AALBORG UNIVERSITY
DENMARK

Deep Neural Network for Alzheimer's disease detection

Aalborg University

M.Sc. Innovative Communication Technologies and Entrepreneurship

Raquel Cacho Zurrunero

June 2019

Aalborg University Copenhagen
A.C. Meyers Vænge 15
2450 København SV

Semester Coordinator: Henning Olesen

Secretary: Maiken Keller

Title:
A Deep Neural Network for Alzheimer's disease detection

Project Period:
Spring Semester 2019

Semester Theme:
Master Thesis

Supervisor:
Per Lynggard

Project group no.:
Non applicable

Members:
Raquel Cacho Zurrunero

Copies:
1

Page numbers:
62

Date of completition:
June 6, 2019

Abstract:

The purpose of this thesis is to develop a system that shows how deep learning can be used to improve the diagnosis of Alzheimer's disease. The project will be based on the literature review of the disease and the research of the most recent machine learning techniques in order to perform the analysis of a possible solution.
As a result of this analysis, a conceptual design of the system is proposed. The proposed solution will be based on a Multilayer Perceptron architecture that allows to predict the probability of having the disease based on the patient's clinical data.
Finally, the solution will be implemented and tested, achieving an accuracy of 82.61%.

Contents

List of acronyms and abbreviations

AD Alzheimer's Disease
ADCs Alzheimer's Disease Centers
ADGC Alzheimer's Disease Genetics Consortium
ADNI The Alzheimer's Disease Neuroimaging Initiative
ADRDA Alzheimer's Disease and Related Disorders Association
APP Amyloid Precursor Protein
AUC Area Under the Curve
CNN Convolutional Neural Network
CPAD The Critical Path for Alzheimer's Disease
CSF Cerebrospinal Fluid
CSV Comma-Separated Values
CT Computed Tomography
DL Deep Learning
DNN Deep Neural Network
fMRI Functional MRI
FNR False Negative Rate
FPR False Positive Rate
FTLD Frontotemporal Lobar Degeneration
LBD Lewy Body Disease
MDS Minimum Data Set
ML Machine Learning
MLP Multilayer Perceptrons
MMSE Mini-Mental State Exam
MRI Magnetic Resonance Imaging
NACC The National Alzheimer's Coordinating Center
NCRAD National Centralized Repository for Alzheimer's Disease
NIAGADS National Institute on Aging Genetics of Alzheimer's Disease Data Storage Site
NINCDS National Institute of Neurological and Communicative Disorders and Stroke
NP Neuropathology Dataset
OASIS The Open Access Series of Imaging Studies
PET Positron Emission Tomography
RDD Researcher's Data Dictionary
RNN Recurrent Neural Network
ROC Receiver operating characteristic
TNR True Negative Rate
TPR True Positive Rate
UDS The Uniform Data Set

Chapter 1

Introduction

Alzheimer's Disease (AD) is a neurological disorder that causes the death of nerve cells in the human brain. AD usually begins gradually and its first symptoms may be attributed to the increment of the age or common forgetfulness. As the disease progresses, the patient's cognitive abilities deteriorate, including the ability to make decisions and carry out daily tasks. Currently there is no cure for the disease, only a series of guidelines can be followed to perhaps delay the progress of it. For this reason, an effective diagnosis will be a key factor in order to improve the quality of life of their patients.

The motivation for the creation of innovation to support the battle against Alzheimer's disease is evident, not only from an ethical perspective but also due to the continuous proliferation of Alzheimer's cases in our society. Today, 50 million people worldwide live with dementia, where two-thirds of them have Alzheimer's disease [1]. Alzheimer's cases have overtaken cancer ones to become the most feared disease in the United States, with a new case appearing every three seconds in the world [2]. At the moment the diagnosis of this disease is made by combining an analysis of the patient's medical history, different cognitive tests and various clinical tests, such as photographic scans of the brain. But is all this enough given the importance of an early diagnosis in the treatment of the disease?

Nowadays, through Machine Learning, it is possible to analyze data on a large scale and with different algorithms, detecting patterns and models in a very short period of time. In this way, there is a significant improvement in diagnostic methods using techniques which are even imperceptible to human experience and reasoning. On the top of that, these days the world of Machine Learning is more advanced than ever before, thanks to the newest deep neural networks. Simply explained, deep neural networks enable the creation of systems which

are powerful enough to represent any finite deterministic mapping between any given set of inputs and a set of corresponding outputs. These networks allow powerful data processing, allowing processes as complex as image identification or natural language processing.

In view of all the aforementioned, the aim of this project will be to analyze the possible connection between an improvement in the diagnosis of AD and the latest deep learning techniques. The number of variables that can influence the appearance or not of Alzheimer's disease are numerous and above all uncertain, being the human capacity a bounded resource to detect early cases of the disease with confidence. So, would it be possible to analyze all these variables through different deep learning techniques in order to offer a result that indicates the probability of developing such disease? Perhaps technology can be united once again with the medicine to discover new methods that allows to reveal the most determining parameters in the presence of the disease.

1.1 Problem formulation

As has been mentioned, AD is a growing problem in our society. More and more cases of AD are being found, and there is still no cure. Currently the area of machine learning is on the rise, developing projects in all sorts of areas of society. This learning allows predicting an output from different variables, being very useful for different clinical diagnostic processes [3].

The aim of this project will be to find the link between these two areas: the diagnosis of Alzheimer's disease and machine learning techniques. To do so, each of these areas will be studied separately, obtaining a critical view of the current situation. This vision will allow to initiate an elaborated analysis, where once understood the present problem, the presence or not of a possible solution through machine learning techniques can be analyzed.

The final objective of the report is presented below in the form of a research question. In order to address this goal, different sub-questions have also been elaborated. These sub-questions will enable to structure the way towards a final solid solution.

How could machine learning techniques be used to improve the diagnosis of Alzheimer's disease?

- Which data will be necessary in order to train the system successfully?

- Which is the most suitable architecture and parameters to achieve an accurate result?
- What level of accuracy can be achieved?
- What framework could be used to implement and test the selected model?

1.2 Limitations

The realization of this project will be affected by different limitations. These limitations will present what is not expected to be addressed with the implementation of the project, or various factors that have influenced the implementation of it.

In the first place, given the academic objective of the report, the time available for the realization of the project will be limited. This will directly affect the complexity of the system. Given the time constraints for research and training in various machine learning technologies, it will not be possible to address all the current techniques. This will make us discard the most complex techniques, such as image processing or sound through machine learning systems.

Aspects such as the security or privacy of the system will not be analyzed. The present project will present a solution in which the possible factors that provide security and privacy in the processing of the information have not been examined. In the case that the project will be used with potential real patients, these aspects would have to be analyzed and implemented.

In the same line, the economic viability or possible business value of the system will not be evaluated. A business model and possible cases of use will not be discussed. For this reason, in the situation of a real implementation of the system in society, the value proposal of the system and its possible costs should be analyzed too.

Finally, the data set used for the implementation of the system represents a bias of the world population. It does not represent all possible countries, therefore, it will be difficult to generalize the results to all the geographic regions.

1.3 Methodology

The methodology used in the elaboration of the project will be presented below. This methodology will allow us to address the research question and organize the way of working. Two ways of approaching the project have been used. In the first place, a preliminary plan has been drafted based on the time constraint for the realization of the report. This plan is illustrated in the form of a Gantt Chart, and will represent a time-plan to follow that has as its goal the resolution of our research question in the time available. Secondly, a schema that represents the current way of working will be presented. This scheme will be an Agile representation of the elaboration of the project, where different loops are represented in which the feedback received by the academic supervisor is continuously applied.

Both schemas are illustrated on Appendix A and B.

1.3.1 Gantt Chart of the report

As it has been previously mentioned, due to the time constraint on the realization of the project, the first thing that has been made is a time planning of the elaboration of the report.

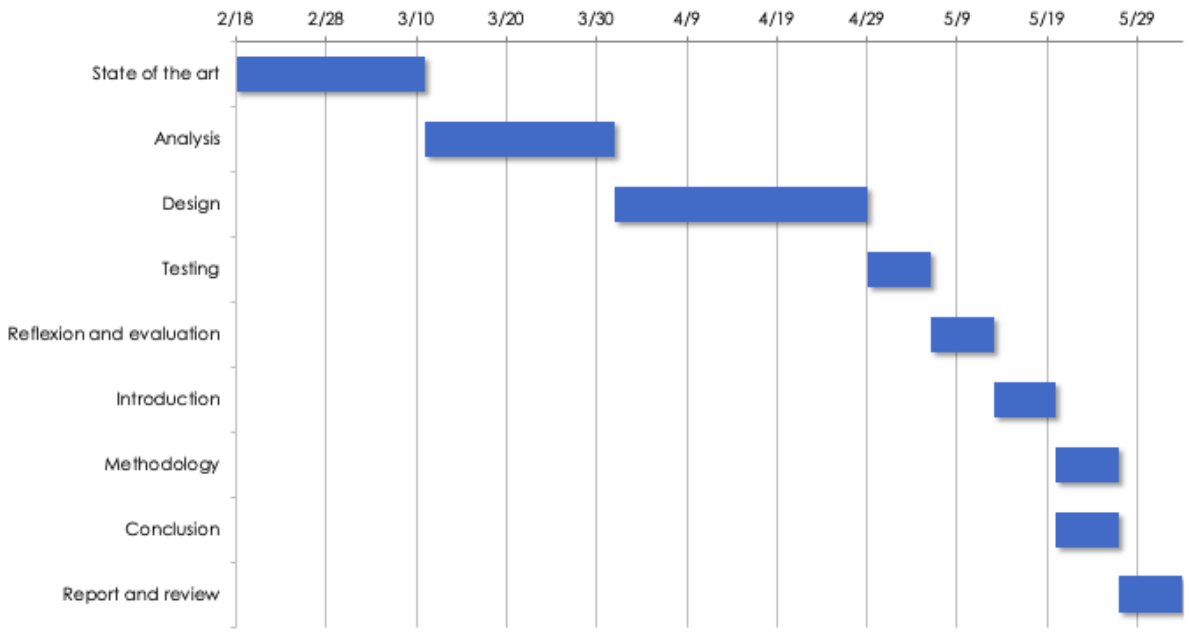


Figure 1.1: Gantt Chart representing the realization of the project

To do so, different ideally milestones has been established with the goal of delivering on time the present project. As it can be seen on Figure 1.1, each of the milestones corresponds with the realization of different parts of the report.

The realization of this graph has been very useful to establish each one of the parts or chapters necessary in the elaboration of the project. At the same time, it is also of great utility when organizing the optimal time to use in each one of them, offering a global vision of each one of the maximum times available for every stage of the project. The full Gantt Chart can be found on the Appendix A.

1.3.2 Agile realization of the project

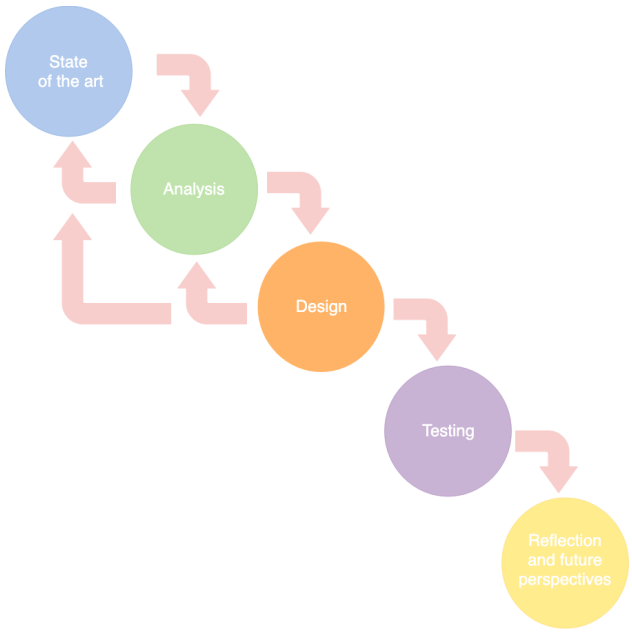


Figure 1.2: Agile process of the project

Although the creation of a Gantt Chart is very useful for organizing the project on a temporary basis, it does not address the daily way of working in the execution of the project. For this reason, an Agile schema has been produced representing the implementation of a system that allows to approach the research question presented in Chapter 1.

As it can be seen on Figure 1.2, this Agile process begins with a *State of the Art* phase. During this phase, the required knowledge regarding the AD topic and machine learning techniques should be acquired.

Once knowledge has been gathered in both areas, the *Analysis* phase will begin. This phase aims to build the requirements of the system through a critical view of the knowledge previously acquired. During this phase a first loop can be observed. In this loop it is possible to go back to the first phase to gain additional information and then include it in the analysis.

When the analysis has been completed, the *Design* and implementation of the system will begin. During this phase there may also be a need for new insights or more detailed analysis, which is why different loops are presented.

The results obtained will then be evaluated in the *Testing* phase. Finally, the entire system development process will be discussed through the *Reflection and Future perspectives* phase.

A full description of this process can be found on Appendix B. It is important to understand the reasons behind the choice of this working method. Although a static goal-oriented plan, such as that presented in Section 1.3.1, is necessary in order to organize resources and limited time, it is not sufficient to reflect the day-to-day work in implementing the project. This is why an agile methodology has been chosen that allows to improve the results returning to previous phases if necessary. This methodology also represents very well the feedback received by the academic supervisor, improving initial versions of the project until a final result is reached.

1.4 Structure of the report

The structure of this report will be presented below. This structure represents the flow made for the elaboration of the system, so that the reader can analyze in a sequential way each one of the necessary steps for the elaboration of such system.

An introduction to the project has been presented in the first chapter. Its motivation, limitations, methodology and the problem to be solved have been elaborated. In this chapter, the research question has been been presented, which will be analyzed throughout the elaboration of the project. Also, the methodology used along the implementation of the system has been exposed. This methodology will represent the plan to address the research question and each of the steps that have been decided to implement.

The second chapter, State of the Art, presents a theoretical introduction in each of the topics to be addressed in this report. First, the current situation of Alzheimer's disease

will be presented, focusing on its diagnostic methods. Secondly, different concepts necessary to understand the future realization of a machine learning system will be explained. This theoretical base will allow the future analysis and implementation of the system.

The discussion of each of the elements necessary to approach the research question presented in chapter one, Introduction, will be carried out in the third chapter, Analysis. This chapter will be based on the knowledge acquired on the second chapter and will conclude with the presentation of the requirements of the system.

The fourth chapter of the project, Conceptual Design, will present the conceptual design of the system. This solution will be based on the fulfillment of the requirements established in chapter three, Analysis.

The implementation of the system will be addressed in chapter five, Implementation. It will be based on the conceptual design presented in chapter four, Conceptual design, and will be evaluated through chapter six, Testing.

In chapter six, Testing, the system will be proved. This will serve to analyze the overall result of the report.

Finally, the conclusions of the project will be presented in chapter seven. This chapter will recapitulate all the work done and will answer the research question presented at the beginning of the report.

Chapter 2

State of the Art

In this chapter the basic concepts for understanding Alzheimer's disease will be introduced, addressing its main causes, symptoms and diagnostic methods. In the second part of the chapter, the basis to perform a definition of machine learning will be presented. The types of problems that can be solved with such technology and its architecture will be explained.

All this theoretical background will be the first step in order to confront the research question presented on Chapter 1 and represents the foundations needed in order to start an analysis on Chapter 3.

2.1 Alzheimer disease

Dementia is defined as the deterioration acquired in cognitive abilities that interferes with the satisfactory performance of activities of daily living . Alzheimer's Disease (AD) is a type of progressive dementia that has memory deficit as one of its earliest and most pronounced symptoms [4].

As a general rule the patient progressively deteriorates, exhibiting perceptual, language, and emotional problems as the disease progresses. This deterioration is due to the fact that the nerve cells, or neurons, that allow cognitive function in the brain have been damaged and no longer function normally. In addition, Alzheimer's disease usually occurs in combination with other types of dementia, which is called mixed dementia.

AD has become a major social problem for millions of families and national health systems

worldwide. It is one of the most important causes of death in developed countries, behind cardiovascular disease and cancer [5]. This dementia have such a strong impact on the health system and society. Not only for its irreversible nature and the lack of curative treatment, but also due to the huge burden that the disease impose on the family of the patients. Although the most prevalent symptom of AD is the gradual loss of the ability to remember new information, the following ones are also common of the disease [6]: difficulties planning activities or solving problems; challenges completing familiar tasks at home, work or at leisure; confusion about time or place; problem of knowing the current day of the week or where they are; speech or writing difficulties; decreased ability to organize personal items and remember where they are located; apathy or depression, including drastic personality or mood changes.

The appearance of these symptoms as well as the progression of the disease, varies greatly from one individual to another, making his diagnosis a difficult and laborious labor on each patient.

2.1.1 Causes

Although the exact causes of Alzheimer's disease and why it occurs are still unclear, abnormal presences of two proteins have been identified in the brains of the patients of AD [7]. It is know that the brain is made of neurons, which are interconnected to form a vast network. Such connections, named as synapses, enables the transmission of information from one neuron to another. In the case of the patients of AD, two main brain lesions are formed which affects directly to those connections in their brains.

The first one is produced by the beta-amyloid protein, or commonly called amyloid. This protein tends to accumulate in form of plaques in the brains of AD patients. Such aggregates, which disrupt the interconnection between neurons, are called senile plaques or amyloid plaques. Senile plaques are irreversible, meaning that once they are formed, their disappearance is no longer possible. Elderly individuals suffering from Down's syndrome, are particularly prone to develop insoluble amyloid deposits which results on the probable development of AD [8]. The reason behind this risk factor is the additional copy of the Amyloid Precursor Protein (APP) gene on individuals with Down syndrome. This gene usually increases the production of beta amyloid protein, triggering the chain of biological events leading to Alzheimer's disease.

The second protein that is linked with the presence of the diasease is the tau protein.

When a neuron communicates with another, a signal goes from the body connection, knows as soma, to the neuron's synapse to transfer the information. The signal passes through the skeleton of the neuron which is composed by microtubules. These microtubules are stabilized by the tau protein. In healthy neurons, tau normally binds to and stabilizes microtubules. In AD however, tau protein becomes defected and detaches from the microtubules sticking to other tau molecules [9]. Thus, the skeleton of the neuron disassociates since his microtubules are not longer maintained by the tau protein. Without the skeleton the neuron degenerates, loosing all his connections with the rest of the neurons and generating neurofibrillary tangles which causes sooner or later his dead.

