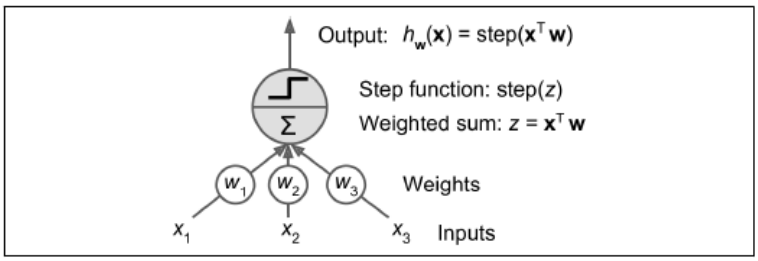


Figure 3.1: Threshold logic unit

A single LTU can be used for simple linear binary classification. It computes a linear combination of the inputs and if the result exceeds a threshold, it outputs the positive class or else outputs the negative class. Below it is reported the Perceptron algorithm retrievable online. Where $\mathbf{x}$ represents the matrix of input features, $\mathbf{w}$ the weight matrix and T the maximum number of iterations.

A Perceptron is simply composed of a single layer of TLUs, with each TLU connected to all the inputs. To represent the fact that each input is sent to every TLU, it is common to draw special passthrough neurons called input neurons: they just output whatever input they are fed. All the input neurons form the input layer. Moreover, an extra bias feature is generally added ($x_0 = 1$): it is typically represented using a special type of neuron called a bias neuron, which just outputs 1 all the time. It is often inserted in the notation.

In their 1969 monograph titled *Perceptrons* [24], Marvin Minsky and Seymour Papert highlighted a number of serious weaknesses of Perceptrons, in particular the fact that they are incapable

Perceptron

initialize: $\mathbf{w}_1 = \mathbf{0}$
for $t = 1, 2, \ldots, T$
  receive $\mathbf{x}_t$
  predict $p_t = \text{sign}(\langle \mathbf{w}^{(t)}, \mathbf{x}_t \rangle)$
  if $y_t \langle \mathbf{w}^{(t)}, \mathbf{x}_t \rangle \leq 0$
    $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + y_t \mathbf{x}_t$
  else
    $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)}$

Figure 3.2: Perceptron algorithm

of solving some trivial problems (e.g., the Exclusive OR (XOR) classification problem). However, it turns out that some of the limitations of Perceptron can be eliminated by stacking multiple Perceptron. The resulting ANN is called a Multi-Layer Perceptron (MLP).

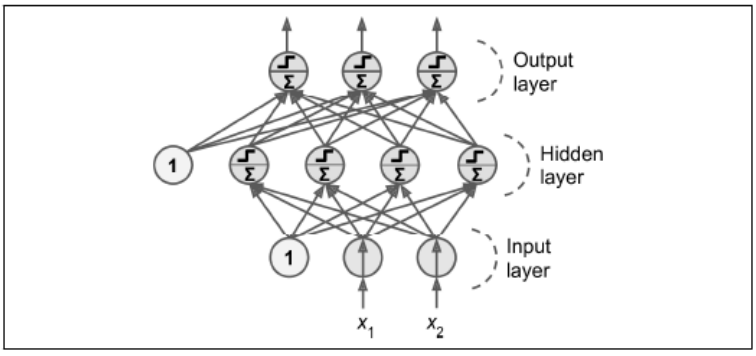


Figure 3.3: Multi-Layer Perceptron

An MLP is composed of one (passthrough) input layer, one or more layers of TLUs, called hidden layers, and one final layer of TLUs called the output layer. Every layer except the output layer includes a bias neuron and is fully connected to the next layer. It uses a backpropagation training algorithm; in short, it is Gradient Descent, using an efficient technique for computing the gradients automatically. Below we will run through the algorithm in a bit more detail:

- It handles one mini-batch at a time, and it goes through the full training set multiple times. Each pass is called an epoch.
- Each mini-batch is passed to the network's input layer, which just sends it to the first hidden layer. The algorithm then computes the output of all the neurons in this layer (for every instance in the mini-batch). The result is passed on to the next layer, its output is computed and passed to the next layer, and so on until we get the output of the last layer, the output layer. This is the forward pass.
- Next, the algorithm measures the network's output error (i.e., it uses a loss function that compares the desired output and the actual output of the network, and returns some measure of the error).

- Then it computes how much each output connection contributed to the error. This is done analytically by simply applying the chain rule, which makes this step fast and precise.

- The algorithm then measures how much of these error contributions came from each connection in the layer below, again using the chain rule and so on until the algorithm reaches the input layer. As we explained earlier, this reverse pass efficiently measures the error gradient across all the connection weights in the network by propagating the error gradient backward through the network (hence the name of the algorithm).

- Finally, the algorithm performs a Gradient Descent step to tweak all the connection weights in the network, using the error gradients it just computed.

In order for this algorithm to work properly, it was made a key change to the MLP's architecture: the step function was replaced with the logistic function, $\sigma(z) = 1/(1 + e^{-z})$. This was essential because the step function contains only flat segments, so there is no gradient to work with (Gradient Descent cannot move on a flat surface), while the logistic function has a well-defined nonzero derivative everywhere, allowing Gradient Descent to make some progress at every step.

The backpropagation algorithm works well with many other activation functions, not just the logistic function. Two other popular activation functions are:

- The hyperbolic tangent function: $\tanh(z) = 2\sigma(2z) - 1$. Just like the logistic function it is S-shaped, continuous, and differentiable, but its output value ranges from –1 to 1, which tends to make each layer's output more or less centred around 0 at the beginning of training. This often helps speed up convergence.

- The Rectified Linear Unit function: $ReLu(z) = \max(0, z)$. It is continuous but unfortunately not differentiable at z = 0 (the slope changes abruptly, which can make Gradient Descent bounce around), and its derivative is 0 for z < 0. However, in practice it works very well and has the advantage of being fast to compute.

As many supervised techniques, MPL can be used both for regression and classification, binary and multi-class.

3.2.2 Support Vector Machine

A Support Vector Machine (SVM) is a very powerful and versatile Machine Learning model, capable of performing linear or nonlinear classification, regression, and even outlier detection. The SVM algorithmic paradigm faces the sample complexity challenge by searching for "large margin" separators. The margin of a hyperplane with respect to a training set is defined to be the minimal distance between a point in the training set and the hyperplane [25]. If a hyperplane has a large margin, then it will still separate the training set even if we slightly perturb each instance.

Hard-SVM is the learning rule in which we return an ERM hyperplane that separates the training set with the largest possible margin. Below it is reported the hard-SVM algorithm that

searches for w of minimal norm among all the vectors that separate the data and for which $|\langle \mathbf{w}, \mathbf{x}_i \rangle + b| \geq 1$ for all i. In other words, we enforce the margin to be 1, but now the units in which we measure the margin scale with the norm of w. Therefore, finding the largest margin halfspace boils down to finding w whose norm is minimal.

Hard-SVM

input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
solve:

$$(\mathbf{w}_0, b_0) = \operatorname*{argmin}_{(\mathbf{w},b)} \|\mathbf{w}\|^2 \text{ s.t. } \forall i,\ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad (15.2)$$

output: $\hat{\mathbf{w}} = \frac{\mathbf{w}_0}{\|\mathbf{w}_0\|}, \quad \hat{b} = \frac{b_0}{\|\mathbf{w}_0\|}$

Figure 3.4: Hard SVM algorithm

Restricting the algorithm to output a large margin separator can yield a small sample complexity even if the dimensionality of the feature space is high (and even infinite). There are two main issues with hard margin classification: first it only works if the data is linearly separable, and second it is quite sensitive to outliers. To avoid these issues, it is preferable to use a more flexible model. The objective is to find a good balance between keeping the street as large as possible and limiting the margin violations (i.e., instances that end up in the middle of the street or even on the wrong side). This is called soft margin classification. Below is reported the Soft-SVM optimization problem where we allow the constraint to be violated for some of the examples in the training set. This can be modeled by introducing non-negative slack variables, $\xi_1, ..., \xi_m$, and replacing each constraint $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ by the constraint $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$. That is, ξ_i measures by how much the constraint $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ is being violated. Soft-SVM jointly minimizes the norm of w (corresponding to the margin) and the average of ξ_i (corresponding to the violations of the constraints).

Soft-SVM

input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
parameter: $\lambda > 0$
solve:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \left(\lambda \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i \right) \quad (15.4)$$
$$\text{s.t. } \forall i,\ \ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \ \text{ and } \ \xi_i \geq 0$$

output: $\mathbf{w}, b$

Figure 3.5: Soft SVM algorithm

Although linear SVM classifiers are efficient and work surprisingly well in many cases, many datasets are not even close to being linearly separable. One approach to handling nonlinear datasets

is to add more features, such as polynomial features; in some cases, this can result in a linearly separable dataset. This technique enables us to enrich the expressive power of half-spaces by first mapping the data into a high dimensional feature space, and then learning a linear predictor in that space. While this approach greatly extends the expressiveness of half-space predictors, it raises both sample complexity and computational complexity challenges. The basic paradigm is the following:

1. Given some domain set X and a learning task, choose a mapping: $\varphi : X \to F$, for some feature space F, that will usually be $\mathbf{R}^n$ for some n (however, the range of such a mapping can be any Hilbert space, including such spaces of infinite dimension, as we will show later).

2. Given a sequence of labeled examples, $S = (x_1, y_1), ..., (x_m, y_m)$, create the image sequence $\hat{S} = (\varphi(x_1), y_1), ..., (\varphi(x_m), y_m)$.