Neurofibrillary tangles and senile plaques do not follow the same pathway in the brain over time. Neurofibrillary tangles first develop in a region called the hippocampus, which is the responsible of learning and memory functions. The progression involves brain atrophy, being reflected in memory problems, speech difficulties, recognition or incapacity to perform organized tasks. Senile plaques develop differently, they are initially observed in the cortex, secondly in the hippocampus to eventually reach the whole brain following a central pattern movement. Their progression does not usually corresponds to the symptoms of the disease. And, even if these brain abnormalities are common causes of the AD, what is still a challenge to be solved is the reason behind these unusual levels of both proteins. Currently, the few theories that attempt to explain the unusual behavior of these proteins are disparate, from possible genetic, hereditary causes or the patient's lifestyle. In addition, amyloid plaques and neurofibrillary tangle formation may occur on different time scales. Amyloid concentration is thought to develop first during the long preclinical phase, while the development of neurofibrillary pathology accelerates slightly before the appearance of the symptomatic phase of AD [10].

As a result, even if the cause of AD remains controversial and is incompletely understood, the presence of senile plaques and neurofibrillary tangles constitute the major neuropathological characteristics of AD resulting in an important area to continue the research [11].

2.1.2 Diagnosis

The criteria for the clinical diagnosis of Alzheimer's Disease (AD) were established by a National Institute of Neurological and Communicative Disorders and Stroke (NINCDS) and Alzheimer's Disease and Related Disorders Association (ADRDA) workgroup in 1984 [12]. This initial criteria were designed with the expectation that in most cases, subjects who man-

ifest the common symptoms of the disease would have the AD pathology. However, in the following years of research it has become clear that this clinical-pathological correspondence is not always consistent. For example being possible for a patient to present amyloid plaques in the absence of any obvious symptoms [13]. Nowadays, the diagnosis of the AD can be divided in two main areas depending on whether it is oriented into the different qualitative and quantitative clinical expressions of disease or in the pathophysiological process that underlies the syndrome. As a result, the actual criteria combines clinical and neuropathological patterns assigning three different level of diagnosis, "possible AD", "probable AD" or "definite AD" [12].

Pathophysiological diagnosis Biomarkers are parameters - physiological, biochemical or anatomic - that can be measured in vivo that reflect specific features of the disease related to pathophysiological processes [14]. In the case of Alzheimer's disease, biomarkers are measured by image scanning, blood analysis or lumbar puncture tests. But in order to use a biomarker as core of a diagnostic, it should be validated beforehand. This requires multiple studies in large groups of people establishing if the biomarker accurately and reliably indicates the presence of disease. Unfortunately, this is not the current situation of the Alzheimer's biomarkers, which can not offer an accurate and standardized threshold that indicates the presence of the disease. As a result, is important to highlight that most of the diagnostics of AD dementia are based in clinical processes, being the pathophysiological diagnosis a complementary method for the medical practitioner.

The first biomarker that can help in the diagnosis of the AD is the Cerebrospinal Fluid (CSF) examination. Due to the free transport of proteins between the brain and the the body through the CSF, beta-amyloid and tau levels are reflected in the CSF analysis which can be significant even at an early stage of disease [15]. However, the association between CSF biomarkers and the concentrations of deposited amyloid or neurofibrillary tangles in the brain remains unclear. It has been proposed that the generation of amyloid plaques in the brain may results in a reduction of amyloid level in the CSF analysis. Whereas the presence of neurofibrillary tangles will be represented by high tau protein levels on the CSF results [16]. But not all scientific studies affirm these hypotheses, making such a biomarker an inconsistent test with uncertain value [17][18].

Neuroimaging is among the most promising areas of research focused on early detection of Alzheimer's disease. Structural imaging provides information about the shape, position or volume of the brain. Structural techniques include Magnetic Resonance Imaging (MRI) and

Computed Tomography (CT). Through those image analysis are focused on the identification of a possible atrophy in the hippocampus or in the entorhinal cortex, which is associated with a decline in memory function and an increased risk of AD [19]. However, scientists have not yet agreed upon standardized values of the brain volume that would establish an accurate prevalence of the disease. In the case of functional imaging, it reveals how well cells in various brain regions are working by measure their sugar and oxygen levels. Functional techniques include Positron Emission Tomography (PET) and Functional MRI (fMRI). Functional imaging research suggests that those with Alzheimer's typically have reduced brain cell activity in certain regions. For example, studies with fluorodeoxyglucose in PET analysis indicate that Alzheimer's is often associated with reduced use of glucose in brain areas important in memory, learning and problem-solving [20]. However, as always, there is not yet enough information to translate these general patterns of reduced activity into standardized diagnostic information of the disease.

In resume, biomarkers are mainly used to detect anomalies in the brain related with the amyloid or tau accumulations or as a tool to discard other possible diseases. But there is important to understand, again, that although sophisticated image and CSF analysis methods do exist, should not be used as the only procedure for the diagnosis of AD [12]. The reasons behind this limitations are firstly, that the core clinical criteria provide an accurate diagnostic in most of the patients. Secondly, the need of more research to be done in order to establish standardized biomarker's results and methods. And finally the limitation that not every medical institution across the world has access to such sophisticated tests.

Clinical diagnosis Regarding the clinical diagnosis of the AD. A comprehensive physical examination of the patient should be performed. It includes a brief neurological and mental status evaluation, a review of the lifetime medical history and an analysis of the patient's lifestyle.

In order to test the mental status of the patient, the most common symptoms of the disease will be evaluated through a cognitive examination. The cognitive capacity of the patient is evaluated based on a combination of two methods. Firstly, a personal interview with the patient and close family members will be performed. And secondly, a brief cognitive exam which reflects the mental state of the patient. The Mini-Mental State Exam (MMSE) is one of the most commonly used screening tests to evaluate cognitive functioning [21]. The MMSE is a brief, structured test that takes about 10 minutes to complete. The test also includes the evaluation of variables such as the gender of the patient, his age, his educational

level or various risk factors such as diabetes or hypertension.

Finally the physician should discard other similar diseases as a diagnosis of mild cognitive impairment (MCI), fronto-temporal dementia, Lewy body disease or vascular dementia [22].

As a result of this cognitive examination, the physician will be able to apply the standardized DSM-IV Criteria for the Diagnosis of Alzheimer's disease, resulting on a clinical diagnosis of the dementia [23]. The complete version of the DSM-IV Criteria for the Diagnosis of Alzheimer's can be found in Appendix C. This diagnosis is defined as impairment in two or more cognitive domains which corresponding with the memory domain, and one or more of the following: aphasia (language problems), apraxia (impaired motor ability), agnosia (failure to recognize known objects), or deterioration in executive function.

2.1.3 Summary

As a summary of all the information gathered on the previous sections, the following list offers the main points that can be highlighted from a medical prespective of the disease:

- The AD is characterized by affecting the cognitive function of the patients, specially to the memory domain.
- The abnormal presence of beta-amyloid and tau proteins produces senile plaques and neurofibrillary tangles, respectively. Although they have been identified as the main neuropathological characteristics of the disease, the cause behind the unusual behavior of these proteins is still uncertain.
- The diagnosis of the disease can be divided in two main areas:
  - Pathophysiological diagnosis: based on the measure of biomarkers (CSF, MRI, PET .. etc).
  - Clinical diagnosis: based on the medical history of the patient, an analysis of his lifestyle or habits and cognitive/metal evaluation.

2.2 Machine Learning

Arthur Samuel in 1959 defined Machine Learning (ML) as the "field of study that gives computers the ability to learn without being explicitly programmed". Nowadays machine

learning can be understood as a combination of several disciplines such as statistics, information theory or functional analysis.

Depending on the type of learning task to perform, ML can be subdivided into two different fields, supervised or unsupervised learning. Supervised learning requires a priori knowledge of what the result should be [24]. Pairs of data inputs and data outputs have to be presented to the ML system during the learning phase. The learning process will be focus on trying to guess the output for a particular input which, later on, will be contrasted with the real output. Therefore the ML system will learn with the computation of this process for each corresponding input. On the other hand, unsupervised learning is based on the clustering approach. In this type of learning there is no tagged information. There is no distinction between data input or output, being just a sum of different information. In this case the learning will be focused on the searching of patterns, resulting in the creation of different clusters or classifications among the information [25].

Approaching the different ML techniques to the present project, the following sections can be addressed towards supervised learning. This is because the objective will be to determine a specific output, the presence or not of Alzheimer's disease for a given patient. This output will be determined by the combination of different medical input data, being the objective of the project to generate a model which could generate an accurate result when new information is presented to the system.

Supervised problems can be categorized into regression and classification problems, depending if the output is continuous or discrete. In this project, the focus is set in a classification problem, where the goal is to learn a mapping from inputs x to outputs y, where y $\varepsilon \{1, ..., C\}$, C being the number of classes to whom y may belong [26]. As a result, the output will be a class probability vector which is limited to only two values in the case of binary classifications, C = 2. In this case, like it is the case for this project, the result will be each of the probabilities for a certain input x to belongs (y = 1) or not (y = 0) to a certain group.

In order to understand the learning process of a classification problem, it is important to notice that the supervised keyword comes from the idea of having a previously labelled and classified data set, that is, having a sample set which is already known to which group, value or category the examples belong. With this group of data, called training data, a model is designed to predict future outputs. The algorithm learns to classify the input samples by comparing the result of the model, and the real label of the sample, making the respective compensations to the model according to each error in the estimation of the result.

To measure the efficiency of learning, it should be tested if the generated model can, from the trained examples, generalize the learned behavior so that it is good enough on data not seen a priori. The most common way to measure this accuracy is by saving some of the initial examples to be used later as validation of the learned machine. The correctness of a classification can be evaluated by computing the number of correctly recognized class examples (true positives), the number of correctly recognized examples that do not belong to the class (true negatives), and examples that either were incorrectly assigned to the class (false positives) or that were not recognized as class examples (false negatives) [27]. These four measures can be represented on the confusion matrix.

2.2.1 Classification in Deep Learning

Even ML technology powers many different supervised issues in the society, his ability to process natural data in their raw form will be limited [28]. For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data, such as the pixel values of an image, into a suitable internal representation or feature vector from which the learning system could work with [29]. The big drawback of ML methods reside in this handmade feature selection which will be needed in order to identify and remove unneeded, irrelevant and redundant attributes from data that do not contribute to the accuracy of a predictive model.

Deep Learning (DL) allows to receive raw data as input of the system, being able to automatically discover all the necessary relations to perform the classification. This is possible building a layered structure composed by different simple modules which are able to learn by themselves and compute non-linear mappings, known as Deep Neural Network (DNN) [28].

DNN is a mathematical representation of the human neural architecture [30]. A neural network is composed of a series of nodes, or neurons, which are organized in layers. It is formed by an input layer with as many neurons as features has the input, as many hidden layers as the models requires and an output layer. In the case of binary classification, the output layer has only one unit that outputs the probability to belong to the positive class. Each cell of the network has an output that is transmitted to other neurons in the network. At the same time, each neuronal connection has a coefficient called weight, by whom the outputs are multiplied. Thus, each neuron receives as input, the weighted outputs of the

previous neurons. If the value of this sum is above a threshold, it fires, sending its output to the next neurons. The weighted inputs are summed and passed through an activation function, which is a simple mapping of summed weighted input to the output of the neuron. If the summed input was above a threshold, for example 0.5, then the neuron would output a value of 1.0, otherwise it would output a 0.0. In addition, a single bias node is added for the input layer and every hidden layer, which typically will produce constant value 1 [31]. This bias node is going to be weighted too, being possible to transform it into negatives or positive values, allowing the output of an activation function to be shifted [32].

Figure 2.1 illustrate one of the simplest example of a DNN with two input neurons $a_1^{[1]}$ and $a_2^{[1]}$, a hidden layer with two neurons $a_1^{[2]}$ and $a_2^{[2]}$, and a single output neuron $a_1^{[3]}$. Both input neurons $a_1^{[1]}$ and $a_2^{[1]}$ correspond to the input feature vector $\mathbf{x}$, which in this case only has two features, being $\mathbf{x} = \{x_1, x_2\}$. An activated function is applied by the neurons in the hidden layers to the weighted inputs $a_1^{[2]}(\mathbf{x}) = \phi(za_1^{[2]}(\mathbf{x}))$, being $za_1^{[2]}(\mathbf{x}) = \theta_{11}^{[1]}a_1^{[1]} + \theta_{12}^{[1]}a_2^{[1]} + 1 \times \theta_{10}^{[1]}$ where $\theta_{11}^{[1]}$ and $\theta_{12}^{[1]}$ are weights and $1 \times \theta_{10}^{[1]}$ the bias term. Similarly, the output neuron applies an activation function to the weighted output of the hidden layer where $zo(\mathbf{x}) = \theta_{31}^{[2]}a_1^{[2]}(\mathbf{x}) + \theta_{32}^{[2]}a_2^{[2]}(\mathbf{x}) + 1 \times \theta_{10}^{[2]}$.

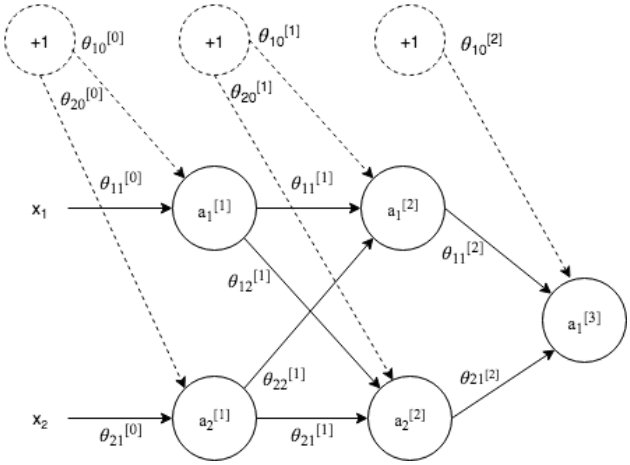


Figure 2.1: Example of neural network with one hidden layer

The learning process The DNN finds the correct mathematical manipulation to turn the input into the desired output, whether it be a linear or non-linear relationship. As it has been mentioned before, each of the connections between neurons is defined by a synaptic weight, which indicates the strength of that specific connection. The learning process is achieved by changing iteratively the values of the connection weights, trying to find the best hypothesis between the input and output of the network. This learning process, known as

the training of the network, is based on different training algorithms which minimize the difference between the network output and the desired result [33].

Trying to map it into mathematical expressions, the model uses a training dataset constituted by pairs input-target, $D = \{x_i, y_i\}$ where x_i is a vector of features an y_i are the expected outputs. Let θ denote the weights of the model and y represent the output value. The aim is to find a combination of θ that maximizes the likelihood of the output and the target. The training process starts making the samples go through the network and computing the error, which is called the forward pass phase . After the forward pass, the error is calculated and propagated backwards so as to optimize the θ. This method is called back-propagation, and is one of the most common training algorithms [34][35].

Thus, the training of a neural network by back propagation takes place in three stages: feed forward of the input information, calculation and back propagation of the associated error and adjustments of the weights. The process of forward pass and back-propagation is repeated until the gradient converges to an optimal solution. Thus, back propagation takes the error computed for the output of the network and propagates it backwards to all the neurons [36]. It calculates the error associated with each unit from the preceding layer and continue until the input layer is reached. These error measurements for each unit can be used to calculate the partial derivatives in every node (or neuron). These partial derivatives are used to minimize the cost function and update the weights. It is important to notice that bias nodes do not receive input from previous layer,thus, they should not be included in back propagation optimization algorithm [37].

As a result, DL can be understood as a type or evolution of different ML techniques, allowing to learn very complex functions through their layered structure. But this ability to perform such a difficult tasks has a price, the computational cost of training these complex networks [38]. In addition, DNN requires significant volumes of data to reach a decent level of accuracy, being not possible to apply DL architectures to small datasets [39]. Finally,the complexity of an architecture with so many layers makes it difficult to interpret all the intermediary operations. Thus, the algorithm which maps input and expected output is not as straightforward as ML techniques, being difficult to predict the consequences of small changes on the whole network.

Chapter 3

Analysis

In this chapter each of the key aspects in the implementation of a machine learning system that allows the diagnosis of AD will be analyzed. This analysis will be focused on each of the challenges exposed through the research question at the beginning of the report. The concepts learned in Chapter 2 will be taken into consideration, adding the critical vision that allows us to combine the field of AD with Machine Learning technology. Through this analysis it will be examined the ideal characteristics for a dataset candidate, the bases of the classifier architecture and the main programming languages and frameworks that can be applied in the project.

3.1 Identification of the required data

One of the key aspects to create value by means of automatic learning system is to collect the right input data to work with. In the case of this project, the information should be focused on all the data that are commonly used by the professional practitioners in the diagnosis of Alzheimer's disease.

As explained in the section 2.1, two major sources of information can be distinguished depending on the different methods used for the diagnosis of the disease. All the information that the doctor can obtain from the patient without external tests, or on the other hand, data obtained from specific tests such as brain scans or CSF examination. These last tests require a complex and professional exploration, such as the analysis of the images scanned. And although this analysis can be carried out through a deep learning system, as it have been

Risk factors analyzed	**Medical report**
Age	[44] [45] [46]
Female gender	[44] [46]
Low education level	[44] [45] [46]
Family history of AD	[44]
Presence of APOE4 gen	[45] [46]
High blood pressure	[44] [45] [46]
Heart disease	[44] [45]
Diabetes	[44] [45] [46]
Smoking	[44] [45] [46]
Frequent consume of alcohol	[44] [45] [46]
High cholesterol	[44] [46]
Depression	[44]
Head injuries	[44] [45]
Vitamin B12 deficit	[46]
Obesity	[44] [45] [46]

Table 3.1: Risk factors under scope for different medical researches

stated on Section 1.2, due to the time constraint and given its initial complexity, the project will be restricted to already processed quantitative data instead of images. In addition, the study of the disease based on the use of the analysis of MRI images by means of automatic learning algorithms or Machine Learning, is a field that in the last decade has given rise to a significant amount of scientific studies [40–43]. Being able to be, in the future, a complement to the present project in order to improve the performance of his outcome. Consequently, the search for a future database will be oriented only to that information that comprise the clinical diagnosis, such as medical history of the patient, age, sex or the presence of different risk factors. Also, any mental status evaluation such as the MMSE or any similar test, will be also valuable data since they are summarized in a standardized numerical score being easy to interpret.

In order to determine the necessary features that a dataset should contain, related works have been analyzed. Firstly, focusing on a medical point of view, a research about different medical investigations of probable risk factors has been conducted. Table 3.1 shows each of the parameters that have been analyzed through these medical studies to conclude their influence on the presence of the disease. On the other hand, an analysis of the selected features by similar machine learning projects has been performed showing the results on Table 3.2. As a result, merging the medical perspective of the main risk factors of the disease and the feature decision of different machine learning projects, the ideal dataset

Selected features	**Machine learning projects**
Age	[47] [48] [49]
Gender	[47] [48] [49]
Education level	[49]
Family history of AD	[48]
Vision/hearing deficiency	[49]
Hypertension	[47] [48] [49]
Heart disease	[47] [48] [49]
Diabetes	[47] [48] [49]
Smoking	[47] [48] [49]
Frequent consume of alcohol	[47] [48] [49]
Depression	[49]
Vitamin B12 deficit	[49]
Obesity	[47] [48] [49]
MMSE score or related cognitive tests	[47] [48] [49]

Table 3.2: Selected features by related machine learning projects

should contain, ideally, the maximum of the parameters mentioned in Tables 3.1 and 3.2.

However, not only the quality of a dataset will be enough to reach good results. Also, the available amount of data will be a key factor to consider in the selection of an appropriate dataset. Being useless to find a set of data that includes all the desired parameters if it is limited only to a small number of samples [39]. The system will need many samples from different patients in order to learn and deliver reliable results. Making a research about similar projects, a threshold value in the minimum size of the dataset can be 8.000 samples. This value is based on the projects exposed on 3.2, which are trying to solve a similar problem having 9.000, 5.432 and 22.594 samples respectively in each of them.

Likewise, it is important to emphasize the fact that these data must be labeled, linking the information collected from each patient to the outcome of his diagnosis. Since, at it was explained in 2.2 section, this labeled information allows the system to learn the desired output being able to guess in a future a likely label from an input unlabeled data. Furthermore, the variety of such information will also be another aspect to consider. Two types of results will be needed, those patients who have been diagnosed with AD and those who have not, and, the amount of information comprising each of these two groups should ideally be balanced [50]. A dataset will be considered as balance if it contains the same number of patients with AD as without the disease. But, as can be understood, if the search is focused on a database of patients who have gone through the possible diagnosis of the disease, most of the time the

cases of AD will predominate against those who have not, being impossible to find a fully balanced database.

3.2 Defining the classifier

Even there are many different machine learning paradigms to design a classifier: logistic regression, decision trees or support vector machines among others [51]. As it has been mentioned in Section 2.2, deep learning solutions provides a more powerful and flexible framework for supervised learning problems. By adding more layers and more units within a layer, a deep network can represent functions of increasing complexity which depends on a huge number of features. This flexibility on working with complex problems and non-linear situations, will be one of the main points to choose a DNN solution rather than traditional ML algorithms.

In addition, Deep Learning machines usually work better than traditional ML tools because they also learn the feature extraction part. In the case of the present project, the data input will be composed by a large amount of features that can influence in the presence or not of the disease. Due to this, the design of a DNN makes even more sense rather than the implementation of traditional ML algorithms where the engineering of that amount of features could be a really tedious work. This feature engineering not only implies a laborious task, it also requires a huge domain knowledge in order to select the most relevant features for the model [52]. Due to the lack of this medical knowledge, again the selection of a DNN rather traditional ML methods become even more clear than before.

But once that the DL path has been taken, the immediately next step to consider is the selection of the most appropriate type of DNN. There is no general rule about which type of DNN is the best for each specific problem, since most of the times, the same problem can be solved by different approaches. However, it is important to understand the data that will feed the model and the type of problem that is trying to be solved in order to, at least, discard some types of DNN which usually offer poor results under those situations.

Three main presentations of DNN has been analyzed to evaluate their possible applications to the project: Multilayer Perceptrons (MLP), Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN). Recurrent Neural Network (RNN) are neural networks which have a backward connection between hidden layers creating directed cycles in memory [53]. As a result of introducing feedback into the network structures, it is possible

to accumulate information and use it later. This type of DNN are the preferred ones for sequential data, as time series, audio or video [54]. But approaching the present project, RNN were discarded due to the reason that they are designed to work with time dependent prediction problems. In the case of the project, the input data will be composed by different patient's parameters that, usually, can be obtained on the first medical session. Therefore, it will be beyond the scope of the project to analyse the evolution of these parameters over time, definitively eliminating the time dependency of the data and consequently, the possibility of using a RNN solution.

Multilayer Perceptrons (MLP), also known as feed-forward neural networks, consists of a large number of simple neuron-like processing units, organized in layers. Is a feed-forward layered network of artificial neurons, where the data circulates in one way, from the input layer to the output layer [55]. Thus, every cell in the layer is connected to all the cells in the previous one but has no link with the neurons of the same layer. Due to the reason that MLP are suitable for classification prediction problems where inputs are assigned a class or label [56], it will be one of our preferred candidates. In addition, MLP are one of the simplest architectures to implement, being a very good choice in order to build a first version of the project.

However, the main problem of MLP architectures is that these networks do not scale well with image data as input [57]. Thus, Convolutional Neural Network (CNN) were created in order to face those difficulties working with multidimensional inputs, as images or sound processing [58]. The benefit of using CNN is their ability to develop an internal representation of a two-dimensional image using, instead of the normal activation functions, convolution and pooling functions through their hidden layers. As a result CNN are mostly used to work with multidimensional data, since at his proper name suggest, the convolution operation will be performed on the data which only makes sense for spatial information. Therefore, due to the reason that the input information of the present project will not be composed by images or any other multidimensional data, the use of a CNN architecture will not be one of the preferred choices.

3.3 Model hyperparameters

In order to implement a solution based on a neural network architecture, different parameters should be defined.

In a standard MLP there is a layer of input nodes, a layer of output nodes, and one or more intermediate layers. As a remider, a single-layer neural network can only be used to represent linearly separable functions. This means very simple problems where the two classes of the classification problem can be neatly separated by a line. This is not the case of the domain of the project, where most of the times, it can be found outliers defined by different patients with specific characteristics. Having an architecture composed by more than one layer allows to build more complicate functions that can fit with the solution of these complex problems, as the case of a MLP.

3.3.1 Number of layers and neurons

But, how many intermediate layer should be needed? And, by how many neurons should the layer be composed? These will be first issues to analyze for the implementation of the present system.

Even that there is not a fixed solution about the most appropriate number of intermediate layers, usually one or two hidden layer are sufficient to solve most of the non linear complex problems [59]. Regarding the number of neurons the most common rule-of-thumb methods are the following:

- The number of hidden layer neurons are 2/3, or between 70-90%, of the size of the input layer [60].
- The number of hidden layer neurons should be less than twice of the number of neurons in input layer plus the number of neurons in the output one [61].
- The size of the hidden layer neurons is between the input layer size and the output layer size [62].

It must be considered that all the information previously presented are suggestions in the design of the architecture, and none of them has been proven to be the correct one in all domains. It will be for this reason that, in a future design of the system, these references can be used as a base on the initial design but it will convenient to continue using the trial and error method to establish the best number of layers and nodes for a particular model. In addition, even though in machine learning there is the theorem of "No Free Lunch", which comes to mean that a solution cannot be generalized to any type of problem. As an extra reference it is possible to observe the architecture of similar projects, that is to say, with

the same source of data or similar and the same objective. In these projects, [49] [63] , an architecture with 2 and 1 hidden layers respectively has been implemented, obtaining an accuracy greater than 90%.

3.3.2 Activation functions

Another important aspect to analyze is which activation function will be used for each of the layers. The most common activation function used in MLP is Sigmoid activation function [64]. It takes a real valued input varied from $-\infty$ to $+\infty$ and saturates it to a bounded range between 0 and 1, which are the values used to represent the output class for a binary classification problem. In particular, large negative numbers become 0 and large positive numbers become 1. It is a useful activation function for the output layer of a binary classifier as its output can be interpreted as the probability of the input to belong to the positive class ($y = 1$). However networks using activation functions whose derivatives tend to be very close to zero, such as the Sigmoid function, are especially susceptible to the vanishing gradient problem [65]. The vanishing gradient problem can occur in artificial neural networks trained using gradient descent with backpropagation. When training such a network, the gradient of the loss function is used to adjust the weights of the network on each iteration. The vanishing gradient problem occurs when the gradient is sufficiently small so as to effectively prevent weights from updating during training, blocking the network from learning [66]. Also, the Sigmoid output is not zero-centered, which can lead to undesirable zig-zagging dynamics in the gradient updates for the weights [67] .

Other of the most common activation function used in backpropagation learning is Hyperbolic tangent, or Tanh, activation function. Hyperbolic tangent activation function is similar to Sigmoid activation function but it transform the input to an output between -1 and +1 instead 0 and +1 [68]. Like the Sigmoid function, its activations saturate, but its output is zero-centered.

While Sigmoid and Tanh have been commonly used activation functions, Relu provide faster and more effective learning of deep neural networks on complex and high-dimensional data [69]. Relu function works by thresholding values at 0. It outputs 0 when $x < 0$, and conversely, it outputs a linear function when $x \geq 0$ [70]. As it has a linear, non-saturating form, it was found that it greatly accelerates the convergence of the stochastic gradient descent, compare to the Sigmoid and Tanh. Also, compared to Tanh or Sigmoid neurons that involve expensive mathematical operations, Relu function can be implemented by simply

	Mathematical expression	Advantages	Disadvantages
Sigmoid	$h_\theta(x) = \frac{1}{1+e^{-x}}$	– Produce binary output, in range (0,1) [67] – Suitable for binary classification output layer [71]	– Saturates and kills gradients [65] – The result of the outputs is not zero centered [67]
Tanh	$h_\theta(x) = \frac{2}{1+e^{-x}} + 1$	– Output zero centered, in range (-1,1) [65]	– Saturates and kills gradients [65]
Relu	$h_\theta(x) = max(0, x)$	– Greatly accelerate the convergence, due to its linear, non-saturating form [67] – Non computational expensive [65]	– Dead neurons, once the gradient is zero the neuron will never activate on any datapoint again [72]

Table 3.3: Comparison between Activation functions

thresholding a matrix of activation at zero. However, it has an important drawback, Relu units are prone to die during training once that the gradient has reached the zero value.

As a resumen, in the Table 3.3, it can be found a comparison between the previous activation functions, including their mathematical expressions.

3.4 Evaluation metrics

The most commonly used metric to measure the performance of an automatic learning model is its accuracy [73]. This metric will not only be used to evaluate the overall performance of the system, but will also be the reference when selecting between different model hyperparameters. In this way, the combination of parameters that offer greater accuracy can be chosen.

In this subsection different metrics and evaluation techniques will be presented for two different objectives. The first one will concern all those metrics that allow to choose the

most suitable parameters for the model. These metrics mainly include cross-validation techniques for the choice of model hyperparameters [74] and the use of a validation set for the identification of under-fitting or over-fitting situations [75]. On the other hand, the main techniques for evaluating the final results of a classification model will be introduced. These metrics will be the accuracy of the model, its confusion matrix or its Receiver operating characteristic (ROC) curve [76].

The concept of cross validation will be necessary to be able to evaluate the functioning of our model with different parameters. Different models, with different parameters, will be trained through the training set and their performance will be validated with the validation set. This set will be necessary because if we use the test subset to validate the different models and choose the most appropriate one, then this test will have already been seen by our final model and will not be valid for the final evaluations.

Cross Validation or k-fold validation consists of dividing the training set into k subsets and, at the time of training, each k subset will be taken as the model's test set, while the rest of the data will be taken as the training set [77]. This process will be repeated k times, and in each iteration a different test set will be selected, while the remaining data will be used, as mentioned, as a training set. The performance will be the average of the performance in each iteration. This technique will be very useful when a small dataset is available. Another way to perform cross validation will be through the hold-out method. This method differs from the previous one in that the dataset is directly divided into training and validation subsets. This method will be much less computationally expensive, but will require a larger dataset size [74].

Comparing the two methods, the k-fold method has the advantage that all data are used to train and validate, so more representative results are obtained a priori. On the contrary, by means of the hold-out technique it is possible to have bad luck when making the a priori division between training and validation which may result in no representative samples of the dataset. For this reason, even if the dataset is not small, if there is sufficient computational capacity, it will always be better to perform a k-fold method instead of a hold-out [78].

K-fold validation process is used to select the model hyperparameters, repeating this process for each of the candidates. Thus, the precision and error are calculated for each of the models produced so that they can be easily compared and a finalist with the best results can be chosen. This technique will be very useful when a small dataset is available, but on the contrary, the great disadvantage of this technique will obviously be its great computational expense [79].

When training the model, it will be necessary to analyse the possible presence of two possible typical phenomena in machine learning: over-fitting and under-fitting situations. A model is going to be over-fitted when it performs well with training data, but its accuracy is noticeably lower with test data; this is because the model has memorized the data it has seen and could not generalize the rules to predict the data it has not seen [80]. On the other hand, under-fitting occurs when there is an excess of generalization of the model, which practically ignores all or most of the training samples.

It will be necessary to find a middle point in the learning of our model in which we are not under-fitting or over-fitting, and for this purpose the validation set will be used. By training our model with the training set and validating it at the same time through the validation set, the evolution can be seen along the training epochs. This evolution will be represented by its learning curve, which show the relationship between training set size and its error rate on the training and validation sets. They can be an extremely useful tool when diagnosing the performance of your model, as they can reveal whether the model is suffering from bias, under-fitting, or variance, over-fitting, problems.

On the other hand, once we have chosen the right parameters for our model and trained it, its results will be evaluated with new input data that will result in different metrics. As mentioned above, the most common metric to evaluate the model will be its accuracy. Accuracy simply measures how often the classifier makes the correct prediction. It is the ratio between the number of correct predictions and the total number of predictions. But the main limitation of the accuracy metric is that it assumes equal cost for both kinds of errors [81] [76]. In the area of application of the project it will be relevant to identify the errors of each type of class, since it will not have the same impact to diagnose a patient with AD erroneously, than not diagnose a patient with AD when he or she had AD. For this reason, it will be necessary to represent the confusion matrix of the model. A confusion matrix is a table that categorizes predictions according to whether they match the actual value in the data [82]. One of the table's dimensions indicates the possible categories of predicted values while the other dimension indicates the same for actual values. Therefore, through this confusion matrix, the correctness of a classification can be evaluated by computing the number of correctly recognized class examples (true positives), the number of correctly recognized examples that do not belong to the class (true negatives), and examples that either were incorrectly assigned to the class (false positives) or that were not recognized as class examples (false negatives) [27].

Finally, the classifier's ability to avoid a false classification can be measured with the ROC

curve, created by plotting True Positive Rate (TPR) versus its False Positive Rate (FPR) [83]. The true-positive rate is also known as sensitivity, recall and the false-positive rate is also known as the fall-out. Hence, the ROC curve shows the trade-off between sensitivity and specificity: the closer the curve is to the diagonal, the less accurate the test is. ROC provides tools to compare and select the most optimal models and it is considered an effective method of evaluating the quality or performance of diagnostic tests [83].

3.5 Programming languages and libraries

Currently there are a variety of programming languages that can be used for the implementation of DL systems. Therefore, one of the first issues that has been faced is which of these languages is the most appropriate for the present project. Before choosing one language or another, it is necessary to establish different selection criteria, seeing which language is the most suitable for each type of scenario. The analysis has been focused on two criteria, performance and ease of use. The performance criteria has been chosen due to the importance of reliable and fast results. Also, due to the limitation of time for the realization of the project, the ease of use will be an important aspect to analyze too.

Performance indicates that the model is executed as quickly as possible. This criterion is important, since many artificial intelligence applications must work and provide results in an acceptable time, otherwise they can be considered totally useless. In this case, the languages that offer the best performance are those with the lowest level, such as C or C++ [84]. The second criterion is the ease of use, or learning-ability, of the programming language. This criterion is not only affected by the programming language itself, but also by the number of libraries available. The programming languages with the fastest learning curve are those of the highest level, such as Python or R [85]. In the case of this project, given the limited time for its realization, the ease of learning and use of the programming language will be the most important criterion. Focusing the search on this approach two candidates have been analyzed, Python and R.

Python can be used both to structure data and to generate DL algorithms. One of the most remarkable features of Python is that it is an interpreted language, this means that it is not compiled unlike other languages such as Java or C/C++, but is interpreted at runtime [86]. It has a very extensive library catalog, and although many of these packages are being ported to R the machine learning libraries are more predominant for Python [87] [88]. R is one of the best languages for analyzing and manipulating data, because it was

designed for statistical and mathematical purposes [89]. R has special features that make it especially versatile for handling statistical elements, specifically for operations with matrices and vectors, which facilitates the manipulation of datasets. The great advantage of R will be its visualization possibilities, offering more complete packages than Python. As against R, its learning curve tends to be slower and more complicated if we compare it with Python.

In the case of the present project, prevalence more the amount of libraries and community of users with respect to the possibilities of visualization or previous manipulation of the data. For this reason, it has been decided to focus the project on the Python programming language.

Subsequently, it has been decided to analyze the most famous frameworks for the implementation of DNN in Python. As Andrej Karpathy, famous machine learning research scientist, stated on the Figure 3.1, the most mentioned frameworks on DL projects during the 2018 are: Tensorflow, Keras, Caffe and Pytorch.