3. Train a linear predictor h over $\hat{S}$.

4. Predict the label of a test point, x, to be $h(\varphi(x))$.

The success of this learning paradigm depends on choosing a good φ for a given learning task: that is, a φ that will make the image of the data distribution (close to being) linearly separable in the feature space, thus making the resulting algorithm a good learner for a given task. Picking such an embedding requires prior knowledge about that task. However, often some generic mappings that enable us to enrich the class of halfspaces and extend its expressiveness are used.

However, the computational complexity of such learning may still pose a serious hurdle; computing linear separators over very high dimensional data may be computationally expensive. The common solution to this concern is kernel based learning. A kernel is a type of a similarity measure between instances. The special property of kernel similarities is that they can be viewed as inner products in some Hilbert space (or Euclidean space of some high dimension) to which the instance space is virtually embedded. The "kernel trick" enables computationally efficient implementation of learning, without explicitly handling the high dimensional representation of the domain instances. Kernel based learning algorithms, and in particular kernel-SVM, are very useful and popular machine learning tools. Their success may be attributed both to being flexible for accommodating domain specific prior knowledge and to having a well developed set of efficient implementation algorithms. I used two types of kernel: polynomial and RBF kernels.

In the first case, adding polynomial features is simple to implement and can work great with all sorts of Machine Learning algorithms (not just SVMs), but at a low polynomial degree it cannot deal with very complex datasets, and with a high polynomial degree it creates a huge number of features, making the model too slow. Fortunately, when using SVMs you can apply an almost miraculous mathematical technique called the kernel trick (it is explained in a moment). It makes it possible to get the same result as if you added many polynomial features, even with very high degree polynomials, without actually having to add them. So there is no combinatorial explosion of the number of features since you don't actually add any features.

As said above, the other technique used to tackle nonlinear problems is to add features computed using a similarity function that measures how much each instance resembles a particular landmark. The Gaussian Radial Basis Function (RBF) is the most used: $\phi_\gamma(\mathbf{x}, l) = \exp -\gamma||\mathbf{x} - l||^2$, with $\gamma = 0.3$. It is a bell-shaped function varying from 0 (very far away from the landmark) to 1 (at the landmark). Just like the polynomial features method, the similarity features method can be useful with any Machine Learning algorithm, but it may be computationally expensive to compute all the additional features, especially on large training sets. However, once again the kernel trick does its SVM magic: it makes it possible to obtain a similar result as if you had added many similarity features, without actually having to add them.

Other kernels exist but are used much more rarely. For example, some kernels are specialized for specific data structures. String kernels are sometimes used when classifying text documents or DNA sequences (e.g., using the string subsequence kernel or kernels based on the Levenshtein distance).

3.2.3 Decision Tree

A decision tree is a predictor, $h : X \to Y$, that predicts the label associated with an instance x by traveling from a root node of a tree to a leaf. For simplicity we focus on the binary classification setting, namely, $Y = \{0, 1\}$, but decision trees can be applied for other prediction problems as well. At each node on the root-to-leaf path, the successor child is chosen on the basis of a splitting of the input space. Usually, the splitting is based on one of the features of x or on a predefined set of splitting rules. A leaf contains a specific label.

Decision tree learning algorithms are based on heuristics such as a greedy approach, where the tree is constructed gradually, and locally optimal decisions are made at the construction of each node. Such algorithms cannot guarantee to return the globally optimal decision tree but tend to work reasonably well in practice. A general framework for growing a decision tree is as follows.
We start with a tree with a single leaf (the root) and assign this leaf a label according to a majority vote among all labels over the training set. We now perform a series of iterations. On each iteration, we examine the effect of splitting a single leaf. We define some "gain" measure that quantifies the improvement due to this split. Then, among all possible splits, we either choose the one that maximizes the gain and perform it, or choose not to split the leaf at all.

Below we provide a possible implementation. It is based on a popular decision tree algorithm known as "ID3" (short for "Iterative Dichotomizer 3"). We describe the algorithm for the case of binary features, $X = \{0, 1\}$, and therefore all splitting rules are of the form $\mathbf{1}_{[x_i=1]}$ for some feature $i \in [d]$. The algorithm works by recursive calls, with the initial call being $ID3(S; [d])$, and returns a decision tree.

We can extend this result to the case of real-valued features and threshold-based splitting rules, namely, $\mathbf{1}_{[x_i<\theta]}$. Such splitting rules yield decision stumps. The basic idea is to reduce the problem to the case of binary features as follows. Let $\mathbf{x}_1, ..., \mathbf{x}_m$ be the instances of the training set. For each real-valued feature i, sort the instances so that $x_{1,i} \leq ... \leq x_{m,i}$. Define a set of

thresholds $\theta_{0,i}, ..., \theta_{m+1,i}$ such that $\theta_{j,i} \in (x_{j,i}, x_{j+1,i})$ (where we use the convention $x_{0,i} = -\infty$ and $x_{m+1,i} = \infty$. Finally, for each i and j we define the binary feature $\mathbf{1}_{[x_i < \theta(j,i)]}$. Once we have constructed these binary features, we can run the ID3 procedure described in the previous section. It is easy to verify that for any decision tree with threshold-based splitting rules over the original real-valued features there exists a decision tree over the constructed binary features with the same training error and the same number of nodes.

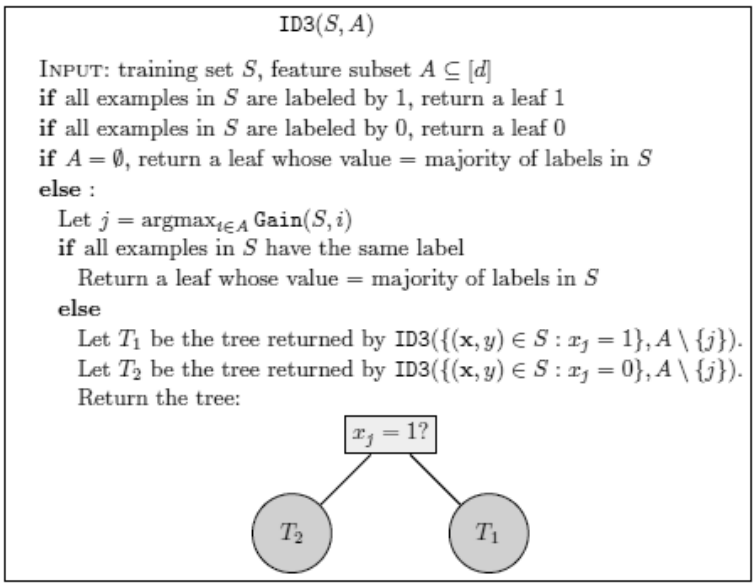


Figure 3.6: ID3 algorithm

Different algorithms use different implementations of $Gain(S; i)$. Here we present three. We use the notation $\mathbf{P}_S[F]$ to denote the probability that an event holds with respect to the uniform distribution over S, the training set.

Train Error: The simplest definition of gain is the decrease in training error. Formally, let $C(a) = min(a, 1-a)$. Note that the training error before splitting on feature i is $C(\mathbf{P}_S[y=1])$, since we took a majority vote among labels. Similarly, the error after splitting on feature i is

$$\mathbf{P}_S[x_i = 1]C(\mathbf{P}_S[y = 1|x_i = 1]) + \mathbf{P}_S[x_i = 0]C(\mathbf{P}_S[y = 1|x_i = 0]).$$

Therefore, we can define Gain to be the difference between the two, namely,

$$Gain(S, i) := C(\mathbf{P}_S[y = 1]) - (\mathbf{P}_S[x_i = 1]C(\mathbf{P}_S[y = 1|x_i = 1]) + \mathbf{P}_S[x_i = 0]C(\mathbf{P}_S[y = 1|x_i = 0])).$$

Information Gain: Another popular gain measure that is used in the ID3 and C4.5 algorithms of Quinlan (1993) is the information gain. The information gain is the difference between the entropy of the label before and after the split, and is achieved by replacing the function C in the previous expression by the entropy function,

$$C(a) = -a\log(a) - (1-a)\log(1-a).$$

Gini Index: Yet another definition of a gain, which is used by the CART algorithm of Breiman, Friedman, Olshen & Stone (1984), is the Gini index,

$$C(a) = 2a(1 - a).$$

Both the information gain and the Gini index are smooth and concave upper bounds of the train error. These properties can be advantageous in some situations (see, for example, Kearns & Mansour (1996)).

Another largely used algorithm to train Decision Trees, mentioned above, is Classification And Regression Tree (CART) [26]. The idea is quite simple: the algorithm first splits the training set in two subsets using a single feature k and a threshold t_k. It searches for the pair (k, t_k) that produces the purest subsets (weighted by their size). The cost function that the algorithm tries to minimize is given by:

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right};$$

where

- $G_{left/right}$ measures the impurity of the left/right subset,
- $m_{left/right}$ is the number of instances in the left/right subset.

Once it has successfully split the training set in two, it splits the subsets using the same logic, then the sub-subsets and so on, recursively. It stops recursing once it reaches the maximum depth, or if it cannot find a split that will reduce impurity.

Decision Trees make very few assumptions about the training data (as opposed to linear models, which obviously assume that the data is linear, for example). If left unconstrained, the tree structure will adapt itself to the training data, fitting it very closely, and most likely overfitting it (overfitting is the production of a model that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably). To avoid overfitting the training data, we need to restrict the Decision Tree's freedom during training, this is called regularization. The regularization hyperparameters depend on the algorithm used, but generally we can at least restrict the maximum depth of the Decision Tree.