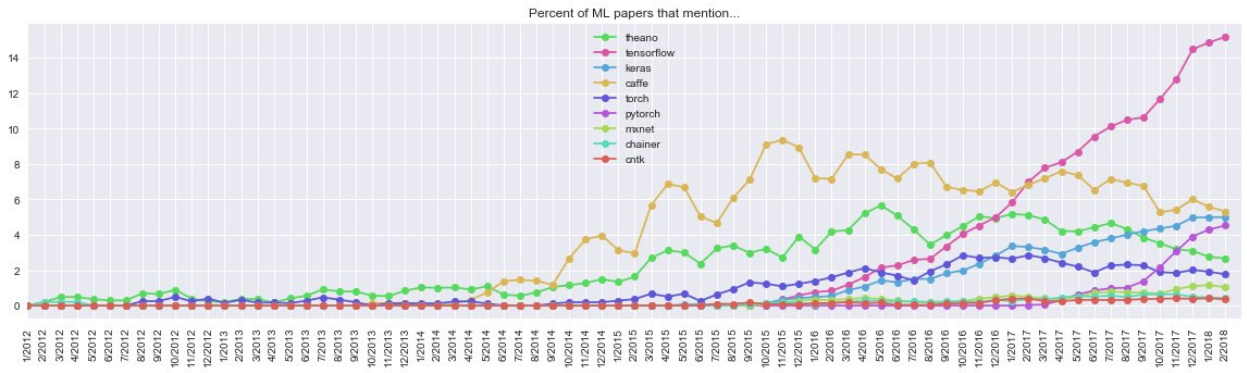


Figure 3.1: Most mentioned Machine lerning frameworks on academic papers. Source: Andrej Karpathy

Caffe is designed to build deep learning solutions in a very high level way of working, almost without writing any code [90]. This is a great advantage, as a solution concept can be reached in a very short time. But given the academic objective of this project, this framework has been discarded. Consequently, from now on, the analysis will focus on a comparison between Tensorflow, Keras and Pytorch. Keras is a high-level API for neural network development written in Python. It uses other libraries internally such as Tensorflow. Precisely because of this, it was developed with the purpose of facilitating and speeding up the development and experimentation with neural networks, offering the same results as TensorFlow but at a higher level of abstraction. But when a neural network becomes more complex, with many layers and parameters, the Keras framework may be limited. Here is

where TensorFlow is introduced, a numerical computation library that allows the neural network to be completely adapted to the user's needs, reaching a lower and deeper level of detail as compared to Keras. Pytorch, on the other hand, is a lower-level API focused on direct work with array expression. Pytorch offers the best performance, with a short training duration. Also PyTorch is much easier and simpler to debug than TensorFlow [91]. However, in terms of data visualization, Tensorflow is the most advanced one [92]. As a result, Keras is characterised by its ease of use, Tensorflow by its flexibility, level of detail and visualizations and Pytorch by its performance and facility to debug. Once again, given the purpose of the present project and the time limitation, its realization will focus on the simplest solution of use. This does not mean that in the future or in a professional context, a possible change will be considered to a lower level of abstraction, as Pytorch or Tensorflow, where better results or level of detail can be found.

3.6 Requirements

The following section can be understood as a result of all the analysis performed through this chapter. This requirement specification will be an important starting point on the design of the system, since it summarizes the main relevant aspects that the solution should complain.

They have been divided into functional – ones describing what the system should do - and non-functional requirements – ones describing what a system should be. Most of the functional requirements has been obtained from the 2.1 section, based on the actual procedures for the diagnosis of the disease. Non-functional requirements are based on the learning obtained from Section 2.2 and the analysis performed in Sections 3.1, 3.2 and 3.5. All of them are listed in the following tables, Table 3.4 and Table 3.5. Also, the requirements are prioritized using the MoSCoW prioritization technique. MoSCoW is an acronym for Must, Should Could and Would [93]. The categorization and meaning of categories in the MoSCoW technique are described in Table 3.6 .

3.6.1 Functional requirements

As described before, functional requirements are the ones that characterize what a system should do. Most of the time, they are functions which user can directly perceive and try. The list of functional requirements can be seen in Table 3.4.

ID	Description	Prioritisation	Reference
#1	The system should be able to process tabular information regarding the clinical history of the patient, habits, lifestyle or results from different cognitive evaluations as data input for future predictions.	M	Sections 2.1 and 3.1
#2	The system should clean and preprocess the training data.	S	Section 3.1
#3	The system should offer accurate results on the diagnosis of AD.	M	Section 2.1
#4	The system should present relevant metrics in order to evaluate the accuracy of the results.	S	Sections 2.2

Table 3.4: List of functional requirements gathered for the system.

3.6.2 Non-Functional requirements

Non-functional requirements, on the other hand, are the ones which users cannot directly perceive, but they describe what a system should be. Usually, the fulfillment of these requirements can be measured and evaluated. The list of non-functional requirements can be seen in Table 3.5.

ID	Description	Prioritisation	Reference
#5	The dataset used by the system should be big enough, at least 9.000 samples, to be processed by the deep learning architecture.	S	Section 3.1
#6	The dataset used by the system should be labeled, offering a field with the diagnosis provided by the doctor.	M	Section 3.1
#7	The dataset used by the system should be balanced, containing both types of diagnosis: AD and non-AD .	M	Section 3.1
#8	The dataset used by the system should contain relevant features which allows to perform accurate predictions. It should contains the maximum number of features mentioned in Tables 3.1 and 3.2	M	Section 3.1
#9	The architecture of the system should be based a MLP.	S	Section 3.2
#10	The architecture of the system can be upgraded into a CNN if there is time left.	C	Section3.2
#11	The system should be evaluated through his accuracy, confusion matrix and ROC curve.	S	Section 3.4
#12	Learning curves of the system should be analyzed, identifying the presence or not of a bias or variance problem	S	Section 3.4
#13	Model hyperparameters should be tuned through cross-validation techniques	S	Section 3.4
#14	The main programming language should be Python.	S	Section 3.5
#15	The preferred framework to build the system should be Keras.	S	Section 3.5

Table 3.5: List of non-functional requirements gathered for the system, with the reference where the given requirements was gotten from.

Prioritization group	**Description**
MUST	Minimum requirements for the system to function.
SHOULD	Desired requirements. The system will function without them, however, they have a high priority
COULD	Requirements that could be considered if there is time left
WOULD	Requirements that can be considered maybe in a future. Sometimes these requirements are listed here but are infeasible to achieve within the constraints of the project.

Table 3.6: Description of MoSCoW prioritisation

Chapter 4

Conceptual Design

The Conceptual Design chapter places all pieces, described and analyzed before, together in order to create a design of the system. To do so, all the decisions presented below will be based on the fulfilment of the requirement list exposed on Section 3.6. In first place, the chapter will introduce the general system overview, where the chosen system's parts are presented and the rationale for choosing them is given. Once that a general overview of the solution has been understood, the following sections will go through each of the modules that compose the system. Their required functionalities, interfaces or expected outputs will be defined. At the end of the chapter, a summary section is provided which allows to understand precisely which are the different functionalities to be implemented by each module in the following chapters of the report in order to answer the research question presented on Chapter 1.

4.1 System overview

This section presents the general architecture of the system. The architecture has been divided into smaller parts, to be able to tackle in a simpler manner each one of the functionalities required by the system. These parts will be independent modules, in charge of implementing specific functions. At the same time, these modules must be able to connect to each other, since the result of one will be the input of data from another. In figure 4.1, the overview of the solution can be appreciated. The system is divided into three main modules: Data collection and pre-processing, Neural Network and Testing. The data flow will run through each of the modules, presenting a final result through the last of them. In addition,

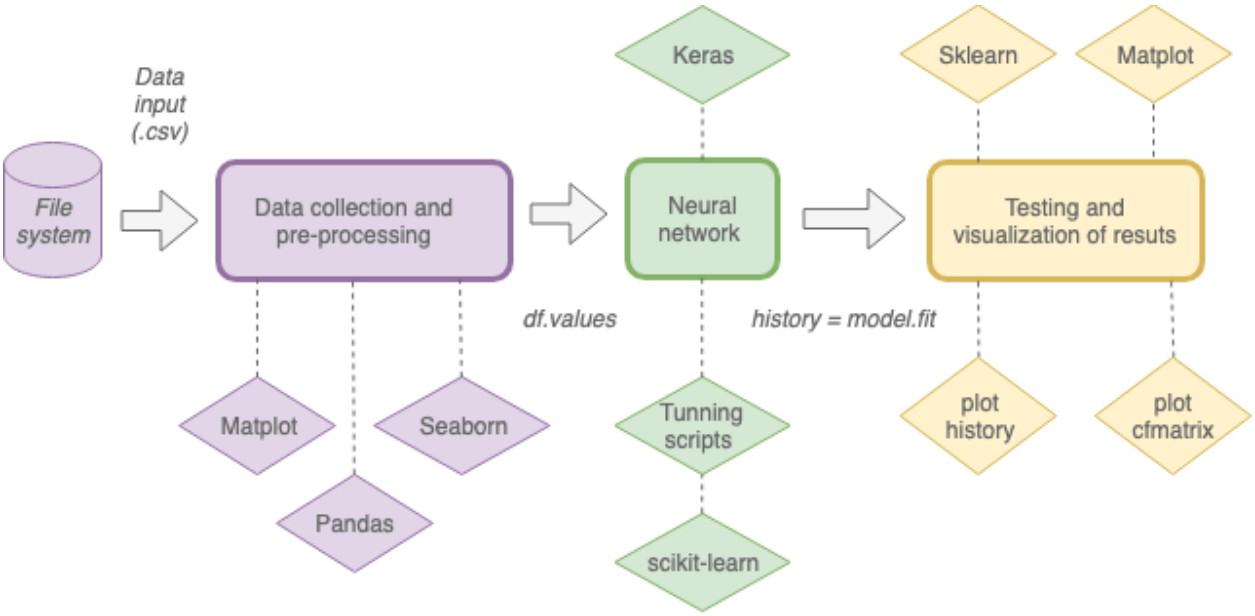


Figure 4.1: System overview

each of the modules makes use of different libraries that provide them with predetermined functions. In case a specific function needs to be implemented, that is not implemented by default by a library, the function will appear as a sub-module providing a specific functionality to its parent module. Likewise, it is important to understand each one of the interfaces required by each module, in order to transform the data flow to a format supported by it.

Starting from the left, the first part of the system will be composed by the input data. The selection of the appropriate dataset will meet the non-functional requirements #5, #6 , #7 and #8 and will be presented as a Comma-Separated Values (CSV) file to the system. This input information will feed the first module.

The Data collection and pre-processing module will be in charge of modifying the data to be clean and ready for the next one. Also, different simple statistical analysis will be performed by this module in order to understand possible relations between variables. This module will be in charge to fulfill functional requirements #1 and #2, and non-functional requirements #7.

In the following module the development of the neural network will be conducted. Parameters such as the number of layers, activation function or number of neurons in the network will be decided as it was stated in requirement #13. This module will represent the core of our system. It will be based on the results obtained from the first module and will generate a result that will be interpreted later by the last part of the system. It is designed to meet

functional requirement #3 and non-functional requirements #9, #13 and #10.

The last module will allow us to evaluate the performance of the system. Therefore, it will be powered by the results obtained in the previous module, as it was presented on requirements #11 and #12. These findings will serve to improve the proposed solution and detect possible issues on the performance of the model, being also in compliance with functional requirement #4. At the same time, as defined in the non-functional requirements #11 and #12, the implementation of the entire architecture will be programmed in Python and using the Keras library.

As it has been explained previously through each of the modules, the decision of this architecture is justified by the fulfillment of the requirements presented in section 3.6. At the same time, it will also be based on a literature review of several machine learning projects, where it has been observed that most of the projects address each of the modules presented above: data preparation, model implementation and testing [94] [95] [96].

In the following subsections each of the interfaces, functionalities and components of each module are described in more detail.

4.2 Required data input

One of the first requirements for the development of our system will be to find a good set of data to work with. In Section 3.6 different requirements about the ideal dataset can be found, as non-functional requirements #5, #6, #7 and #8. So, the outcome exposed in this section has been the result of an intensive search based on those parameters.

As a reminder of the requirements defined in previous chapters, it should be noted that: the chosen dataset must contain the maximum number of features defined in Tables 3.2 and 3.1; it must be labelled, with a field representing the diagnostic result; and it must be large enough, containing at least 9.000 entries.

The Alzheimer's Disease Neuroimaging Initiative (ADNI) database collects, validates and utilizes data, including MRI and PET images, genetics, cognitive tests, CSF and blood biomarkers as predictors of the disease. It is one of the most common datasets used in machine learning projects in the field of Alzheimer's disease, being present in a lot of projects [97–101]. In the same manner, The Open Access Series of Imaging Studies (OASIS) database can be commonly found as data entry for many artificial intelligence projects [102–105].

These databases contain information on a large number of patients, offering the result of their diagnosis. But they will not be valid for the present project since they are only based on different external medical tests - such as MRI, CSF examinations or PET scans - not fulfilling the features requirement demanded in Section 3.6.

At this point, narrowing the search to the project features requirement, all databases that are not oriented to merely clinical data have been discarded. Based on the research conducted by Arthur W. Toga, Priya Bhatt, and Naveen Ashish about the databases currently available on Alzheimer's disease [106] , The Critical Path for Alzheimer's Disease (CPAD) and The National Alzheimer's Coordinating Center (NACC) databases can be highlighted as the two closest to the requirements outlined above. The CPAD is a unified clinical trial database with primary focus is on AD, created and maintained by the Critical Path Institute of London. CPAD has a mission to develop new technologies and methods to accelerate the development and review of medical products for neurodegenerative diseases. On the other hand, the NACC database collects data from different Alzheimer's disease centers across the United States maintaining a large relational database of standardized clinical research data. Both of them are characterized by containing a lot of the features required by the Section 3.6. But even that the CPAD database contains features about the demographic data, medical history of the patient or MMSE cognitive results, it does not contain information about the patient lifestyle as smoking habits, alcohol abuse or years of patient's education. In addition, the NACC database contains around 35000 patients against the 7000 stored by CPAD database, and the number of machine learning projects implemented with those data inputs is significantly larger in the case of NACC database [107].

As a result of the analysis performed through different databases that can fit into the system requirements, the NACC will be the selected one.

4.2.1 Description of the NACC Database

The chosen dataset belongs to NACC of Washington University. The National Alzheimer's Coordinating Center was established by the National Institute on Aging/NIH in 1999 to facilitate collaborative research. Using data collected from the Alzheimer's Disease Centers (ADCs) across the United States, NACC has developed and maintains a large relational database of standardized clinical and neuropathological research data. In partnership with The Alzheimer's Disease Genetics Consortium (ADGC), The National Centralized Repository for Alzheimer's Disease (NCRAD), and The National Institute on Aging Genetics of

Alzheimer's Disease Data Storage Site (NIAGADS), NACC provides a valuable resource for both exploratory and explanatory Alzheimer's disease research. The NACC database is made up of three research datasets defined as follows: The Minimum Data Set (MDS), The Uniform Data Set (UDS) and The Neuropathology Dataset (NP).

The collection of data started in 1984 with the creation of the Minimum Data Set (MDS). It was the first attempt on the creation of the dataset ending on 2005 with implementation of the UDS. The Uniform Data Set (UDS) collects all the data submitted by the ADCs from September 2005. The centers use the UDS Forms to collect standardized clinical data from subjects who are evaluated on an approximately annual basis. Since 2005, the UDS forms have undergone two major revisions to reflect advances in the science and incorporate new diagnostic criteria. To combine data across the three versions, a Researcher's Data Dictionary (RDD) was created. From the beginning of February 2012, the UDS contains a Frontotemporal Lobar Degeneration (FTLD) Module. It is composed by detailed clinical information related to frontotemporal lobar degeneration but it is only provided by a subset of UDS subjects. Also, from August 20017, the UDS also contains a Lewy Body Disease (LBD) module. At centers participating in this voluntary effort, subjects with suspected LBD and/or controls are evaluated with the LBD Module in addition to the standard UDS Forms. In adittion, a subset of UDS subjects have one or more MRI available to download as zip files. And, a very small subset of UDS subjects also have one or more amyloid PET scans available to download. It is important to understand that only a minor part of all the UDS users provides also data to the FTLD and LBD modules or MRI images.

The last module is the Neuropathology Dataset (NP). It contains subjects who have died and consented to autopsy. The NP data-collection form has undergone numerous revisions to reflect advances in the science and incorporate new diagnostic criteria. To combine data across versions, also a RDD was created.

4.2.2 Analysis of the required characteristics

In order to validate our dataset, each of the requirements defined in Chapter 3 will be referred to, seeing if the NACC datasets complies with them.

One of the first requirements refers to the search for a data set containing the maximum relevant features defined in the 3.1 and 3.2 Tables.

As explained above, the NACC dataset is very broad and complete and can be very

difficult for a new user to understand. For this reason, the University of Washington offers researchers a consultation service and help in the creation of a personalized dataset. This service has been of great help, as a specialist in the database has provided help via e-mail in the choice of the most appropriate parameters.

The database chosen was the UDS dataset, as it contains most of the features stated on the non-funcional requirement #8. At the same time, the possible incorporation of the NP dataset was also interesting as it offers an error-free diagnosis based on autopsies. But finally the use of NP dataset was discarded, since although it can offer a quality diagnostic variable, not all the required clinical data of the living patient can be related.

As a result, the input information of the system will be based on UDS dataset as a combination of the required features and their corresponding clinical diagnostic variables. In Appendix D, each of the final features of the dataset can be observed, being able to notice a high match with the desired features defined on the Section 3.6. The data source will sum a total of 25 variables, a unique identifier for the patient and the result of his diagnosis. Again, it is essential to emphasize the great support offered by the NACC of Washington, since an extensive documentation about the meaning of each one of the variables and their possible values has been provided through the website.

The other requirements that have been taken into consideration for validating the dataset are: the presence of labelled results, to be a balanced dataset and to have a sufficiently large number of records.

Referring to the requirement of tagged data, it can be observed that the information obtained in NACC dataset offers fields with the results of the clinical diagnosis. These variables provide a positive or negative result in the presence of the disease, being perfectly adequate to what was being looked for. But the presence of both cases will not be enough to fullfill the non-functional requirement #7, the dataset should be also balanced. Although the NACC dataset provides data on the two types of patients, the number of cases of each of them is not fully balanced. This will not be a problem, as it will be something to bear in mind in the implementation of the system to process the information properly to achieve a balanced data set and correctly meet the requirement.

Finally, the non-functional requirement #5 makes reference to the necessary large amount of data. This condition will also be covered, as the total amount of information gathered will be 30990 records representing each of them, an unique patient .

4.3 Data collection and pre-processing module

The first of the modules that compose the system will be in charge of processing the input dataset. As input to the module there will be unprocessed information in tabular form, ideally in Comma-Separated Values (CSV) format. As it has been exposed at the beginning of the chapter, the goal of this module will be to meet functional requirements #1 and #2, and non-functional requirements #7. The pre-processing and cleaning of this input flow of information will be an essential task, since without quality input data the effectiveness of a future automatic learning would be drastically affected. This will allow to fulfill the requirements exposed above, processing the information, cleaning the data and balancing the dataset. The first step will be to carry out a small exploratory analysis of the dataset, that is to say, to apply statistical readings and modifications in the variables together with some visualizations to understand a better the data available. Below are presented each of the problems that must be addressed in this analysis and pre-processing, providing reliable data input to the next module of the system:

- Identify and treat possible null values.
- Analyze the individual value ranges for each field, identifying. possible erroneous or anomalous entries
- Check the number of cases for every classification class, and balance the dataset if necessary.
- Mix the information, randomly presenting each of the user entries. This will avoid, for example, locating all non-AD cases at the beginning of the dataset and all entries for patients with AD at the end.
- Normalize the input data if necessary, establishing a range of values or similar scale for all variables.

In order to define each of the previously described steps, the procedures most commonly used by the majority of machine learning projects in their dataset cleaning and preparation phase have been used as a reference [108] [109]. In addition, such steps are the result of meeting the requirements described in section 4.1 for this module.

All the functionalities of this module, like the rest of the system, will be programmed in Python as it was stated on Section 3.6 through the non-funcional requirement #14. The

implementation of the functionalities are supported by three libraries: Pandas, Matplotlib and Seaborn. Seaborn and Matplotlib will be in charge of data visualization, leading to the analysis of possible relationships between variables. Pandas library, will enable to import the tabular information in a dataframe, allowing to make all the analysis and preprocessing of the information in a fast and simple way.

4.4 Neural Network

Once the input data has been processed and analyzed, the information is ready to be computed by a neural network. The neural network will be in charge of learning from these data to be able to make predictions on new input data in the future. To do so, the implementation of a MLP will be addressed, meeting the non-functional requirement #9 exposed on Section 3.6. One of the first things to be done in this module will be the separation of the dataset into several subsets. These sets will be: a training set, a validation set and a test set. Generally speaking, the first of the subsets will serve to train our network, the second to choose the most appropriate hyperparameters for the model and the last to test the results. The selection of the most appropiate features will be an important phase of the project, since it will be the basis of the fullfilment of the non-functional requirement #3 offering accurate results for future predictions. The implementation of the present module will be programmed in Python and supported by Keras library as it was stated on non-functional requirements #14 and #15.

Once that the flow information obtained from the previous module has been divided into different sets, it is time to create the structure of the neural network. It will be based on a Sequential model, which can be defined as a stack of layers. As it has been mentioned before, the selection of this specific model will be aimed by the requirement about the MLP architecture. Thus, different layers will be added specifying their activation functions, number of neurons and input data. When the structure is created, the model will be compiled. Compilation requires specifying a set of parameters: the optimization algorithm to be used to train the network and the loss function used to evaluate the network that is minimized by the optimization algorithm.

When the compilation of the model has been done, the network will be trained. To do so, training data must be provided as input. Validation data can also be provided in order to analyse the evolution of the training. As parameters, it will be necessary to define the batch size (number of records to use in each iteration) as well as the number of epochs (how

many times is going to cross the whole set of data to train).

All the parameters that has been mentioned in the compiling and training phase should be tuned beforehand, as the best activation function, optimization algorithm or batch size fulfilling non-functional requirement #13. Doing it, accurate results will be achieved and the fulfilment of the non-functional requirement #3 will be made.

At this point, the neural network has been trained and its behavior should be evaluated with new input data. For this purpose, each of the metrics specified above in the compilation of the model will be processed by the following module. Figure 4.2 summarizes all the steps to be implemented in this module. In this figure it can be seen that the required input data will be the dataframe values of the previous module, and the output generated will be the training response of the model. This response will be analyzed and processed by the next section of the system.

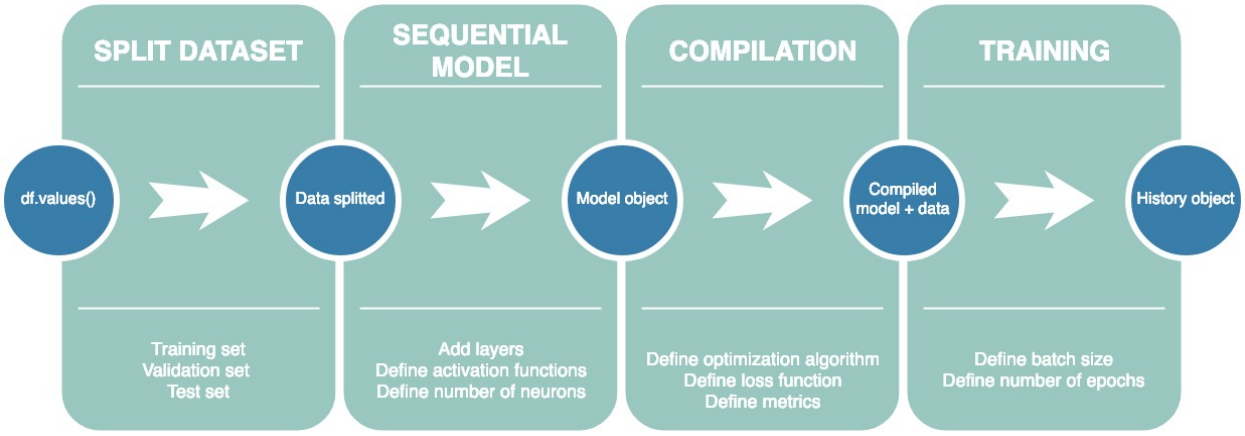


Figure 4.2: Design of the implementation of the neural network

4.5 Testing and visualization of results

Although the most commonly used metric for model evaluation is its accuracy, the purpose of this last module will be to present a deeper and more solid analysis of the model as it was stated on functional requirement #4 and specified through non-functional requirements #11 and #12. Also, the evaluation of the model through these different metrics will allow to fulfill requirement #3.

One of the first metrics to be represented by this section of the system will be its confusion

matrix. This matrix will allow to analyze the types of correct and incorrect predictions made by the model in each of its categories. The tool enables the evaluation of false positives, when the result is incorrectly classified as positive when it turns out to be negative, as well as false negatives, when the result is positive and is incorrectly classified as negative. The analysis of the matrix will be very meaningful given the scope of this project. Since there can be many ethical and moral discussions about the importance of false positives or negatives in the detection of Alzheimer's disease. Also the creation of the Receiver operating characteristic (ROC) curve of the model will be really valuable, since it summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.

On the other hand, as a result of the model metrics defined in its computation the final result of its accuracy can be obtained. This will be a static result, corresponding to the final result of the whole training. But it will also be interesting to be able to analyze the evolution of the model throughout the training phase. The easiest way to analyze this progression is to use a graph. This graph will represent the evolution of the error throughout the training periods. In this way it is possible to see if, for example, the model can tend to continue learning if we increase the number of times, or if, on the contrary, the learning had reached its limit. Therefore it will also be interesting, and the objective of this module, to represent these learning curves for each of the subsets of data: training and validation. Plotting the evolution also in the validation set allows to also evaluate the generalization capacity of the model. This will allow us to discuss possible cases of over-fitting or under-fitting as it was mentioned on non-functional requirement #12.

4.6 Summary

As a result of everything explained in the previous points, this section offers a recapitulation of each one of the functionalities to be implemented by the solution in order to meet the requirements exposed in Chapter 3. To do this, it has been decided to use a table that gathers each of the functionalities required in each module, as well as their input and output data. In table 4.1 all this information can be observed, and it will be the basis and guide for the implementation of the system.

	Input	Functionalities	Output
Data collection and preprocessing	Unprocessed dataset (.csv)	– Exploratory analysis: nulls values, valye ranges, balance data, shuffle and normalization – Analysis of correlation between variables – Visualization of data	Processed pandas dataframe
Neural network	Dataframe.values()	– Split the data – Create the model – Tune hyperparameters – Compile the model – Train the model	History and metrics of the model
Testing and visualization of results	Metrics provided on compilation and history object of the model	– Confusion matrix, ROC – Learning curves	Analysis of the solution

Table 4.1: Summary table of each of the modules that compose the system

Chapter 5

Implementation

This chapter will describe the implementation of the system based on the conceptual design presented in Chapter 4. The implementation will be always oriented to meet the requirements exposed on Chapter 3, aiming to be useful to answer the research question presented at the beginning of the report on Section 1.1.

The chapter will be composed as the explanation of each of the steps performed in the implementation of the data collection and pre-processing module and neural network one. The output of this implementation will be used as input of the last module, Testing, which will be exposed on Chapter 6.

5.1 Data collection and pre-processing

Once the dataset is available in CSV format, different aspects of it should be analyzed in order to conclude that the dataset is ready to be input information for the neural network. As it has been stated on Section 4.3, the first aspect that has been approached is a simple statistical exploration of each one of the fields. To do this, the CSV file has been imported into a dataframe. This has been done through the Pandas library.

In the quick analysis of each variable is included, for example, the identification of maximum values, minimum values or calculation of the mean. Thanks to this short exploration, it has been possible to observe that most of the values of each variable are more or less within the same range. Being the variables with more disparate values the MMSE, from 0 to 30, years of education, from 0 to 36 and age of the patient, from 18 to 109. The rest of

values are in a range between 0 and 10, which made us discard the need to normalize the values in a first instance.

Next, the presence of null values has been analyzed. In the case of the present project, the dataset does not contain null values, so it will not be necessary to deal with this problem. This is probably due to the high quality of data collection required by the NACC dataset.

The possible correlation between variables was then analysed. In Figures 5.1, 5.2 and 5.3 there are three graphs representing these investigations. In order to perform them, the Matplot and Seaborn libraries have been used.

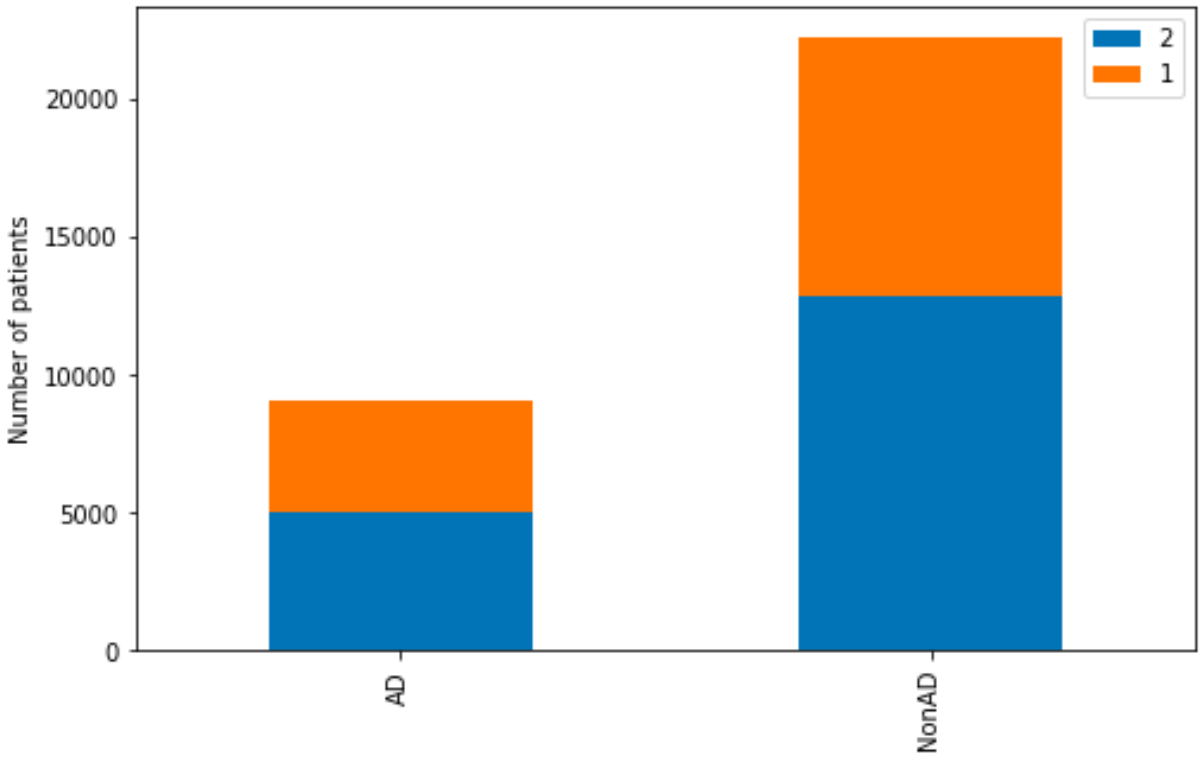


Figure 5.1: Relation between gender and AD variables

Figure 5.1 has studied the possible relationship between the different gender of the patients and their tendency to have or not have the disease. And as can be observed, the graph indicates that women are more likely to develop the dementia than men. Figure 5.2 shows the relationship between the score obtained in the MMSE and the result of the diagnosis. Clearly, non-AD group got much more higher MMSE scores than AD group. Finally, Figure 5.3 represents the correlation between patient age and the disease. There is a higher concentration of 70-85 years old in the AD patient group than those in the non-AD patients, which may reveal certain critical age ranges for the develop of the Alzheimer disease.

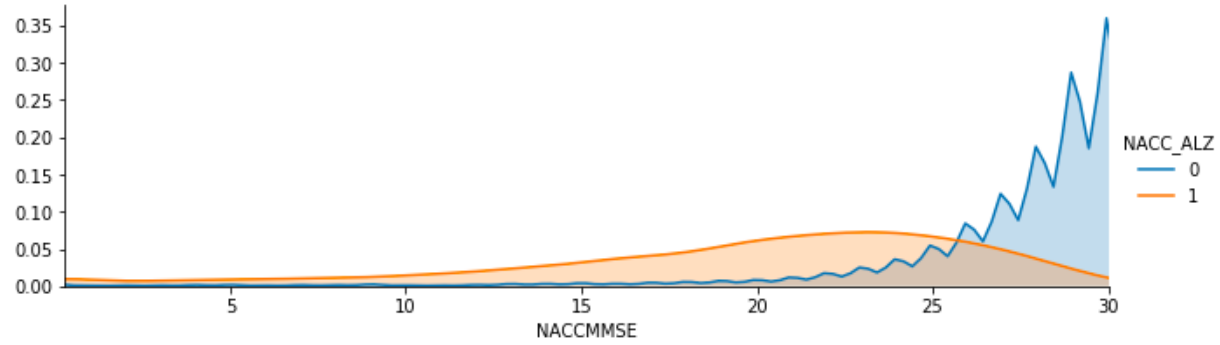


Figure 5.2: Relation between MMSE and AD variables

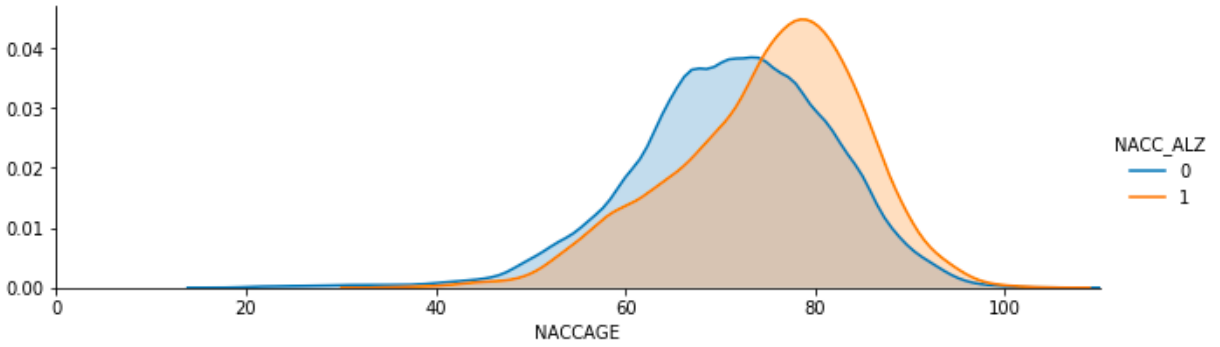


Figure 5.3: Relation between age and AD variables

One of the requirements established in Chapter 3 is to have a balanced data set. To do this, the same number of samples of each class will be taken. Even though this means losing number of records, the total amount of information is still more than enough. It is important to bear in mind that when the two subsets of data (AD cases and non-AD cases) are put together again, they will not be placed randomly. All cases in one category would be presented first and then the other. This is not positive for the future implementation of the model, since seeing only cases of one category for a long time does not allow the model to learn. Therefore, it will be important in the last place to shuffle the information. In the Figure 5.4 it can be observed the entire process mentioned above, that is, the balancing and subsequent shuffling of the records.

As a result, the implementation of the module satisfies all the required functionalities exposed on the Chapter 4 - Conceptual Design. And, at the same time, following those guidelines the functional requirements #1 and #2 and non-functional requirements #7 can be considered successfully implemented.

```
df0 = df[df['NACC_ALZ'] == 0]
df1 = df[df['NACC_ALZ'] == 1]

df0 = df0.sample(8500)
df1 = df1.sample(8500)

df = pd.concat([df0, df1])

#Shuffle
df = df.sample(frac=1).reset_index(drop=True)
```

Figure 5.4: Python code to balance and shuffle the dataset

5.2 Neural Network

As a reminder of the goal of this module, a reference can be made to Section 4.4. That section exposed the conceptual design needed to be implemented by the neural network module in order to complain with the requirements presented on Section 3.6.

The implementation of the module starts retrieving the information processed by the data collection and pre-processing part of the system. This data will be obtained from the Pandas dataframe making use of the *dataframe.value* method.

Afterwards, as defined in Section 4.4, the dataset will be divided into two parts: training and test. It has been partitioned by the *train test split* method resulting in 20% test data and 80% training data. This last subset will be split again, giving rise to the validation data set, which will be composed by a 10% of the training subset.

The first issue that should be addressed for the creation of the model will be the appropriate choice of each of the model hyperparameters. Regarding its architecture, the most adequate number of layers, neurons and their respective activation functions have to be defined. Also, the most suitable optimizer should be chosen in order to compile the system. And finally, it will be necessary to establish the most optimal number of epochs and batch size to optimize the future training of the model.

In order to be able to choose these parameters properly, as it can be seen on 5.1, four independent scripts have been implemented based on the research done by PhD. Jason Brownlee [110]. Inside of each script, different models based on different values of these parameters have been created, presenting the result of the accuracy of all of them. The implementation done by these scripts is the result of the analysis exposed on Section 3.3,

	Parameter to be tuned	**Options presented**	**Result**
Script_1	Number of neurons in the hidden layer	10, 15, 20, 25, 30, 35, 40, 45, 50	25
Script_2	Activation function	Softmax, Relu, Tanh, Sigmoid	Tanh
Script_3	Optimizer	SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam	Adamax
Script_4	Batch size and number of epochs	– Batch size: 800, 2000, 5000, 10000, 13000, 15000 – Epochs: 600, 800, 1000, 1200, 1400, 1600	Batch size of 5000 and 1600 epochs

Table 5.1: Table exposing each of the scripts implemented in order to tune the hyperparameters of the model

where it was exposed that the most appropriate manner to choose the parameters of a model was to directly try them. Therefore, these scripts will allow us to choose the most suitable number of neurons, their activation function, number of epochs, batch size and optimizer.