Decision Trees are fairly intuitive and their decisions are easy to interpret. Such models are often called white box models. In contrast, Random Forests or neural networks are generally considered black box models. They make great predictions, and you can easily check the calculations that they performed to make these predictions; nevertheless, it is usually hard to explain in simple terms why the predictions were made.

3.2.4 K-means

K-means is one of the most used algorithm of the unsupervised learning tasks. Unsupervised algorithm can be divided in:

1. Dimensionality reduction: as many Machine Learning problems involve thousands or even millions of features for each training instance that make training extremely slow and make it much harder to find a good solution, so the aim is to reduce the number of features, turning an intractable problem into a tractable one.

2. Clustering: the goal is to group similar instances together into clusters. This is a great tool for data analysis, customer segmentation, recommender systems, search engines, image segmentation, semi-supervised learning, dimensionality reduction, and more.

3. Anomaly detection: the objective is to learn what "normal" data looks like, and use this to detect abnormal instances, such as defective items on a production line or a new trend in a time series.

4. Density estimation: this is the task of estimating the probability density function (PDF) of the random process that generated the dataset. This is commonly used for anomaly detection: instances located in very low-density regions are likely to be anomalies. It is also useful for data analysis and visualization.

For this project I used a clustering techniques, in fact, K-means is the most popular between them. The K-Means algorithm is a simple algorithm capable of clustering this kind of dataset very quickly and efficiently. It was proposed by Stuart Lloyd at the Bell Labs in 1957 as a technique for pulse-code modulation, but it was only published outside of the company in 1982. It is sometimes referred to as Lloyd-Forgy.

This type of clustering starts by defining a cost function over a parameterized set of possible clusterings and the goal of the clustering algorithm is to find a partitioning (clustering) of minimal cost. Under this paradigm, the clustering task is turned into an optimization problem. In K-means the data is partitioned into disjoint sets $C_1, ..., C_k$ where each C_i is represented by a centroid μ_i. It is assumed that the input set X is embedded in some larger metric space (X', d) (so that $X \in X'$) and centroids are members of X'. The k-means objective function measures the squared distance between each point in X to the centroid of its cluster. The centroid of C_i is defined to be

$$\mu_i(C_i) = \operatorname*{argmin}_{\mu \in X'} \sum_{x \in C_i} d(x, \mu)^2.$$

Then, the k-means objective is

$$G_{k-means}((X, d), (C_1, ..., C_k)) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu(C_i))^2.$$

This can also be rewritten as

$$G_{k-means}((X, d), (C_1, ..., C_k)) = \min_{\mu_1, ..., \mu_k \in X'} \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu)^2.$$

The k-means objective function is quite popular in practical applications of clustering. However, it turns out that finding the optimal k-means solution is often computationally infeasible (the problem is NP-hard, and even NP-hard to approximate to within some constant). As an alternative,

the simple iterative algorithm reported below. We describe the algorithm with respect to the Euclidean distance function $d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||$.

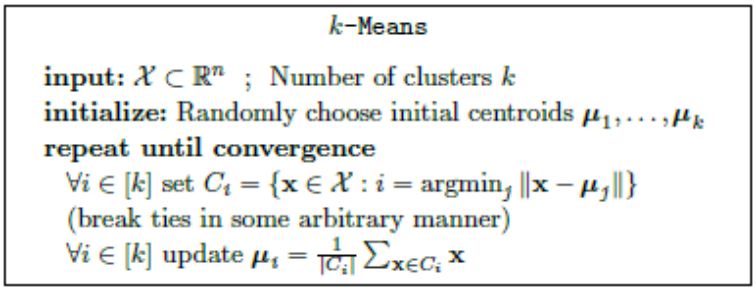


Figure 3.7: K-means algorithm

An important improvement to the K-Means algorithm, called K-Means++, was proposed in a 2006 paper by David Arthur and Sergei Vassilvitskii: they introduced a smarter initialization step that tends to select centroids that are distant from one another, and this makes the K-Means algorithm much less likely to converge to a suboptimal solution. They showed that the additional computation required for the smarter initialization step is well worth since it makes it possible to drastically reduce the number of times the algorithm needs to be run to find the optimal solution. Here is the K-Means++ initialization algorithm:

- Take one centroid μ_i, chosen uniformly at random from the dataset.
- Take a new centroid μ_i, choosing an instance $\mathbf{x}_i$ with probability: $D(\mathbf{x_i})^2 / \sum_{j=1}^{n} D(\mathbf{x}_j)^2$ where $D(\mathbf{x}_i)$ is the distance between the istance $\mathbf{x}_i$ and the closest centroid that was already chosen. This probability distribution ensures that instances further away from already chosen centroids are much more likely be selected as centroids.
- Repeat the previous step until all k centroids have been chosen.

Despite its many merits, most notably being fast and scalable, K-Means is not perfect. It is necessary to run the algorithm several times to avoid sub-optimal solutions, plus you need to specify the number of clusters. Moreover, K-Means does not behave very well when the clusters have varying sizes, different densities, or non-spherical shapes.

Chapter 4

Data processing

In this chapter we will firstly describe the technologies adopted and the files structure. Then we will go through all the phases of data pre-processing in order to prepare the dataset for Machine Learning operations.

4.1 Development environment and files architecture

This project is developed in Python, an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development. Python's simple syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

In particular, I used Anaconda, a free and open-source distribution of Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The principal files were created with Jupyter notebook (*.ipybn*). In this case, "notebook" denote documents that contain both code and rich text elements (such as figures, links, equations, etc.). Because of the mix of code and text elements, these documents are the ideal place to bring together an analysis description, and its results, as well as, they can be executed to perform the data analysis in real time. The Anaconda package allow to download all the basics python libraries such as Numpy, Scipy, Matplotlib, Pandas, Scikit-learn and many others. All the listed libraries were used for the project and some others:

- Numpy: adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- Scipy: used for scientific computing and technical computing. It contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image

processing, ODE solvers and other tasks common in science and engineering.

- Matplotlib: plotting libary that provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits.
- Math: contained in the base installation of Python, provides access to some common math functions and constants, which we can use throughout our code for more complex mathematical computations.
- Pandas: a software library written for for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- Datetime: a module that supplies classes for manipulating dates and times in both simple and complex ways.
- Scikit-learn: is a library in Python that provides many unsupervised and supervised learning algorithms [27]. It features various classification, regression and clustering algorithms and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.
- Csv: module used to import or export spreadsheets and databases.
- PyDotPlus: is an improved version of the old pydot project that provides a Python Interface to Graphviz's Dot language (Graphviz is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks).

The list above contains all the libraries used in this project and their description. Now, taking into consideration the file structure, as already mentioned, I divided the project in: "Dataset creation" and "Machine Learning application". In this chapter we will focus only on the first part.

"Dataset creation" is composed by two files: *Function.py* and *PDTA_datasetCreation.ipynb*. The first one is imported in the second as it contains the definitions of the functions used to create and tag the dataset. The second one receives as input a reduced version of the starting dataset decribed in the previous chapter and output three different files: *Tagged_Dataset2.csv*, the binary classified dataset, *Tagged_Dataset3.csv*, the multi-class dataset, and *Dataset_notag.csv* previous vectors without any tag.

The input vector is slightly different from the described one as I kept only patients with glycated haemoglobin and glycated haemoglobin index exams. In fact, at the beginning I thought I had to consider both the exams in the classification but, as their meaning is almost the same and so the results, lately I decided to use only the first one. Moreover I performed a sort of homogenization of pathology codes as most of them identified the same problem but differentiate from each other by a simple letter or number (e.g. a 0 before or after the right code).

We can now focus on the description of the process performed to obtain the final outputs.

4.2 Vectors creation

In the second file mentioned above, the reduced dataset is loaded with *pandas.read_ csv()* that receives as input the *.csv* file containing the dataset, the separator symbol between values, the symbol for quotations identification and the encoding to use for UTF when reading. I printed the first five rows, extracted and saved the minumum and maximum values (4.8 and 6.5) for the glycated haemoglobin results directly from the dataset.

Then the dataset is given as input to the method *vector_ creation1()* defined in the *Function.py* file. This is a first elaboration useful to create a unique vector for patient, complete and without repetition. However, this is not the last one as this is only a temporary construction useful for the final step. In fact, the new elaborated dataset is given as input to the second method *vector_ creation2()*, always defined in *Function.py*. This method returns the final set of vectors together with an auxiliary set to save the exam dates in "datetime" format and patient codes useful only to plot the result that I reported at the end of the chapter. The final dataset without tag is saved with *csv.writer()* and *writerows()*.

We will see in detail the operations performed by the two "creation methods" and how the final vector per patient is composed.

4.2.1 Temporary vector

As said above, the first method receives as input the reduced dataset. It was reduced in a first time using simple tools of Mycrosoft Access and then more complicated elaboration were done with these passages. So, initially each row presented:

1. Patient ID;
2. Exam date;
3. Exam code: only 1235 for Glycated haemoglobin and 4348 for Glycated haemoglobin index exams;
4. Numerical result of the exam;
5. Minimum value of the numerical result;
6. Maximum value of the numerical result;
7. Pathology code;
8. Pathology description;

We have a different row for each exam or each pathology as already explained, and in this step the target was to build a unique vector for each patient. Considering that the row are ordered by patient ID and, for each patient, by exam dates (with increasing order), we were able to perform our tasks rapidly and with linear complexity.

In fact the method run a while cycle for the length of the dataset and save the first patient ID encountered; then it performs a second while cycle that runs on the rows checking if the new patient ID is equal to the current one under analysis. The second while cycle will last at most 32 turns, the maximum number of exams taken by patients, and stops when it finds a patient ID different from the one under analysis.

So if the code encountered correspond to the one we are analyzing, firstly the exam date is elaborated using *datetime.strptime()* that transforms a string to a "datetime" object and receives as input the "string date" and the format to read it (e.g. year, month, day). Now I can check if the date is not already in the exams list and if the exam code is equal to 1235 (glycated haemoglobin). If these conditions are respected I will update the counter of exams taken by that patient and save the date and the numerical result in the proper lists. In fact, the algorithm build three different list to temporary collect the exam results, the exam dates and the pathologies.

After this I load the pathology code and, even for it, the algorithm checks if it is already present in the list assigned. If not it adds it together with its description. When we encounter a new patient the second while loop ends and saves the values found until now in the temporary vector, in the format:

1. Patient ID;
2. Number of Glycated haemoglobin analysis taken during the examined years;
3. A list of pathology codes and their descriptions;
4. A list of exam dates in chronological order;
5. A list of numerical results results in chronological order.

After all patients are elaborated the algorithm stops and return the new constructed dataset. Below the representation of a row.

Patient ID	N° of exams	Pathologies [...]	Exam dates [...]	Exam results [...]

Figure 4.1: Temporary vector

4.2.2 Final vector

The previous set of vectors is given as input to the second method that for each line checks if the second element (the number of exams) is strictly greater than two and if the code "013" is in the pathologies list. That code is the one representing patient affected by diabete that we want to examine. If these conditions are respected the algorithm saves the patient ID in a new vector that will form the final dataset and in a dates vector useful for plotting the exam trend.

Then I took into consideration the last three exam dates from the list and computed, in order, the difference from the first and the second, the second and the third, the third and the last day

available for exams (01/08/2017). These differences are computed in number of days and saved in the line, in chronological order after patient ID, only after the division by 365, the days in a year. So, the values are in a continuous format in the interval $[0, 4]$. In the dates dataset all the exam dates are inserted keeping the "datetime" format.

The next step was to consider the exam results and put them in the vector in chronological order after the date differences. This part ends the vector creation, then the algorithm returns the vector of exam dates and the final vector useful for machine learning computations in the format:

1. Patient ID;
2. difference in years between first and second exam;
3. difference in years between second and third exam;
4. difference in years between third exam and last possible date;
5. numerical result of first exam;
6. numerical result of second exam;
7. numerical result of third exam.

This is done for every row, below its representation.

Patient ID	Exams difference 1	Exams difference 2	Exams difference 3	Exam result 1	Exam result 2	Exam result 3

Figure 4.2: Final vector

4.3 Vectors tagging

After the final vector was created the script starts the tagging phase. These part should be "of human interest" as it is difficult to highlight only linear rules in the identification of patients that are following or not an appropriate therapy. However, as already explained, with the help of an Expert in diabete pathologies and treatments, we pointed out a first concrete set of rules in order to proceed with tests waiting for more proper datasets, useful for our targets.

As already described, our goal is to classify patients in two group, a binary classification that identifies with "0" the patients that are following properly the correct therapy and with "1" the patients that present an inadequate therapy or are not following it. I started putting in an algorithm format the rules exposed in the previous chapter. Then analysing the results of this first process we found some supposed extreme case that deserved better consideration, like patient with all exams in the limits but date differences outside them or patient "in improvement". So we started adding more precise rules, looking also at the plots of patients trend, that will be described in the following

subsection. Moreover, we decide to try the introduction of a new intermediate category, to see if assigning borderline patient to a different category could improve machine learning results.

With this addition, the "0" and "2" tags will represent previous categories, in order, while "1" identifies patients with improving results and that need more time to be correctly judged in the first two sets, or patients that are following the correct therapy properly but that need to keep more attention on their exam dates. This division go against the idea of definitive classification but describes a more human reasoning. We will discuss in the result section the differences brought by multi-classification on patient analysis.

So, the final dataset previously described is given as input to two tagging method one for the binary classification and the other for the three classes one, together with the limits of numerical results. After this we obtain two tagged datasets, both saved thanks to *csv.writer()* and *writerows()*. I printed also the number of element in each class and some example plot that will be reported at the end of this chapter. Now we can focus on the operations performed by the methods.

4.3.1 Binary tag

We can divide the algorithm in two parts: checking if the established rules are respected in the vector and tagging with the proper number using the previous results. So, for each line it checks if the last two exams are in the limits, in $[4.8, 6.5]$, plus/minus a variance, set to 0.3, and saves it in a boolean variable. Then it searches if the three exams are decreasingly approaching the upper limit, with last exams almost in it; or if they are increasing towards the lower limit, with last exam almost in it. Both results are saved in a boolean variable.
Finally I considered also the exam dates, checking how many of them were taken before a year plus some variance, approximately two months, saving this number on a variable. I do the same for exams taken before three month less a variance, almost a month, the number of them is saved on a variable but without considering the last difference as if last exam were taken nearly the last useful date the value will be obviously small but not considerable.

Now the tagging process uses the variables to examine rules and assign the tag:

- "0" if all exams were taken within a year, at least one exams after 3 months and last two exams are in the limits;

- "0" if all exams were taken within a year, all the exams after at least 3 months and results are decreasing/increasing towards upper/lower limit;

- "1" elsewhere, so if any exam was taken with more than a year distance and/or last exams are out of limits and/or non-decreasing/increasing trend, etc.

This method returns 200 patients of type "1" and only 110 of type "0". At the end of the chapter I will report some graphic examples with the vector completed with tag as title.

4.3.2 Multi tag

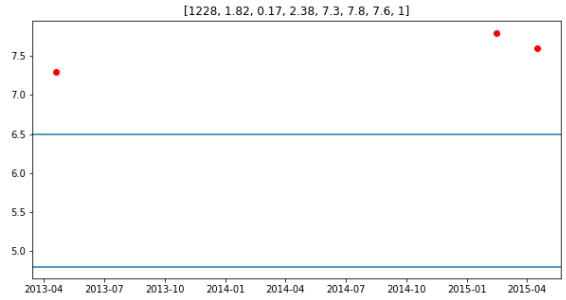
As before, I divided the process in two parts saving all the results in variables that will form the tagging process. The rules examined in the first phase are the same described above with a single addition. In fact, I also checked if the last two exams are in the limit plus/minus a more high variance, fixed to 0.6, with last exam less or equal to the second last exam, and this is saved in boolean variable. This is done to give more weight to patient with last results almost stable even if a bit too much over the limits. Then I assigned the tags:

- "0" if all exams were taken within a year, at least two exams after at least 3 months and last two exams are in the limits;
- "1" if all exams were taken within a year, at least two exams after at least 3 months and results are decreasing/increasing towards upper/lower limit on in the larger limits;
- "1" if in large or strict limits and exams are taken with more than a year between them or less than 3 months;
- "2" elsewhere, so if any exam was taken with more than a year distance and/or last exams are out of limits and/or non-decreasing/increasing trend, etc.

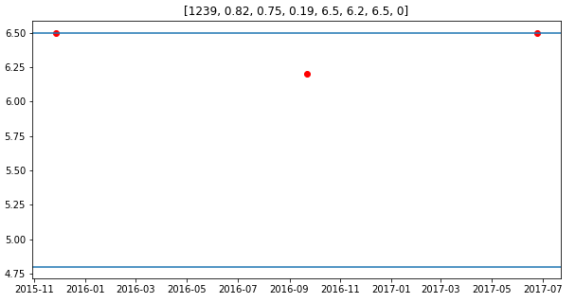
This method returns 129 patients of type "2", 100 of type "0" and 81 of type "1".

Below (Figure 4.3 and Figure 4.4) I report the plots of four patient trends of the exams analyzed. They are the plot of the same patients but with both binary and multi-class tag. On the horizontal axis we find the exam dates and on the vertical one their numerical results (red dots). In blue I highlighted also the limits of these results.

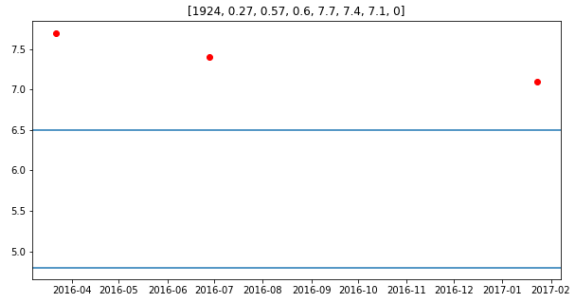
As it is visible, the first two vector congruently remain classified as the extremes both in binary and multi classification. While the last two change from previous binary categories ending in the mid category, after multi-classification. In fact, we see that the third example presents improving values but not enough to be classified as patient following the correct therapy. On the other hand the forth example shows results contained in the limits plus a large variance that implies that the patient is containing the problem but should be kept under control however it does not deserve a classification in the other categories.