But before trying to tune all those complex model hyperparameters, some assumptions have been resulted from the analysis done on Section 3.3:

- The number of hidden layers will be one, since at it has been exposed on the Chapter 3, in most of the cases has been exposed as enough to achieve a good accuracy.
- The number of neurons in the input layer will be 26, being the equivalent to the number of input variables in the data [91].
- The number of neurons in the output layer will equal to the number of outputs associated with each input, 2 [111].
- Considering that the model is oriented to a classification problem, as it has been analyzed also in Section 3.3, the Sigmoid function will be the selected one for the output layer.

In figure 5.5 the output of one of the scripts used to tune the rest of the model hyperparameters can be seen, where it is indicated that the optimizer with the best accuracy result

for our model is the Adamax optimizer. At the same manner, as a result of the rest of the scripts the choice of the remaining hyperparameters of the model has been reached, being:

- 25 neurons in the hidden layer
- Tanh as activation function for the hidden layer
- 5000 as batch size
- 1600 number of epochs

The reason behind the implementation of these independents scripts will be to obtain higher accuracy results. This will be an important goal for the system, since it will allow to fulfill the requirement #3 exposed on Section 3.6. This has been implemented by means of the Grid search model hyperparameter optimization technique. This technique can be used though the GridSearchCV class inside of the Scikit-learn library. When constructing this class, a dictionary of hyperparameters to evaluate in the *param grid* argument have to be provided. By default, accuracy is the score that is optimized, but other scores can be specified in the score argument of the GridSearchCV constructor. The GridSearchCV process will then construct and evaluate one model for each combination of parameters and cross-validation is used to evaluate each individual model with 3-folds as default [112]. The complete Python code of all these scripts can be found on Appendix F, G, H and I.

```
Best: 0.826889 using {'optimizer': 'Adamax'}
0.791833 (0.001650) with: {'optimizer': 'SGD'}
0.817667 (0.005079) with: {'optimizer': 'RMSprop'}
0.815111 (0.005780) with: {'optimizer': 'Adagrad'}
0.819056 (0.004502) with: {'optimizer': 'Adadelta'}
0.825333 (0.002361) with: {'optimizer': 'Adam'}
0.826889 (0.003403) with: {'optimizer': 'Adamax'}
0.821667 (0.002453) with: {'optimizer': 'Nadam'}
```

Figure 5.5: Output of the optimizer tuning script

Once that the model parameters have been selected, it is turn to build the core of the neural network. To do so, the guidelines exposed on Section 4.4 have been followed. Firstly, in order to fulfill the requirement #9 and #11 a Sequential model through the Keras library has been built. This model will be characterized by being a MLP composed by one input layer, one hidden layer and one output layer, as it can be seen on Figure 5.6.

Before training a model, it will be needed to configure the learning process, which is done via the compile method. In this method the optimizer previously defined, Adamax

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 25)                650
_________________________________________________________________
dense_2 (Dense)              (None, 25)                650
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 26
=================================================================
Total params: 1,326
Trainable params: 1,326
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.6: Summary of the model composed by: the input layer, one hidden layer and the output layer

optimizer, will be provided as parameter. Also the loss function that the model will try to minimize and the metric to evaluate the process.

Once that the model has been compiled, the neural network is ready to be trained. To be able to train the model, two last parameters should be provided: batch size and number of epochs. These two parameters will be really important in the fulfilment of the requirement #3, since at it will be seen on Chapter 6, they will affect directly to different the over-fitting or under-fitting situation of the model.

As a resume, in the Figure 5.7 it can be seen the creation, compilation and training of the model based on all the hyperparameters previously selected. With the implementation of this module, the ideas exposed on Section 4.4 will be satisfied and the requirements #3, #9, #11 and #12 executed.

```
# create model
model = Sequential()
model.add(Dense(25, input_dim=25, activation='tanh'))
model.add(Dense(25, activation='tanh'))
model.add(Dense(1, activation='sigmoid'))
```

```
# Compile model
model.compile(loss='mean_absolute_error', optimizer='Adamax', metrics=['accuracy'])
```

```
# Fit the model
history = model.fit(X_train, Y_train, validation_split=0.1, epochs=1600, batch_size=5000, verbose=2)
```

Figure 5.7: Creation, compilation and training of the model

Chapter 6

Testing

This chapter will explain and expose the results obtained from the last module of the system which allows to answer the research question of Section 1.1 in the conclusions Chapter. Thus, this module will be in charge of the evaluation and testing of the model. It will follow the guidelines defined on Section 4.5 and it will aim to fulfill requirements #3, #4, #11 and #12. At the beginning of the chapter the evaluation of the accuracy and the confusion matrix of the model will be presented. After, the ROC curve will be illustrated, exposing his results. In addition, at the end of the chapter the learning process of the model will be analyzed in order to study the presence or not of over-fitting or under-fitting problems.

6.1 Accuracy and confusion matrix

As a reminder of the output obtained in the last module, reference can be made to Section 5.2. In this section, the model was created, compiled and trained obtaining a system ready to make future predictions about new data.

But before making new predictions, one of the most immediate metrics to evaluate will be its accuracy. To do this, the model will be evaluated with the test dataset. This group of data was divided at the beginning of the code, and it is important to emphasize that it has never been before seen by the model. Figure 6.1 illustrates how the model has been evaluated based on these test data input, to then obtain its performance according to the accuracy metric. The result obtained is 82.61%, being the percentage of predictions that the model made correctly with respect to the total number of predictions to be made.

```
# evaluate the model
scores = model.evaluate(X_test, Y_test)
```

```
3600/3600 [==============================] - 0s 11us/step
```

```
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
acc: 82.61%
```

Figure 6.1: Python code to evaluate the accuracy of the model

But at it was stated on requirement #11, the accuracy of a model will not be sufficient to evaluate properly his performance. Due to his, also the confusion matrix has been represented. The confusion matrix gives an insight not only into the errors being made by a classifier, but more importantly the types of errors that have being made. To do so, *sklearn.metrics* offers a simple and fast manner to represent the confusion matrix of a model based on his predictions. As it can be seen on Figure 6.2, the correct outputs of the predictions should be provided. In this manner, the confusion matrix can evaluate how well the predictions have been made. Also, an independent function to plot the confusion matrix has been implemented. This function, *plot_confusion_matrix*, can be seen on Appendix E where all the Python code used to implement the system is presented [113].

```
In [93]: cnf_matrix = confusion_matrix(Y_test, y_pred2)
         plot_confusion_matrix(cnf_matrix, classes=target_names, normalize=False,
                               title='Confusion matrix')
```

Figure 6.2: Python code to represent the confusion matrix of the model

As a result, the Figure 6.3 illustrates the confusion matrix of the system. Analyzing the content of the matrix, the upper left quadrant represents the number of correct predictions of subjects who do not have AD. This is referred as the True Negative Rate (TNR), being 1486 cases in our system. The upper right quadrant shows the number of positive predictions when the value should be negative. These errors are called False Positive Rate (FPR), being in our model 309 cases. In the case of the bottom right quadrant, the number of predictions of people with AD are represented. Being 1488 cases which are called True Positive Rate (TPR). And finally, the number of negative predictions when the value would really have to be positive is represented in the bottom left quadrant. These errors are referred to False Negative Rate (FNR), being in our model 317 cases.

Fortunately, the number of incorrect predictions is much lower than the correct ones.

Within these incorrect predictions, it should be noted that there are more false negatives than false positives. This result will be something to take into account in possible users of the system. Since it will not mean the same morally and ethically to diagnose by mistake a person who does not have AD, than not to diagnose AD to a patient who really suffers it.

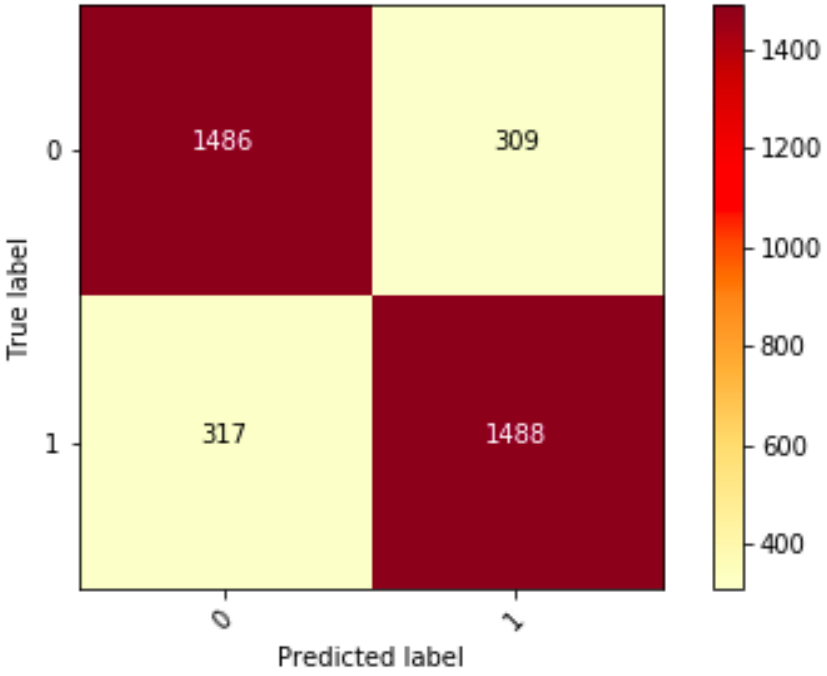


Figure 6.3: Confusion matrix of the model

6.2 Receiver operating characteristic (ROC)

The classifier's ability to avoid a false classification can be measured with the Receiver operating characteristic (ROC), created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, or recall and the false-positive rate is also known as the fall-out. Thus, the ROC chart is a two-dimensional graph in which the sensitivity is drawn on the Y-axis, or vertical, and the fall-out on the X-axis, or horizontal.

The Area Under the Curve (AUC) will be an additional measure that indicates the two-dimensional area below the complete ROC curve. One model whose predictions are 100% incorrect has an AUC of 0.0; the other whose predictions are 100% correct has an AUC of

1.0. This measure will be very useful when several models are to be compared, since trying to visually confront two very similar ROC curves can sometimes be complex.

To plot this ROC curve, the *sklearn.metrics* library has been used again. Through this library, as it can be seen on Figure 6.4, the *roc_curve* and *auc* functions can be used to obtain all the necessary values to build the graph.

```
false_positive_rate, true_positive_rate, thresholds = roc_curve(Y_test, y_pred)
roc_auc = auc(false_positive_rate, true_positive_rate)
```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")

plt.title('Receiver Operating Characteristic')

plt.plot(false_positive_rate, true_positive_rate, 'darkorange',
label='Deep neural network  ROC (AUC = %0.2f)' % roc_auc)

plt.plot([0,1],[0,1],color='slategray',linestyle='--', label = 'Random guess')
plt.legend(loc='lower right')