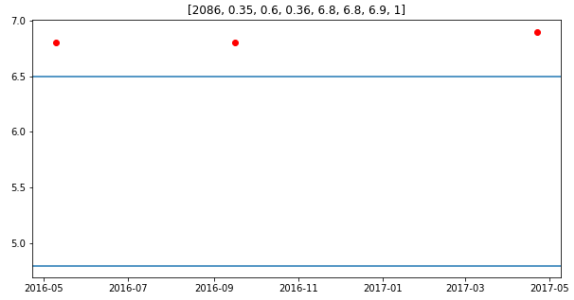
(a)



(b)

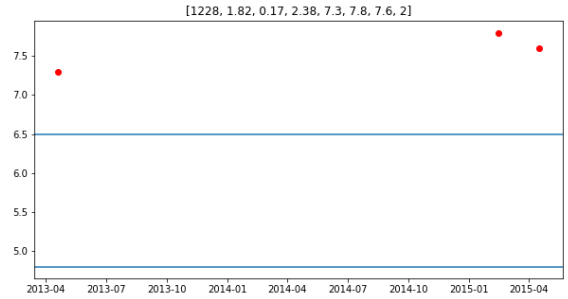


(c)

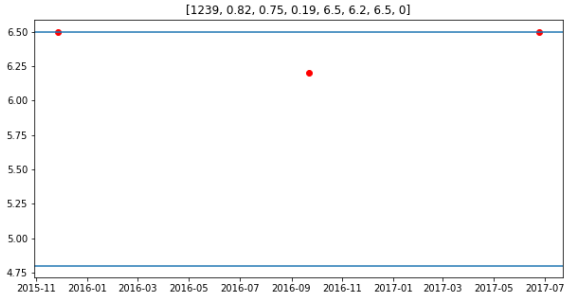


(d)

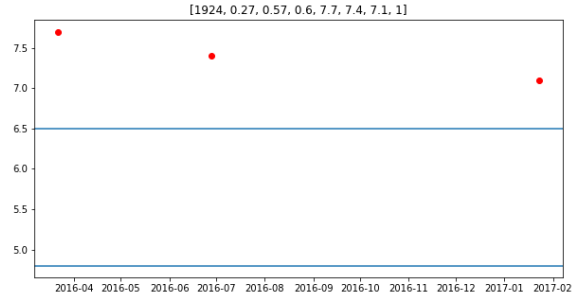
Figure 4.3: In (a), (b), (c) e (d) I plotted four patient graphs with the exam dates on the x-axis and the results on the y-axis. The last position in the title represent the binary tag assigned [0; 1].



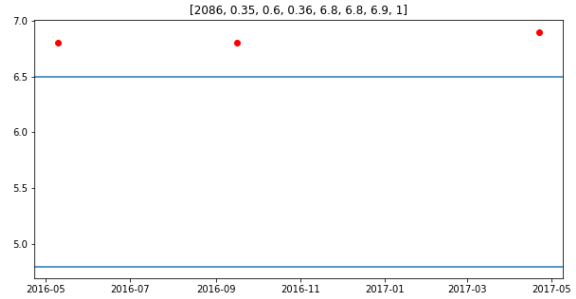
(a)



(b)



(c)



(d)

Figure 4.4: In (a), (b), (c) e (d) I plotted four patient graphs with the exam dates on the x-axis and the results on the y-axis. The last position in the title represent the multi-class tag assigned [0; 1; 2].

Chapter 5

Development and Results

In this chapter, I will describe the code used for the machine learning tests and the functioning of the different strategies adopted, analysing the parameters chosen and the numerical results obtained.

5.1 Code description

For the pre-processing part, I divided the computations for binary and multi-tag classification. This time I produced two different scripts that perform the same operations, but on the two opposite datasets.

As they are exactly the same we can focus on the binary one and the description will comprehend also the second case. Firstly I had to import all necessary classes, those listed above in the thesis, and the specific tools from *Sklearn*, one for each machine learning technique I used.

Then I loaded the dataset without tag, together with the binary one (or multi tags): 310 vectors of 6 features, not considering the patient code. After that, I concentrated firstly on the clustering part using the Kmeans algorithm to see if the groups underlined by this method respect the classification I made in the pre-processing part. To see this, I checked how many vectors are classified correctly by the clustering algorithm with respect to the tagged dataset, printing the percentage of correct matches.

Then I moved to the supervised methods. Initially I split the features vectors and their tags into training and validation sets, the first composed by the 2/3 of the data and the second by the remaining 1/3 of them. This division is done to find the best parameters considering not only the training but also a pseudo-test set.

In fact, before computing the final results with the best classifiers, I made some trials changing the parameters of each algorithm. To do so, I used *GridSearchCV* [28]. This tool performs an exhaustive search over specified parameter values for an estimator. We considered Hyper-parameters: parameters that are not directly learnt within estimators. In *scikit-learn* they are passed as arguments to the constructor of the estimator classes. It is possible and recommended

to search the hyper-parameter space for the best cross validation score. Any parameter provided when constructing an estimator may be optimized in this manner. Two generic approaches to sampling search candidates are provided in *scikit-learn*: for given values, GridSearchCV exhaustively considers all parameter combinations, while RandomizedSearchCV can sample a given number of candidates from a parameter space with a specified distribution. The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid. As said above we considered only the first one who implements the usual estimator API: when "fitting" it on a dataset all the possible combinations of parameter values are evaluated and the best combination is retained.

After the perfect values where found with the use of *GridSearchCV* and some trials done by hand, I trained the different classifiers, Support Vector Machine, Multi-layer Perceptron, and Decision tree, on the training set printing the best error (one minus the accuracy). Then the last step was to predict the tags of the validation set and print, even here, the best error.

5.2 Parameters choice

Now we concentrate on the parameters chosen in each method together with the results in terms of accuracy on training and validation tests. I divided the description for each machine learning algorithm starting from the unsupervised class.

5.2.1 Clustering

The k-means algorithm searches for a pre-determined number of clusters within an unlabeled multidimensional dataset. It accomplishes this using a simple conception of what the optimal clustering looks like: the "cluster center" is the arithmetic mean of all the points belonging to the cluster, and each point is closer to its own cluster center than to other cluster centers. Using the *Sklearn* method [29] we had to specify different parameters, below the most important ones:

1. The number of clusters to form, as well as the number of centroids to generate. I tested the algorithm for 2 and 3 clusters, congruently with the target presented in the thesis.

2. Method for initialization:

   - 'k-means++' : selects initial cluster centers for k-mean clustering in a smart way to speed up convergence.
   - 'random': choose k observations (rows) at random from data for the initial centroids.

3. Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of x consecutive runs in terms of inertia.

4. Maximum number of iterations of the k-means algorithm for a single run.

5. Precompute distances:

- 'auto' : do not precompute distances if number of samples * number of clusters > 12 million. This corresponds to about 100MB overhead per job using double precision.
- True : always precompute distances (faster but takes more memory).
- False : never precompute distances.

In this case I didn't use *GrisSearchCV* for the choice of parameters as it was easy to see that their changing didn't modify a lot the results. So, for my computations, I chose k-means++ method, 1000 maximum iterations and the default values in the other cases. It was easy to notice that the real change was made by the format of data given in input. In fact, the dataset described in the previous chapters leaded to poor result. Then, I modified it by normalization.

I tested different normalization, L1 norm L2 norm and Uniform norm, in order:

$$|\mathbf{x}|_1 = \sum_{r=1}^{n} |x_r| \,;$$

$$|\mathbf{x}|_2 = (\sum_{r=1}^{n} x_r^2)^{1/2};$$

$$|\mathbf{x}|_\infty = \max_{r=1,\ldots,n} (|x_r|).$$

The one that gave the best results was L1, which lead to a percentage of "accuracy" of 55,8% on the dataset divided in two groups and 33% for the division in three groups. In this case the accuracy is computed as the ratio between the number of correct classifications (done by the clustering algorithm with respect to the tags defined) and the total number of vectors/patients.

This results are quite poor but it is understandable as our tagging process is not perfect and should be carried by hand by more Experts. However from these outcomes we can made another consideration: the binary classification is more adapt for this dataset then the multi-class one. So even if the second one seems proper from my point of view, our initial approach was better.

From now on I will report the results for both classifications, binary and multi-class, but I will concentrate deeper in the first one, reporting also some interesting graphs.

5.2.2 Support Vectror Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. The main objective is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin. The porpouse is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal hyperplane in the following steps: firstly generate hyperplanes which segregates the classes in the best way. Then select the right hyperplane with the maximum segregation from the either nearest data points. Some problems can't be solved using linear hyperplane, so SVM uses a kernel trick to transform the input space to a higher dimensional space, better explained in the second chapter together with a more accurate description of SVM mechanism.

Support vector machines presents many andvantages such as it is effective in high dimensional spaces, also in cases where number of dimensions is greater than the number of samples, it is memory efficient, using a subset of training points in the decision function, and versatile, different Kernel functions can be specified for the decision function.
However they include some disadvantages: SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation, and if the number of features is much greater than the number of samples avoid over-fitting in choosing Kernel functions and regularization term is crucial.

Before talking of the parameters chosen it is better to present the most important variables proposed by the method [30]:

1. Kernel : Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (number of samples, number of samples).

2. Tolerance for stopping criterion.

3. Penalty parameter C of the error term.

4. Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

5. Gamma: Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. Current default is 'auto' which uses 1/number of features, if gamma='scale' is passed then it uses 1/(number of features * X.var()) as value of gamma.

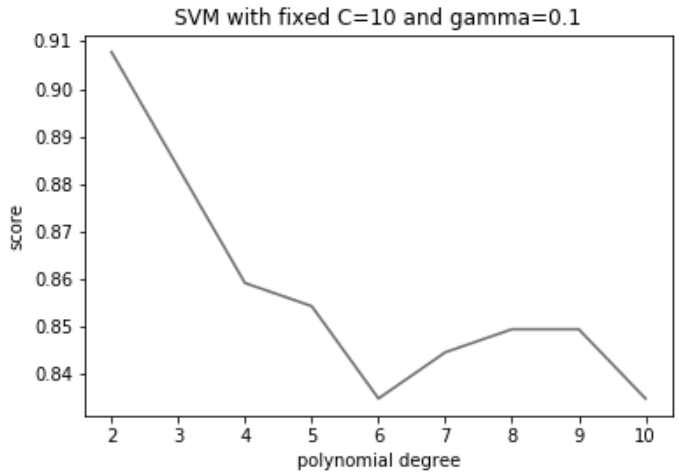


Figure 5.1: SVM results changing degree

I tested all the Kernel possibilities changing the last three parameters of the list with the use of *GrisSearchCV*, as said above. Best results, in terms of mean accuracy on the validation set,

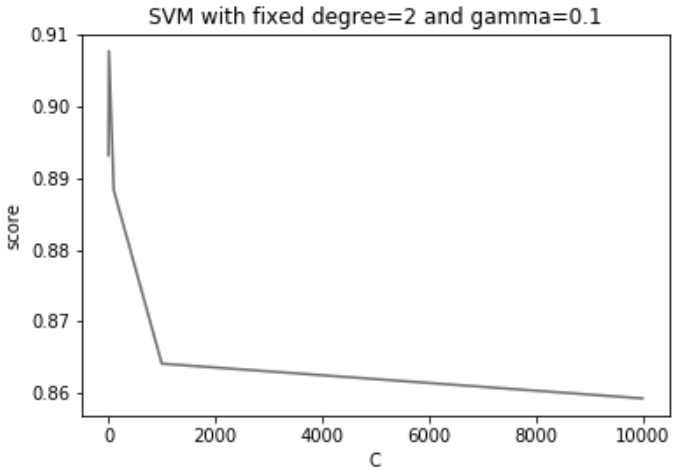


Figure 5.2: SVM results changing C

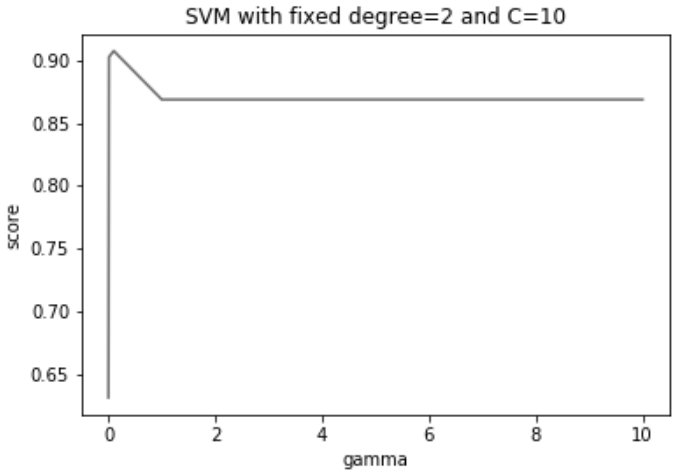


Figure 5.3: SVM results changing gamma

were found with the use of polynomial kernel, so I report above some graphs showing the scores changing due to parameters modification for the best SVM.

So the best SVM for binary classification was the polynomial one with final coefficient of degree 2, C 10 and gamma 0.1 with a result in training error of 0.077 and in validation error of 0.096. These outcomes are not perfect but considering that we have a small dataset to train we can consider them quite good. For the three class classification the results are different and obviously lower in terms of accuracy. The final model is rbf kernel with C 10 and gamma 0.1 that gave a training error of 0.11 and a validation error of 0.18.

5.2.3 Multi-layer Perceptron

As already explained, Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function $f(\cdot) : \mathcal{R}^m \rightarrow \mathcal{R}^o$ by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features $\mathcal{X} = x_1, x_2, ..., x_m$ and

a target y, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that, between the input and the output layer, there can be one or more non-linear layers, called hidden layers.
Even for Multi-layer Perceptron there are different advantages, such as: the capability to learn non-linear models and to learn models in real-time (on-line learning) using partial-fit.
But some disadvantages need to be mentioned: MLP with hidden layers have a non-convex loss function where there exists more than one local minimum, therefore different random weight initializations can lead to different validation accuracy, and it is sensitive to feature scaling.
Now we can consider MLP parameters form the method [31], below the most important ones:

1. Hidden layer sizes : length = number of layers - 2. The i^{th} element represents the number of neurons in the ith hidden layer.

2. Activation function for the hidden layer.

   - 'identity', no-op activation, useful to implement linear bottleneck, returns $f(x) = x$.
   - 'logistic', the logistic sigmoid function, returns $f(x) = 1/(1 + exp(-x))$.
   - 'tanh', the hyperbolic tan function, returns $f(x) = tanh(x)$.
   - 'relu', the rectified linear unit function, returns $f(x) = max(0, x)$.

3. The solver for weight optimization.

   - 'lbfgs' is an optimizer in the family of quasi-Newton methods.
   - 'sgd' refers to stochastic gradient descent.
   - 'adam' refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba [32].

4. Maximum number of iterations. The solver iterates until convergence or this number of iterations. For stochastic solvers ('sgd', 'adam'), note that this determines the number of epochs (how many times each data point will be used), not the number of gradient steps.

In this case I concentrated on the hidden layer size as it was easy to show that the best model is given by MLP with lbfgs as solver, identity as activation function and default values in other cases. Below the graph showing the variations of scores, on validation data, in function of the hidden layer size.

As the score results were identical for a single hidden layer with 10, 50 and 100 neutrons I chose the most simple one: $(10,)$ hidden layer size. The final model gave a training error of 0.092 and a validation error of 0.11. This result are lower than the ones obtained with polynomial SVM, as in general neural networks works better with a great amount of data and features, but we dispose of only 310 vectors of 6 features. Even here the result for the second type of classification are lower in accuracy but the best parameters found are the same of the binary case. So training errors in this case is 0.18 and the validation one is 0.22.

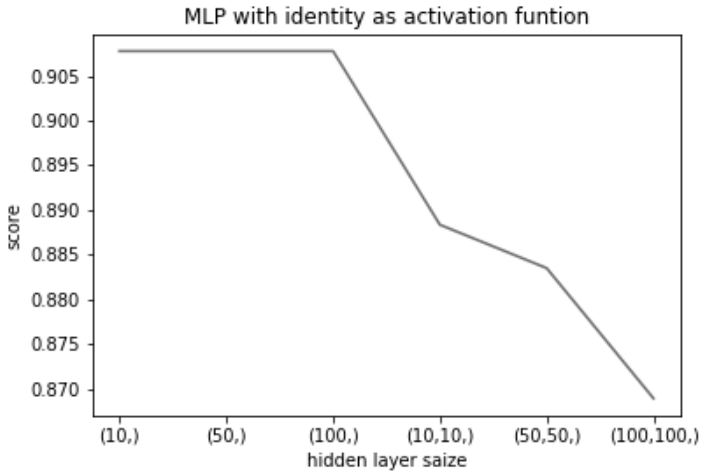


Figure 5.4: MLP results changing hidden layer size

5.2.4 Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
Some advantages of decision trees are:

- Simple to understand and to interpret.
- Requires little data preparation.
- The cost of using the tree is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data.
- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

However they present also disadvantages, such as:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated.

- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

As already said in the third chapter, there exists various decision tree algorithms. Above in the thesis we described the functioning of ID3 and CART. *Scikit-learn* uses an optimised version of the CART algorithm; however, *Scikit-learn* implementation does not support categorical variables. The most important Decision tree parameters that we can modify with the method considered [33] are:

1. Criterion: the function to measure the quality of a split. Supported criteria are 'gini' for the Gini impurity and 'entropy' for the information gain.
2. Splitter: the strategy used to choose the split at each node. Supported strategies are 'best' to choose the best split and 'random' to choose the best random split.
3. The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than minimum samples split.
4. Max features: the number of features to consider when looking for the best split:
   - If int, then consider max number of features at each split.
   - If float, then max number of features is a fraction and $int(maxnumberoffeatures * numberoffeatures)$ features are considered at each split.
   - If 'auto', then $maxfeatures = sqrt(numberoffeatures)$.
   - If 'sqrt', then $maxfeatures = sqrt(numberoffeatures)$.
   - If 'log2', then $maxfeatures = log2(numberoffeatures)$.
   - If None, then $maxfeatures = numberoffeatures$.
5. Min impurity decrease : a node will be split if this split induces a decrease of the impurity greater than or equal to this value.

   The weighted impurity decrease equation is the following:

   $$N_t/N * (impurity - N_{t,R}/N_t * rightimpurity - N_{t,L}/N_t * leftimpurity);$$

   where N is the total number of samples, N_t is the number of samples at the current node, $N_{t,L}$ is the number of samples in the left child, and $N_{t,R}$ is the number of samples in the right child.

   N, N_t, $N_{t,R}$ and $N_{t,L}$ all refer to the weighted sum, if sample weight is passed.

Even in this case we concentrated only on the maximum depth value comparing the results with *GrisSearchCV*, with best result equals to 8, while I kept the other parameters equals to: gini for the split criterion, best for the strategy, default values elsewhere. This combination leads to the best training error of 0 and validation error 0.1 in the binary case. For the second case of classification the results are 0.063 for the training error and 0.23 for the validation one.

As already said Decision tree classification is a white box algorithm so we can represent the classification process with the use of *Graphviz*. At the end of the chapter I reported the graph showing the tree built by the training procedure, in the binary case, that we can compare to the rules established in the pre-processing part to tag the vectors.

The graph starts from zero depth were we see the first criterion of choice according to which a sample take the right or the left branch. The gini value represent the purity of the node, so if all the samples in the node belong to a single class this value would be 0.