plt.show()
```

Figure 6.4: Python code to generate the Receiver operating characteristic (ROC) of the model

The best possible prediction method for a machine learning classifier would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). This (0,1) point is also called a perfect classification.

As a result, in Figure 6.5, the ROC curve of the model is illustrated. Since the curve is very close to the upper left corner of the graph, it can be concluded that the model correctly distinguishes between the two classes. In addition, the value of AUC is 0.87 being a value very close to 1.0.

6.3 Analysis of learning curves

All the metrics presented in previous sections have made it possible to evaluate the classifier's performance. While those metrics provide a static evaluation of the final performance of the model, this section on the contrary, will evaluate the evolution of the model throughout its

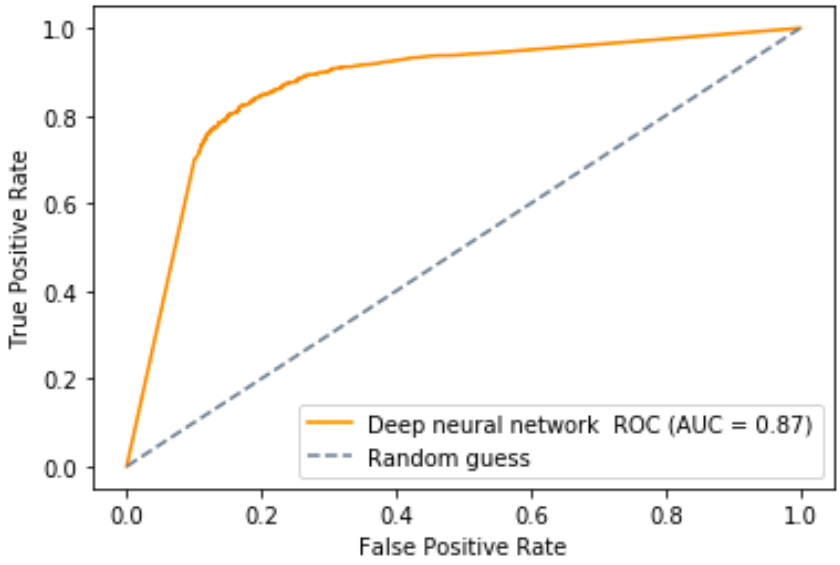


Figure 6.5: Receiver operating characteristic (ROC) of the model

training process.

As it was explained on Section 3.4, illustrating the learning curve of a model show his performance on training and validation sets as a function of the number of training iterations. This means that plotting the learning curve allows to analyze the evolution of the model, since it depends on the number of iterations.

As it can be seen through different machine learning projects [114–117], to illustrate the learning curve of a model is a powerful tool to diagnose over-fitting or under-fitting issues. The learning curve will expose the time or experience on the x axis and the improvement on the y axis. The manner to show the improvement of the model will be to plot its loss as the dimension of the y axis. It is important to understand that both, training and validation sets, should be illustrated. Each of them will provide different insights: the training dataset will give an idea of how well the model is learning and the validation dataset will show how well the model is generalizing. If the evolution of the two error scores is plotted, two curves will be obtained. These are called the learning curves, and in general terms, the goal will be to minimize the loss function since the best model is the one that has the least error.

The loss function of the model is illustrated in Figure 6.6. As it can be seen it decreases over the time, being almost constant between 1400 and 1600 epochs. This suggests that the learning process has reached its limit, meaning that even if the model is trained during more epochs the result will not be improved.

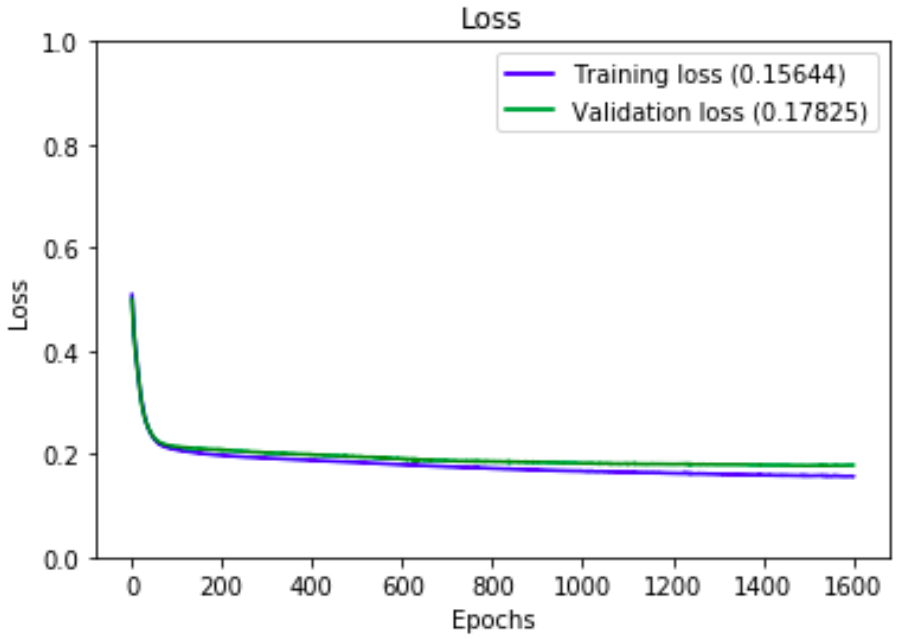


Figure 6.6: Loss function in respect to the number of epochs

This plot will present valuable insights into the amount of over-fitting or under-fitting in the model. In the case of a over-fitted classifier, the training loss will be low and the validation loss will be higher presenting a gap between both of them. The gap between the training and validation loss ratio indicates the amount of over-fitting: the bigger, the worse. Although this model might perform really well on the training data, its performance on the test data will be much worse. In other words, the model will suffer from high variance, which means that it will not be good at making predictions on data it has never seen before.

On the contrary, a model is said to under-fit when the training loss is high and the validation accuracy is high as well. A model is under-fitted when it performs poorly on the training data, and consequently, also on the validation data. The model is unable to capture the relationship between the input examples and the target values.

As a result, the goal will be to obtain a decreasing loss ratio along the number of iterations with an small gap between the validation and training sets. Taking a look again to the Figure 6.6 it can be seen that the graph does not present a huge gap between both set, and achieves a low loss ratio value. As such, it can be said that the model is working properly.

In order to be able to plot the graph, an independent function has been programmed in Python that will have as input object the result of the training of the model [118]. In this

training, as can be seen in Figure 5.7, a fixed percentage of the dataset was reserved as a validation subset allowing the comparison of the learning process of the two datasets through the learning curve. The implementation of the *plot_history* function can be consulted in the Appendix E [118].

Chapter 7

Conclusions

The objective of the present project was to answer the research question presented on Section 1.1. For this purpose, an analysis of the possible improvement on the diagnostic process of AD by means of deep learning techniques has been addressed through the implementation of the project. It is for this reason that, during this last chapter, an answer to this question will be exposed explaining its conclusions and future improvements of the system.

As a resume, the result of the analysis was to build a classifier able to discriminate data from patients with AD or not. To do so, the first step was to collect the data needed to train the classifier. But before that the data can be used, the data had to be cleaned and pre-processed. Once the data was ready to use to train a model, an appropriate model architecture had to be selected to build the classifier. Thus, a deep neural network was designed. In order to do so, a MLP architecture was selected. Different design considerations were tested, as the proper selection of the model hyperparameters or the analysis of the learning process. Finally, the performance of the model has been evaluated through his accuracy, confusion matrix and ROC curve based on previously unseen data.

In order to answer properly the research question, firstly, its sub-questions have to be addressed.

- *Which data will be necessary in order to train the system successfully?* The NACC dataset has been the one selected. It has been proved as being relevant, variate and big enough obtaining an accuracy of 82.61%.
- *Which is the most suitable architecture and parameters to achieve an accurate result?* But not only a good dataset is enough to achieve accurate results, as it can be seen

through the analysis and testing chapter the proper selection of the parameters of the model will be a key aspect to consider. The best architecture will differ from case to case, but given the dataset selected and the problem to be solved, the most suitable architecture has result in a multilayer percepton with one hidden layer and 25 neurons. The sum of model parameters which provides best accuracy results was Adamax as optimizer, Tanh as activation function, 5000 as batch size and 1600 as number of epochs.

- *What level of accuracy can be achieved?* As it has been seen on Chapter 6, the system achieved 82.61% of accuracy, being the percentage of predictions that the model made correctly with respect to the total number of predictions to be made. This result can be defined as successful, since the accuracy values obtained by similar projects on the field are between 60% and 90%.

- *What framework could be used to implement and test the selected model?* The programming language used to implement the system was Python, making use of Keras, Pandas, Matplot and Seaborn libraries. These libraries offer the best trade-off between performance and easy to use, being the best option for the present project.

To conclude, it has been demonstrated that it is possible to measure the probability of having AD by means of combining clinical history, lifestyle habits and cognitive examinations presenting an accuracy result of 82.61%. Moreover, it has been demonstrated that even though this type of data has been medically claimed to be insufficient to confidently diagnose the disease, adequate results can be obtained. Across the report it has be shown that this can be achieved with intensive data cleansing and the selection and design of the right model.

Moving back to the research question presented at the beginning of this project: *How could automatic learning techniques be used to improve the diagnosis of Alzheimer's disease?* His answer can be summarized in the implementation of a deep neuronal network that will be based on a correct selection of input data and proper design of the parameters and architecture of the model, achieving a precise discrimination, 82.61% accurate, between patients with AD or not.

7.1 Future perspectives

What has been presented in this project is a proof of concept. There are further improvements needed to create a real-time classifier that could be used as tool to help the diagnosis of AD patients. Here, some suggestions are presented that could be beneficial to the future system:

- The different influence of each of the features on the model accuracy can be analyzed, optimizing the data input.
- A more precise parameter selection can be done, considering the analysis of extra hyperparameters as the learning rate.
- In order to be closer to a real medical diagnosis procedure, the data that feeds the system can be complemented with MRI images or other bio-markers.
- To increase the complexity of the solution, and to be able to process images as data input, a future implementation of a CNN should be considered.
- In addition, the inclusion of data from different sources will improve the robustness of the model. In this manner, a robust transfer learning can be achieved, in which the model will transfer not only his performance but also robustness from a source model to a target domain.
- The business value of the system should be analyzed, evaluating different cases of use and adapting the solution to the final user needs. At the same time, aspects as the security, reliability or performance of the system must be considered as a priority in the case of a real use of the system with a medical purpose.

Bibliography

[1] *Dementia statistics - Alzheimer's diasease international*. URL: https://www.alz.co.uk/research/statistics.

[2] Christina Patterson. *World Alzheimer Report 2018 The state of the art of dementia research: New frontiers*. Tech. rep. Alzheimer's Disease International (ADI), London., 2018. DOI: 10.1111/j.0033-0124.1950.24{_}14.x. URL: https://www.alz.co.uk/research/WorldAlzheimerReport2018.pdf?2.

[3] J. Manyika et al. *Big data: The next frontier for innovation, competition and productivity*. Tech. rep. May. 2011. URL: https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Big%20data%20The%20next%20frontier%20for%20innovation/MGI_big_data_exec_summary.ashx.

[4] Warren W. Barker et al. "Relative Frequencies of Alzheimer Disease, Lewy Body, Vascular and Frontotemporal Dementia, and Hippocampal Sclerosis in the State of Florida Brain Bank". In: *Alzheimer Disease & Associated Disorders* 16.4 (Oct. 2002), pp. 203–212. ISSN: 0893-0341. DOI: 10.1097/00002093-200210000-00001. URL: https://insights.ovid.com/crossref?an=00002093-200210000-00001.

[5] Melonie Heron. "National Vital Statistics Reports Volume 67, Number 5 July 26, 2018". In: *National Vital Statistics Reports* 67.6 (2018), pp. 1–15. ISSN: 1551-8922. URL: https://www.cdc.gov/nchs/data/nvsr/nvsr67/nvsr67_06.pdf.

[6] Joseph Gaugler et al. "Alzheimer's Association Report". In: *Alzheimer's and Dementia* 11.3 (2015), pp. 332–384. ISSN: 15525279. DOI: 10.1016/j.jalz.2016.03.001. URL: http://dx.doi.org/10.1016/j.jalz.2015.02.003%20https://ac.els-cdn.com/S1552526015000588/1-s2.0-S1552526015000588-main.pdf?_tid=d7c35d3a-03bb-4b42-a93a-11fa491d3599&acdnat=1550351168_f7ed67e075417bdf021d44889920839c.

[7] James C. Vickers et al. “The cause of neuronal degeneration in Alzheimer’s disease”. In: *Progress in Neurobiology* 60.2 (2000), pp. 139–165. ISSN: 03010082. DOI: 10.1016/S0301-0082(99)00023-4. URL: https://ac.els-cdn.com/S0301008299000234/1-s2.0-S0301008299000234-main.pdf?_tid=7722ae13-f4d5-451f-a6e5-98ed91385196&acdnat=1550351534_af1b85fcb126558384dfef49b5240cbe.

[8] Richard J. O’Brien and Philip C. Wong. “Pathological Cascade”. In: *Archives of Medical Research* 12.3 (2012), pp. 173–189. ISSN: 01406736. DOI: 10.1146/annurev-neuro-061010-113613.Amyloid. URL: http://dx.doi.org/10.1016/j.arcmed.2012.11.008%5Cnhttp://www.ncbi.nlm.nih.gov/pmc/articles/PMC3174086/%5Cnhttp://dx.doi.org/10.1016/S0140-6736(12)62145-X.

[9] J. P. Brion. “Neurofibrillary tangles and Alzheimer’s disease”. In: *European Neurology* 40.3 (1998), pp. 130–140. ISSN: 00143022. DOI: 10.1159/000007969.

[10] Clifford R. Jack et al. “Serial PIB and MRI in normal, mild cognitive impairment and Alzheimers disease: Implications for sequence of pathological events in Alzheimers disease”. In: *Brain* 132.5 (2009), pp. 1355–1365. ISSN: 14602156. DOI: 10.1093/brain/awp062. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2677798/pdf/awp062.pdf.

[11] A Klug. “Topographical relationship between f8-amyloid and tau protein epitopes in tangle-bearing cells in Alzheimer disease”. In: 87.May (1990), pp. 3952–3956. URL: https://www.pnas.org/content/pnas/87/10/3952.full.pdf.

[12] Guy M. McKhanna et al. “The diagnosis of dementia due to Alzheimer’s disease: Recommendations from the National Institute on AgingAlzheimer’s Association workgroups on diagnostic guidelines for Alzheimer’s disease”. In: 7.3 (2011), pp. 263–269. DOI: 10.1016/j.jalz.2011.03.005.The. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3312024/pdf/nihms363310.pdf.

[13] Joseph L Price and John C Morris. “Tangles and Plaques in Nondemented Aging and “ Preclinical ” Alzheimer ’ s Disease”. In: (1999), pp. 358–368. URL: https://pdfs.semanticscholar.org/c9b0/c1a767d9b056c3d3ec30ff106cf54b1b8932.pdf.

[14] Clifford R Jack Jr et al. “Introduction to Revised Criteria for the Diagnosis of Alzheimer’s Disease: National Institute on Aging and the Alzheimer Association Workgroups”. In: 7.3 (2011), pp. 257–262. DOI: 10.1016/j.jalz.2011.03.004.Introduction. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3096735/pdf/nihms284298.pdf.

[15] O Hansson et al. "Prediction of Alzheimer's Disease Using the CSF Aβ42/Aβ40 Ratio in Patients with Mild Cognitive Impairment". In: 23 (2007), pp. 316–320. DOI: 10.1159/000100926.

[16] R. Zinkowski et al. "CSF phosphorylated tau protein correlates with neocortical neurofibrillary pathology in Alzheimer's disease". In: *Brain* 129.11 (2006), pp. 3035–3041. ISSN: 0006-8950. DOI: 10.1093/brain/awl269. URL: https://watermark.silverchair.com/awl269.pdf?token=AQECAHi208BE49Ooan9kkhW_Ercy7Dm3ZL_9Cf3qfKAc485ysgAAAkYwggJCBgkqhkiG9w0BBwagggIzMIICLwIBADCCAigGCSqGSIb3DQEHATAeBglgh
qQ4EU7t8PPAgEQgIIB-ZvrZmJI6e7hKVPqkN4ETZcgH6Ii5J2C0N1hI-J0VVXnNobN.

[17] Katharina Buerger et al. "No correlation between CSF tau protein phosphorylated at threonine 181 with neocortical neurofibrillary pathology in Alzheimer's disease [2]". In: *Brain* 130.10 (2007), pp. 180–181. ISSN: 14602156. DOI: 10.1093/brain/awm140. URL: https://watermark.silverchair.com/awm140.pdf?token=AQECAHi208BE49Ooan9kkhW_Ercy7Dm3ZL_9Cf3qfKAc485ysgAAAkUwggJBBgkqhkiG9w0BBwagggIyMIICLgIBADCCAicGCSqGSIb3DQ
LjP_CVwGD7wVhbPjbgzJAj6yWe-o49LUqJmLELPoglhXDoB.

[18] Sebastiaan Engelborghs et al. "No association of CSF biomarkers with APOEϵ4, plaque and tangle burden in definite Alzheimer's disease". In: *Brain* 130.9 (2007), pp. 2320–2326. ISSN: 14602156. DOI: 10.1093/brain/awm136. URL: https://watermark.silverchair.com/awm136.pdf?token=AQECAHi208BE49Ooan9kkhW_Ercy7Dm3ZL_9Cf3qfKAc485ysgAAAkQwggJABgkqhkiG9w0BBwagggIxMIICLQIBADCCAiYGCSqGSIb3DQ
eQLmCAgEQgIIB904dv4d93qAZAV8vEUYgMWEf83w9ww5AcShtB_XD4K7z5mrW.

[19] LG Apostolova, RA Dutton, and D Dinov. "Conversion of Mild Cognitive Impairment to Alzheimer Disease Predicted by Hippocampal Atrophy Maps". In: 63.5 (2006), pp. 693–699. DOI: 10.1001/archneur.63.5.693.

[20] Ann D. Cohen and William E. Klunk. "Early detection of Alzheimer's disease using PiB and FDG PET". In: (2014). ISSN: 15250008. DOI: 10.1056/NEJMcibr1012075.Rosiglitazone. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4226742/pdf/nihms604166.pdf.

[21] Aaron D. Benson et al. "Screening for Early Alzheimer's Disease: Is There Still a Role for the Mini-Mental State Examination?" In: *Primary care companion to the Journal of clinical psychiatry* 7.2 (2005), pp. 62–69. ISSN: 1523-5998. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1079697/pdf/i1523-5998-7-2-62.pdf.

[22] D S Knopman et al. *Practice parameter: Diagnosis of dementia (an evidence-based review)*. Tech. rep. 2001, pp. 1–11. URL: https://pdfs.semanticscholar.org/285f/6bf80dfcbe83444d4dcbcd8512a505f7ca0a.pdf.

[23] Dilip V Jeste, Jane S Paulsen, and Ronald C Petersen. "NIH Public Access". In: 19.3 (2013), pp. 205–210. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3076370/pdf/nihms-273128.pdf.

[24] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Vol. 9781107057. 2013, pp. 22–23. ISBN: 9781107298019. DOI: 10.1017/CBO9781107298019. URL: https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf.

[25] Zoubin Ghahramani. *Unsupervised Learning*. Tech. rep. 2004, pp. 3–4. DOI: 10.1007/978-3-540-28650-9{_}5. URL: http://www.inf.ed.ac.uk/teaching/courses/pmr/docs/ul.pdf.

[26] Manfred Borovcnik, Hans-Joachim Bentz, and Ramesh Kapadia. *A Probabilistic Perspective*. 2011, pp. 62–63. ISBN: 9780262018029. DOI: 10.1007/978-94-011-3532-0{_}2. URL: https://doc.lagout.org/science/Artificial%20Intelligence/Machine%20learning/Machine%20Learning_%20A%20Probabilistic%20Perspective%20%5BMurphy%202012-08-24%5D.pdf.

[27] Marina Sokolova and Guy Lapalme. "A systematic analysis of performance measures for classification tasks". In: *Information Processing and Management* 45.4 (2009), pp. 427–437. ISSN: 03064573. DOI: 10.1016/j.ipm.2009.03.002. URL: http://dx.doi.org/10.1016/j.ipm.2009.03.002%20http://atour.iro.umontreal.ca/rali/sites/default/files/publis/SokolovaLapalme-JIPM09.pdf.

[28] Bram van Ginneken. "Fifty years of computer analysis in chest imaging: rule-based, machine learning, deep learning". In: *Radiological Physics and Technology* 10.1 (2017), pp. 23–32. ISSN: 18650341. DOI: 10.1007/s12194-017-0394-5.

[29] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. *Deep learning*. Tech. rep. 7553. 2015, pp. 436–444. DOI: 10.1038/nature14539. URL: https://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf.

[30] Y. Bengio. *Learning Deep Architectures for AI*. Vol. 2. 1. 2009, pp. 1–127. ISBN: 2200000006. DOI: 10.1561/2200000006. URL: https://www.iro.umontreal.ca/~bengioy/papers/ftml_book.pdf.

[31] Erkam Guresen and Gulgun Kayakutlu. "Definition of Artificial Neural Networks with comparison to other networks". In: *Procedia Computer Science* 3 (2011), pp. 426–433. ISSN: 18770509. DOI: 10.1016/j.procs.2010.12.071. URL: http://dx.doi.org/10.1016/j.procs.2010.12.071%20https://core.ac.uk/download/pdf/82123892.pdf.

[32] J. M. Benítez, J. L. Castro, and I. Requena. "Are artificial neural networks black boxes?" In: *IEEE Transactions on Neural Networks* 8.5 (1997), pp. 1156–1164. ISSN: 10459227. DOI: 10.1109/72.623216.

[33] Daniele Ravi et al. "Deep Learning for Health Informatics". In: *American Journal of Hematology* 83.1 (2007), pp. 1–3. ISSN: 03618609. DOI: 10.1002/ajh.21033.

[34] Vladislav Skorpil and Jiri Stastny. "Neural Networks and Back Propagation Algorithm". In: *Electronics. Bulgaria, Sozopol* (2006), pp. 20–22. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.578.4231&rep=rep1&type=pdf.