Now we could examine the route to each leaf of the tree but, as the tree present 8 layers and 22 leafs, I will considerate only the ones with the major number of samples in them.
For class 1 we find the majority of the samples following the paths:

1. result three higher than 7.15;
2. result three between 6.85 and 7.15 and result one less than 7.55;
3. result three less than 6.85 but date difference three bigger than 1.28.

It is visible that the first and the third routes are coherent with previous criteria of classification with last result over 6.5 plus a bias and last exams taken after a year difference. Instead the second path seams quite unusual with respect to the rules established as a criterion is to have the first result over a certain bound. However it could be considered as the check for the decreasing trend in the results, even if it is not so precise, not considering second result and the date difference.
For class 0 we find the majority of the samples following a single path:

1. result three less than 6.85, date difference three less than 0.95, result two less than 6.85, date difference one less than 1.24 and date difference two between 0.095 and 1.305.

In this case the choice is quite similar to the process considered in the previous chapter where last two results are less than the upper bound plus a bias and exams are taken within a year.

In the end we can say that the paths underlined by this classification method are consistent with the rules pointed out with the Expert and applied in the tagging phase. Even the training and validation errors for the binary case are acceptable, but SVM seems to perform better on the validation set.

5.3 Numerical results

Below I reported in two tables the results of training errors using the best models on the total dataset described in the third chapter, for binary and multi-class classification. The results reported

on the test set were obtained using the patient deleted in the first part of dataset creation: patient with at least three exams of glycated hemoglobin but without the diagnosis of diabete, so that are not following a therapy, ideally.

With these prerogatives the new test created seams a bit meaningless to the classification we are facing. However if we select from them those who made periodical exams and tag them with the algorithm built from the Expert's help and descried in previous chapter, the result is a set of almost 300 samples equally divided among two or three classes. Finally, even if these results will have some limitations, we can use the "new dataset" as a test set to confirm our model. Obviously, as described in the machine learning introduction the test set can not be used for the training and validation phase, it has to be completely new to the process, so I used it only after the final choice of parameters.

Below I reported the results obtained in a table.

Binary case	Training error	Test error
SVM	0.07	0.18
MLP	0.1	0.2
DT	0.003	0.099

Table 5.1: Binary classification results

Multi-class case	Training error	Test error
SVM	0.11	0.18
MLP	0.07	0.19
DT	0	0.2

Table 5.2: Multi-label classification results

As it is visible the best result is obtained with Decision tree for binary classification both on the training and the test set. Both Decision tree and SVM classifiers output good results and perform better on the binary case as already seen in the parameters choice and coherently with the outcomes of the Kmeans method.

On the other hand, MLP gives really poor result in binary classification and improves only in multi-class case. But as already said MLP works better on big datasets.

Other useful results to examine are Receiver Operating Characteristic (ROC) and Precision-Recall metrics, both of them evaluate classifier output quality.
AUC–ROC curve is the model selection metric for bi–multi class classification problem. ROC tells us how good the model is for distinguishing the given classes, in terms of the predicted probability.

The area covered by the curve is the area between the blue line (ROC) and the axis, visible below in the graphs. This area covered is called AUC. The bigger the area covered, the better the machine learning models is at distinguishing the given classes. Ideal value for AUC is 1.

To better understand this concept we have to introduce some definitions. There are two types of errors that can be identified here:

1. Type 1 Error: The model predicted the instance to be a Positive class, but it is incorrect. This is False Positive ($\mathcal{F}_p$);

2. Type 2 Error: The model predicted the instance to be the Negative class, but is it incorrect. This is False Negative ($\mathcal{F}_n$);

remembering:

- $\mathcal{T}_p$ = True Positive: the model predicted the positive class correctly, to be a positive class.
- $\mathcal{T}_n$ = True Negative: the model predicted the negative class correctly, to be the negative class.

So, a typical ROC curve has False Positive Rate (FPR) on the X-axis and True Positive Rate (TPR) on the Y-axis. Where:

$$TPR = \mathcal{T}_p/(\mathcal{T}_p + \mathcal{F}_n);$$

and

$$FPR = \mathcal{F}_p/(\mathcal{F}_p + \mathcal{T}_n).$$

Now we can explain also the concept of Precision-recall curve: it shows the trade-off between precision and recall for different threshold. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results labeled correctly.

Precision ($\mathcal{P}$) is defined as the number of true positives ($\mathcal{T}_p$) over the number of true positives plus the number of false positives ($\mathcal{F}_p$):

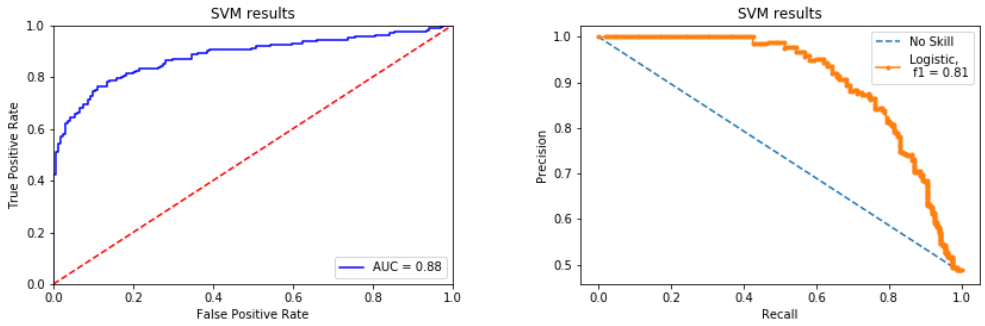
$$\mathcal{P} = \frac{\mathcal{T}_p}{\mathcal{T}_p + \mathcal{F}_p}.$$

Recall ($\mathcal{R}$) is defined as the number of true positives ($\mathcal{T}_p$) over the number of true positives plus the number of false negatives ($\mathcal{F}_n$), as the True Positive Rate:

$$\mathcal{R} = \frac{\mathcal{T}_p}{\mathcal{T}_p + \mathcal{F}_n}.$$

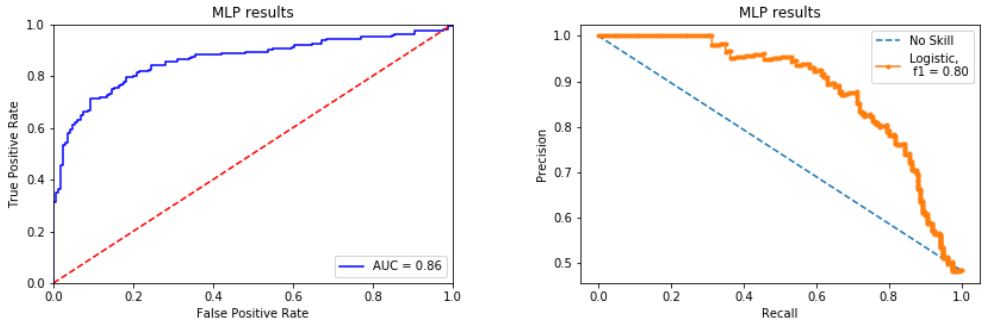
These quantities are also related to the ($\mathcal{F}_1$) score, which is defined as the harmonic mean of precision and recall:

$$\mathcal{F}_1 = 2\frac{\mathcal{P} \times \mathcal{R}}{\mathcal{P} + \mathcal{R}}.$$



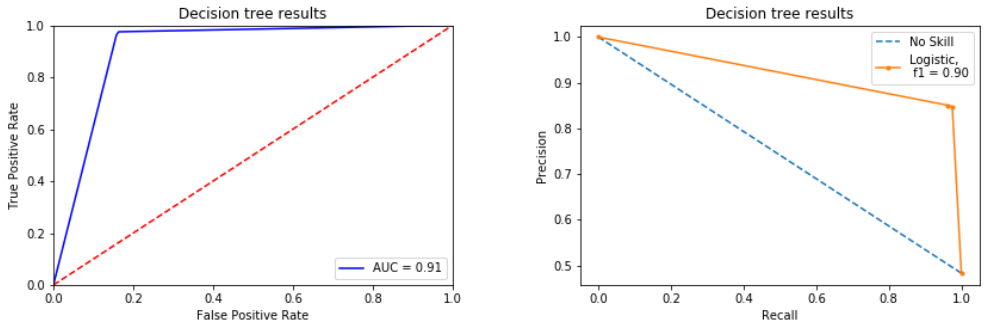
(a) (b)

Figure 5.5: SVM ROC and precision/recall curves



(a) (b)

Figure 5.6: MLP ROC and precision/recall curves



(a) (b)

Figure 5.7: Decision tree ROC and precision/recall curves

Above I reported Roc and Precision-Recall curves for all the supervised algorithms used, only in the binary classification case, because, as already seen, it gave us the most accurate results. As we can see, the obtained results seem better with the use of Decision tree classifiers, in fact AUC is equals to 0.91, almost 0.1 point over the other models, and also in the precision-recall case we obtained $\mathcal{F}_1 = 0.9$ which is the higher result.

Finally we can say that the best model tested is Decision tree classification with gini as split criterion, best as strategy and 8 as maximum depth value.

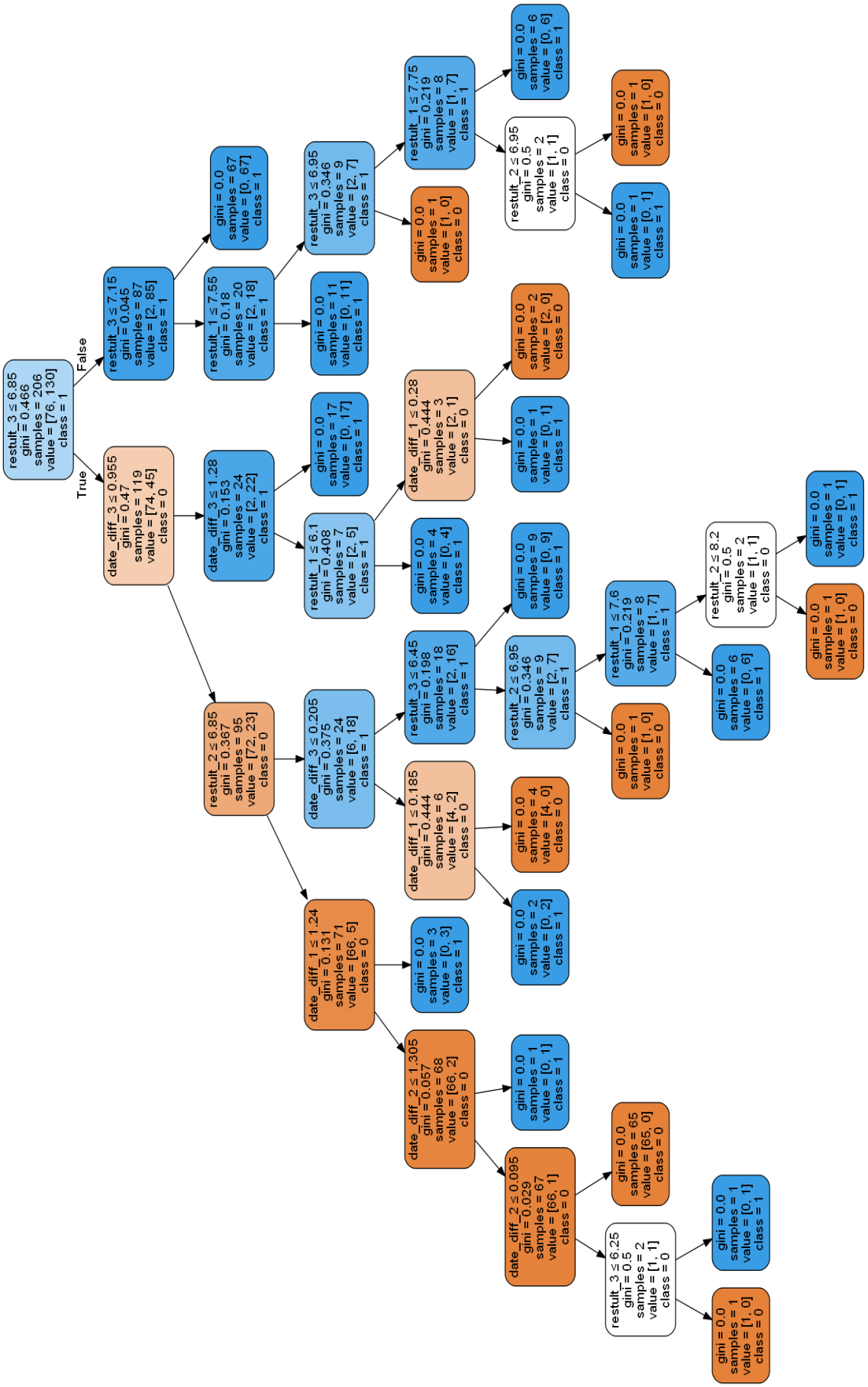


Figure 5.8: Binary decision tree representation

Chapter 6

Conclusions

The results obtained with these first tests are in the order of 10^{-1} or 10^{-2}, while results considered optimum in terms of percentage of test errors are in the order of 10^{-3}. We are not too far from this result but it is quite difficult to achieve it with a small number of vectors and features.

Our architecture, in fact, is improvable with the addition of new data that will probably be available in future. Moreover it should be useful, for the validation of this procedure, to make some tests also over different type of patients, as example those affected of heart disease, hypertension, Parkinson and many others. Another improvement that we could reach in future is the consideration of medicines prescription and withdraw saved in Electronic Health Record. With the addition of these data we could perform a better classification of patient that are following correctly their therapy.

However, the biggest problem encountered in the development of this project was the absence of tags describing the "ground truth" in the dataset. In fact, as already explained, the tagging process is the most difficult part as each vector should be examined by a Doctor or an Expert in healthcare and tagged by hand. Indeed in this case we tried to perform this passage using an algorithm that, of course, presented different limitations in the classification of extreme cases.

Consequently the main problem for the progression of these tests is the lack of data available and and the issues arising from the right to privacy on the patients' data. We hope that in future there will be more possibilities for these applications that could be of great help in people's everyday lives.

Bibliography

[1] Enkin Murray Jadad Alejandro Oh Hans, Rizo Carlos. What is ehealth (3): A systematic review of published definitions. *Journal of Medical Internet Research*, 02 2005.

[2] E.K. Park W. Liu and U. Krieger. ehealth interconnection infrastructure challenges and solutions overview. *IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom)*, 2012.

[3] Anuja Konda. An analysis of ehealth modes, usage levels, enablers, and obstacles. *IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*, 2018.

[4] National Information Board. Personalised health and care 2020, using data and technology to transform outcomes for patients and citizens. 11 2014.

[5] Barbara Millucci. Dalle smart cities alla sanità digitale. *L'Economia*, 06 2019.

[6] Simona Regina. L'oracolo della salute. *Corriere Innovazione*, 03 2019.

[7] L. Tre. Intelligenza artificiale, ecco le nuove linee guida dell'europa. *Il sole 24 ore*, 04 2019.

[8] Ruggiero Corcella. Le linee guida per usare bene la tecnologia in sanità. *Corriere della Sera Salute*, 05 2019.

[9] Agenzia per l'Italia Digitale. Documento di progetto dell'infrastruttura nazionale per l'interoperabilità dei fascicoli sanitari elettronici (art. 12 - comma 15-ter – d.l. 179/2012). 2017.

[10] Milieu Ltd. and Time.lex. Overview of the national laws on electronic health records in the eu member states. *National Report for Italy*, 2014.

[11] Progress of ehr in italy. `https://avanzamentodigitale.italia.it/it/progetto/fse`.

[12] Share of citizens using the electronic health records in italy. `https://www.statista.com/statistics/820788/ehr-usage-among-citizens-by-region-in-italy/`.

[13] Muhammad Jawad Hashim. User interactivity in ehealth applications, a novel taxonomy. *12th International Conference on Innovations in Information Technology (IIT)*, 2016.

[14] M. Farrow E. O'Connor and C. Hatherly. Randomized comparison of mobile and web-tools to provide dementia risk reduction education: Use, engagement and participant satisfaction. *JMIR Ment Health*, 12 2014.

[15] S. M. Kelders B. J. Brandenburg N. Nijland, J. E. W. C. van Gemert-Pijnen and E. R. Seydel. Factors influencing the use of a web-based application for supporting the self-care of patients with type 2 diabetes: a longitudinal study. *J. Med. Internet Res.*, 04 2011.

[16] M. Quade M. Arnhold and W. Kirch. Mobile applications for diabetics: a systematic review and expert-based usability evaluation considering the special requirements of diabetes patients age 50 years or older. *J. Med. Internet Res.*, 2014.

[17] P. W. Stone S. J. Iribarren, R. Schnall and A. Carballo-Diéguez. Smartphone applications to support tuberculosis prevention and treatment: Review and evaluation. *JMIR Mhealth Uhealth*, 2016.

[18] M. Volk M. Isaković, U. Sedlar and J. Bešter. Usability pitfalls of diabetes mhealth apps for the elderly. *J Diabetes Res*, 2016.

[19] Flavio Fabbri. App, cartelle cliniche online e 3d: come cambia la sanità italiana, in un paese che invecchia rapidamente. *key4biz*, 06 2019.

[20] Agenzia sanitaria e sociale regionale dell'Emilia Romagna. Cause di non aderenza al percorso diagnostico-terapeutico assistenziale del diabete mellito: uno studio quali-quantitativo per il miglioramento. 2018.

[21] Sezione Attuazione Programmazione Sanitaria Settore Assistenza Distrettuale e Cure Primarie Regione Veneto, Area Sanità e Sociale. Percorso diagnostico terapeutico assistenziale (pdta) regionale per la gestione integrata della persona con diabete tipo 2. 2015.

[22] Gruppo di studio AMD diabete di tipo 1 e transizione. Percorso diagnostico terapeutico assistenziale per la gestione della persona adulta con diabete di tipo 1.

[23] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 32 Avenue of the Americas, New York, NY 10013-2473, USA, 2014.

[24] Marvin Minsky and Seymour Papert. *Perceptrons: an introduction to computational geometry*. MIT Press Cambridge, MA, USA, 1969.

[25] *Support Vector Machines: Theory and Applications*, volume 2049, 01 2001.

[26] Sharma Himani and Kumar Sunil. A survey on decision tree algorithms of classification in data mining. *International Journal of Science and Research (IJSR)*, 5, 04 2016.

[27] Scikit learn developers. Scikit-learn user guide. 07 2019.

[28] Gridsearchcv. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[29] K-means clustering. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

[30] C-support vector classification. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[31] Multi-layer perceptron classifier. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[32] Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations, San Diego*, 12 2014.

[33] Decision tree classifier. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[34] J. de Paiva Azevedo, H. Delaney, M. Epperson, C. Jbeili, S. Jensen, C. McGrail, H. Weaver, A. Baglione, and L. E. Barnes. Gamification of ehealth interventions to increase user engagement and reduce attrition. In *2019 Systems and Information Engineering Design Symposium (SIEDS)*, pages 1–5, April 2019.

[35] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2019.