[35] Jianfang Cao et al. "Big data: A parallel particle swarm optimization-back-propagation neural network algorithm based on MapReduce". In: *PLoS ONE* 11.6 (2016), pp. 1–18. ISSN: 19326203. DOI: 10.1371/journal.pone.0157551. URL: http://dx.doi.org/10.1371/journal.pone.0157551.

[36] *CS231n Winter 2016: Lecture 5: Neural Networks Part 2 - YouTube.*

[37] Anne Solberg. *Lecture : Backpropagation – learning in neural nets (Reading material).* 2017. URL: https://www.uio.no/studier/emner/matnat/ifi/INF5860/v17/undervisningsmateriale/in5860_lecture7_nnet2.pdf.

[38] Daniel Justus et al. "Predicting the Computational Cost of Deep Learning Models". In: (2018). URL: http://arxiv.org/abs/1811.11880.

[39] Charu C. Aggarwal. *Neural networks and deep learning : a textbook.* 2018, p. 453. ISBN: 9783319944630. DOI: 10.1111/j.1464-5491.2005.01480.x.

[40] Silvia Basaia et al. "Automated classification of Alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks". In: *NeuroImage: Clinical* 21.December 2018 (2018), p. 101645. ISSN: 22131582. DOI: 10.1016/j.nicl.2018.101645. URL: https://doi.org/10.1016/j.nicl.2018.101645.

[41] Jyoti Islam and Yanqing Zhang. "Brain MRI analysis for Alzheimer's disease diagnosis using an ensemble system of deep convolutional neural networks". In: *Brain Informatics* 5.2 (2018). ISSN: 21984026. DOI: 10.1186/s40708-018-0080-3. URL: https://doi.org/10.1186/s40708-018-0080-3%20https://braininformatics.springeropen.com/track/pdf/10.1186/s40708-018-0080-3.

[42] Peter J Goadsby, Tobias Kurth, and Alice Pressman. "A review on neuroimaging-based classification studies and associated feature extraction methods for Alzheimer's disease and its prodromal stages". In: 35.14 (2016), pp. 1252–1260. ISSN: 0959-437X. DOI: 10.1177/0333102415576222.Is. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5511557/pdf/nihms874419.pdf.

[43] Kun Hu et al. "Multi-scale features extraction from baseline structure MRI for MCI patient classification and AD early diagnosis". In: *Neurocomputing* 175.PartA (2015), pp. 132–145. ISSN: 18728286. DOI: 10.1016/j.neucom.2015.10.043. URL: http://dx.doi.org/10.1016/j.neucom.2015.10.043.

[44] Alzheimer Society of Canada. "Risk Factors". In: (2019). DOI: 10.1016/B978-0-12-804000-3/00002-8. URL: https://www.ehs.iastate.edu/services/occupational/ergonomics/risk-factors.

[45] Joan Lindsay et al. "Risk factors for Alzheimer's disease: A prospective analysis from the Canadian Study of Health and Aging". In: *American Journal of Epidemiology* 156.5 (2002), pp. 445–453. ISSN: 00029262. DOI: 10.1093/aje/kwf074.

[46] M. Panpalli Ates et al. "Analysis of genetics and risk factors of Alzheimer's Disease". In: *Neuroscience* 325 (2016), pp. 124–131. ISSN: 18737544. DOI: 10.1016/j.neuroscience.2016.03.051. URL: http://dx.doi.org/10.1016/j.neuroscience.2016.03.051.

[47] Noel O Kelly. "Use of machine learning technology in the diagnosis of alzheimer's disease". In: September (2016), p. 91. URL: http://doras.dcu.ie/21356/1/Noel_s_Master_s_thesis__Copy_%281%29.pdf.

[48] Ezedin Wangoria and Henok Wordoffa. "Alzheimer's Disease Stage Prediction using Machine Learning and Multi Agent System". In: September (2012). URL: http://bth.diva-portal.org/smash/get/diva2:829245/FULLTEXT01.pdf.

[49] Tingyan Wang, Robin G. Qiu, and Ming Yu. "Predictive Modeling of the Progression of Alzheimer's Disease with Recurrent Neural Networks". In: *Scientific Reports* 8.1 (2018), pp. 1–12. ISSN: 20452322. DOI: 10.1038/s41598-018-27337-w. URL: http://dx.doi.org/10.1038/s41598-018-27337-w.

[50] Haibo He and EA Garcia. "Learning from imbalanced data". In: *Ieee Transactions on Knowledge and Data Engin* 21.9 (2009), pp. 1263–1284. ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.239. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5128907.

[51] Hinmikaiye J. O et al. "Supervised Machine Learning Algorithms: Classification and Comparison". In: *International Journal of Computer Trends and Technology* 48.3 (2017), pp. 128–138. ISSN: 22312803. DOI: 10.14445/22312803/ijctt-v48p126.

[52] Liyanaarachchi Lekamalage Chamara Kasun et al. "Representational learning with ELMs for big data". In: 4 (2013), pp. 1–4. URL: https://pdfs.semanticscholar.org/8df9/c71f09eb0dabf5adf17bee0f6b36190b52b2.pdf.

[53] Ivan Nunes da Silva et al. *Artificial Neural Networks*. 2017, p. 24. ISBN: 9783319431611. DOI: 10.1007/978-3-319-43162-8.

[54] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: (2014), pp. 1–9. ISSN: 09205691. DOI: 10.1007/s10107-014-0839-0. URL: http://arxiv.org/abs/1409.3215.

[55] S. Ben Driss et al. "A comparison study between MLP and convolutional neural network models for character recognition". In: *Real-Time Image and Video Processing 2017* 10223 (2017), p. 1022306. DOI: 10.1117/12.2262589. URL: https://hal-upec-upem.archives-ouvertes.fr/hal-01525504/document.

[56] Youssef Ghanou et al. "Multilayer Perceptron: Architecture Optimization and Training". In: *International Journal of Interactive Multimedia and Artificial Intelligence* 4.1 (2016), p. 26. DOI: 10.9781/ijimai.2016.415. URL: https://pdfs.semanticscholar.org/7b79/cccc8de41d76a2ca20eacc3d39f7b45bff5f.pdf.

[57] Josh Patterson and Adam Gibson. *Deep Learning: A Practitioner's Approach*. 1st Editio. O'Reilly Media, 2017, Chapter 4: Major Architectures of Deep Networks. ISBN: 978-1491914250.

[58] Ashwin Bhandare et al. "Applications of Convolutional Neural Networks". In: *International Journal of Computer Science and Information Technologies* 7.5 (2016), pp. 2206–2215. URL: http://ijcsit.com/docs/Volume%207/vol7issue5/ijcsit20160705014.pdf.

[59] Saurabh Karsoliya. "Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture". In: *International Journal of Engineering Trends and Technology* 3 (2012). URL: http://www.internationaljournalssrg.org.

[60] Z. Boger and H. Guterman. "Knowledge extraction from artificial neural network models". In: (2002), pp. 3030–3035. DOI: 10.1109/icsmc.1997.633051. URL: https://pdfs.semanticscholar.org/d907/5d3b920cd5bf7eba470c7d841c24a60ef353.pdf.

[61] M.J.A. Berry and G. Linoff. *Data Mining Techniques, NY: John Wiley & Sons*. Tech. rep. 1997.

[62] A. Blum. *Neural Networks in C++: An Object-Oriented Framework for Building Connectionist Systems*. John Wiley & Sons, Inc. New York, 1992. ISBN: 0471552011.

[63] Sandhya Joshi et al. "Classification and Treatment of Different Stages of Alzheimer'S Disease Using Various Machine Learning Methods". In: *International Journal of Bioinformatics Research* 2.1 (2014), pp. 44–52. ISSN: 09753087. DOI: 10.9735/0975-3087.2.1.44-52.

[64] I. S. Isa et al. "Suitable MLP network activation functions for breast cancer and thyroid disease detection". In: *Proceedings - 2nd International Conference on Computational Intelligence, Modelling and Simulation, CIMSim 2010* (2010), p. 41. DOI: 10.1109/CIMSiM.2010.93.

[65] Andrej Karpathy. *Cs231n - Convolutional neural networks for visual recognition*. 2016. URL: http://cs231n.github.io/neural-networks-1/.

[66] Roger Grosse. *Lecture 15 : Exploding and Vanishing Gradients*. 2017. URL: http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf.

[67] Dominik Lewy. *ANN - Activation function comparison*. 2016. URL: http://www.mini.pw.edu.pl/~mandziuk/2017-11-08.pdf.

[68] Farnoush Farhadi and Andrea//Partovi Nia Lodi Vahid. "Learning activation functions in deep neural networks". In: *Département de mathématiques et de génie industriel* (2017), p. 58. URL: https://publications.polymtl.ca/2945/.

[69] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "2011glorot_DeepSparseRectifierNeuralNetwork In: 15 (2011), pp. 315–323. ISSN: 15324435. DOI: 10.1.1.208.6449. URL: http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf.

[70] Abien Fred Agarap. "Deep Learning using Rectified Linear Units (ReLU)". In: 1 (2018), pp. 2–8. URL: http://arxiv.org/abs/1803.08375.

[71] Steve Renals. "Multi-Layer Neural Networks". In: February (2014), p. 3. URL: https://www.inf.ed.ac.uk/teaching/courses/asr/2013-14/asr08a-nnDetails.pdf.

[72] Dabal Pedamonti. "Comparison of non-linear activation functions for deep neural networks on MNIST classification task". In: 3 (2018). URL: http://arxiv.org/abs/1804.02763.

[73] Information Technology and Information Technology. "A REVIEW ON EVALUATION METRICS FOR DATA CLASSIFICATION EVALUATIONS". In: *International Journal of Data Mining & Knowledge Management Process (IJDKP)* 5.2 (2015), pp. 1–11. URL: https://pdfs.semanticscholar.org/6174/3124c2a4b4e550731ac39508c7pdf.

[74] Chul Kee Park and Dong Gyu Kim. "Historical background". In: *Current and Future Management of Brain Metastasis* 25 (2012), pp. 1–12. DOI: 10.1159/000331070. URL: http://leitang.net/papers/ency-cross-validation.pdf.

[75] Lutz Prechelt. "Early stopping - But when?" In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7700 LECTU (2012), pp. 53–67. ISSN: 03029743. DOI: 10.1007/978-3-642-35289-8-5. URL: https://page.mi.fu-berlin.de/prechelt/Biblio/stop_tricks1997.pdf.

[76] Alice Zheng. *Evaluating Machine Learning Algorithms*. 2015, pp. 7–14. ISBN: 9781491932469. URL: https://pindex.com/uploads/post_docs/evaluating-machine-learning-models(PINDEX-DOC-6950).pdf.

[77] Christopher A. Ramezan, Timothy A. Warner, and Aaron E. Maxwell. "Evaluation of Sampling and Cross-Validation Tuning Strategies for Regional-Scale Machine Learning Classification". In: *Remote Sensing* 11.2 (2019), p. 185. DOI: 10.3390/rs11020185.

[78] Ll Pérez-Planells et al. "Análisis de métodos de validación cruzada para la obtención robusta de parámetros biofísicos". In: *Revista de Teledeteccion* 2015.44 (2015), pp. 55–65. ISSN: 19888740. DOI: 10.4995/raet.2015.4153. URL: https://core.ac.uk/download/pdf/71051261.pdf.

[79] Chao Hu, Byeng D. Youn, and Pingfeng Wang. "Ensemble of Data-Driven Prognostic Algorithms With Weight Optimization and K-Fold Cross Validation". In: (2011), p. 3. DOI: 10.1115/detc2010-29182. URL: https://www.phmsociety.org/sites/phmsociety.org/files/phm_submission/2010/phmc_10_025.pdf.

[80] Haider Khalaf Jabbar and Rafiqul Zaman Khan. "Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study)". In: December 2014 (2015), pp. 163–172. DOI: 10.3850/978-981-09-5247-1{_}017. URL: https://www.researchgate.net/profile/Haider_Allamy/publication/295198699_METHODS_TO_AVOID_OVER-FITTING_AND_UNDER-FITTING_IN_SUPERVISED_MACHINE_LEARNING_COMPARATIVE_STUDY/links/56c8253f08aee3cee53a3707.pdf.

[81] Rich Caruana. *Performance Measures for Machine Learning - Advanced Topics in Machine Learning*. 2006. DOI: 10.2307/1269333. URL: http://www.cs.cornell.edu/courses/cs678/2006sp/performance_measures.4up.pdf.

[82] Luigi Cerulo. *Machine Learning Model Evaluation*. 2018. DOI: 10.1007/978-1-4842-4215-5{_}7. URL: http://www.bioinformatics-sannio.org/wordpress/wp-content/uploads/2015/12/ML09-model-evaluation.pdf.

[83] Seong Ho Park, Jin Mo Goo, and Chan Hee Jo. "Receiver operating characteristic (ROC) curve: Practical review for radiologists". In: *Korean Journal of Radiology* 5.1 (2004), pp. 11–18. ISSN: 12296929. DOI: 10.3348/kjr.2004.5.1.11. URL: https://synapse.koreamed.org/Synapse/Data/PDFData/0068KJR/kjr-5-11.pdf.

[84] Nazish Fatima and Saudi Arabia. "Performance Comparison of Most Common High Level Programming Languages". In: *International Journal of Computing Academic Research (IJCAR)* 5.5 (2016), pp. 246–258. URL: http://www.meacse.org/ijcar/archives/109.pdf.

[85] Stephen J. Humer Elvis C. Foster. "A Comparitive Analysis Of The C++, Java, And Python Languages". In: December 2014 (2014). URL: http://www.elcfos.com/papers-in-cs/index/entry/id/7/title/Comparative-Analysis-of-the-C-Java-and-Python-Languages.

[86] *Python programming language*. URL: https://www.python.org/doc/essays/blurb/.

[87] Suryansh Singh and Saikumar Allaka. "R vs Python , why you should learn both ?" In: (). URL: https://www.quadratyx.com/assets/resources/Featured_Insights/R_vs_Python_Why_learn_both.pdf.

[88] Heikki Huttunen. *Machine Learning in Python*. 2015. DOI: 10.1002/9781119183600. URL: http://doi.wiley.com/10.1002/9781119183600.

[89] Christina B. Madsen et al. "Python for Big Data Analytics and the Role of R". In: (2014). URL: http://www.seagate.com/de/de/tech-insights/python-for-analytics-and-the-role-of-r-master-ti/.

[90] Cassius V C Reis. "Comparative Study of Caffe, Neon, Theano, and Torch for Deep Learning". In: *Neurosurgery* 62.2 (2008), pp. 294–310. ISSN: 15760162. DOI: 10.1227/01.NEU.0000297044.82035.57. URL: https://openreview.net/pdf?id=q7kEN7WoXU8LEkD3t7BQ.

[91] Pradeepta Mishra. *PyTorch Recipes*. 2019, p. 30. ISBN: 9781484242575. DOI: 10.1007/978-1-4842-4258-2.

[92] Kanit Wongsuphasawat et al. "Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 1–12. ISSN: 10772626. DOI: 10.1109/TVCG.2017.2744878. URL: https://idl.cs.washington.edu/files/2018-TensorFlowGraph-VAST.pdf.

[93] Javed Ali Khan et al. "Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique". In: *International Journal of Modern Education and Computer Science* 7.11 (2015), pp. 53–59. ISSN: 20750161. DOI: 10.5815/ijmecs.2015.11.06. URL: http://www.mecs-press.org/ijmecs/ijmecs-v7-n11/IJMECS-V7-N11-6.pdf.

[94] Virginia Mato-Abad et al. "Using Artificial Neural Networks for Identifying Patients with Mild Cognitive Impairment Associated with Depression Using Neuropsychological Test Features". In: *Applied Sciences* 8.9 (2018), p. 1629. DOI: 10.3390/app8091629.

[95] Yosra Kazemi. *A Deep Learning Pipeline for Classifying Different Stages of Alzheimer ' s Disease from fMRI Data*. Tech. rep. 2017. URL: https://core.ac.uk/download/pdf/146505572.pdf.

[96] G. J. Awate et al. "Detection-of-Ad-From-Mri-Using-Cnn-With-Tensorflow". In: (2018). URL: https://arxiv.org/pdf/1806.10170.pdf.

[97] Silvia Basaia et al. "Automated classification of Alzheimer's disease and mild cognitive impairment using a single MRI and deep neural networks". In: *NeuroImage: Clinical* 21.October 2018 (2019), p. 101645. ISSN: 22131582. DOI: 10.1016/j.nicl.2018.101645. URL: https://doi.org/10.1016/j.nicl.2018.101645.

[98] Saman Sarraf et al. "DeepAD: Alzheimer's Disease Classification via Deep Convolutional Neural Networks using MRI and fMRI". In: *bioRxiv* 1.1 (2016), pp. 1–32. DOI: 10.1101/070441. URL: https://www.biorxiv.org/content/biorxiv/early/2016/08/30/070441.full.pdf.

[99] Nick Evans and Andrew Tedder. "Holographic model of hgadronization". In: *Physical Review Letters* 100.16 (2008). ISSN: 00319007. DOI: 10.1103/PhysRevLett.100.162003. URL: https://www.researchgate.net/profile/Timothy_Copeland2/publication/328775405_A_Deep_Learning_Model_to_Predict_a_Diagnosis_of_Alzheimer_Disease_by_Using_18_F-FDG_PET_of_the_Brain/links/5be9808c4585150b2bb12b6f/A-Deep-Learning-Model-to-Predict-a-Diagnosis-o.

[100] Ramon Casanova et al. "Alzheimer's Disease Risk Assessment Using Large-Scale Machine Learning Methods". In: *PLoS ONE* 8.11 (2013), e77949. DOI: 10.1371/journal.pone.0077949. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3826736/pdf/pone.0077949.pdf.

[101] Ammarah Farooq et al. "A deep CNN based multi-class classification of Alzheimer's disease using MRI". In: *IST 2017 - IEEE International Conference on Imaging Systems and Techniques, Proceedings* 2018-Janua (2018), pp. 1–6. DOI: 10.1109/IST.2017.8261460.

[102] Victor Miller, Stephen Erlien, and Jeff Piersol. "Identifying dementia in MRI scans using machinelearning". In: (2012), pp. 1–5. URL: http://cs229.stanford.edu/proj2012/ErlienMillerPiersol-IdentifyingDementiaInMRIScansUsingMachineLearning.pdf.

[103] Jyoti Islam and Yanqing Zhang. "Early diagnosis of alzheimer's disease: A neuroimaging study with deep learning architectures". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* 2018-June (2018), pp. 1962–1964. ISSN: 21607516. DOI: 10.1109/CVPRW.2018.00247. URL: http://openaccess.thecvf.com/content_cvpr_2018_workshops/papers/w36/Islam_Early_Diagnosis_of_CVPR_2018_paper.pdf.

[104] S K Aruna and S Chitra. "Machine Learning Approach for Identifying Dementia from MRI Images". In: 9.3 (2015), pp. 881–888. URL: https://waset.org/publications/10004510/machine-learning-approach-for-identifying-dementia-from-mri-images.

[105] Rishi Yadav, Ankit Gautam, and Ravi Bhushan Mishra. "Classification of Alzheimer Using fMRI Data and Brain Network". In: (2018), pp. 109–119. DOI: 10.5121/csit.2018.80609. URL: https://airccj.org/CSCP/vol8/csit88609.pdf.

[106] Naveen Ashish, Priya Bhatt, and Arthur W. Toga. "Global Data Sharing in Alzheimer Disease Research". In: *Alzheimer Disease and Associated Disorders* 30.2 (2016), pp. 160–168. ISSN: 08930341. DOI: 10.1097/WAD.0000000000000121. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4851599/pdf/nihms724516.pdf.

[107] *NACC productivity - last update 3/26/2019*. URL: https://www.alz.washington.edu/cgi-bin/broker93?_service=naccnew9&_program=naccwww.pubrep1.sas&TYPEF=DISPLAYIDS.

[108] S B Kotsiantis, D Kanellopoulos, and P E Pintelas. "Data preprocessing for Supervised Leaning". In: *International Journal of Computer Science* 1.2 (2006), pp. 1–7. ISSN: 1306-4428. DOI: 10.1080/02331931003692557. URL: http://www.google.com/search?client=safari&rls=en&q=Data+Preprocessing+for+Supervised+Leaning&ie=UTF-8&oe=UTF-8%5Cnpapers2://publication/uuid/AA4424CD-8BE0-43AB-838B-8BBBDE502355.

[109] Jiawei Han and Micheline Kamber. *Data Mining : Concepts and Techniques*. 2nd. Elsevier Editiors, 2006, pp. 47–87. ISBN: 9781558609013. DOI: 10.1093/nar/gku1019. URL: http://ebooks.bharathuniv.ac.in/gdlc1/gdlc1/Software%20Engineering/Data%20Mining%20-%20Concepts%20and%20Techniques%20-%202nd%20Edition%20-%20Impressao.pdf.

[110] Ason Brownlee. *How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras*. 2016. URL: https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/.

[111] Foram S Panchal and Mahesh Panchal. "International Journal of Computer Science and Mobile Computing Review on Methods of Selecting Number of Hidden Nodes in Artificial Neural Network". In: *International Journal of Computer Science and Mobile Computing* 3.11 (2014), p. 456. URL: www.ijcsmc.com.

[112] *Sklearn Documentation - GridSearchCV*. URL: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[113] *plot_confusion_matrix @ github.com*. URL: https://github.com/scikit-learn/scikit-learn/blob/master/examples/model_selection/plot_confusion_matrix.py.

[114] Sylvia C Hewitt, Sylvia Curtis Hewitt, and Kenneth S Korach. "Estrogen Receptors : Structure , Mechanisms and Function Estrogen Receptors : Structure , Mechanisms and Function". In: October 2002 (2016), pp. 193–194. DOI: 10.1023/A.

[115] Paras Lakhani and Baskaran Sundaram. “Deep Learning at Chest Radiography: Automated Classification of Pulmonary Tuberculosis by Using Convolutional Neural Networks”. In: *Radiology* 284.2 (2017), pp. 574–582. ISSN: 0033-8419. DOI: 10.1148/radiol.2017162326.

[116] Andrew Ng. “Machine learning Yearning - Technical strategy for AI Engineers in the Era of Deep Learning”. In: (2018), p. 56. URL: https://www.deeplearning.ai/content/uploads/2018/09/Ng-MLY01-12.pdf.

[117] Claudia Perlich. “IBM Research Report - Learning Curves in Machine Learning”. In: (2009). DOI: 10.1007/978-0-387-30164-8. URL: http://link.springer.com/10.1007/978-0-387-30164-8.

[118] *keras-plot-history @ www.kaggle.com*. URL: https://www.kaggle.com/danbrice/keras-plot-history-full-report-and-grid-search.

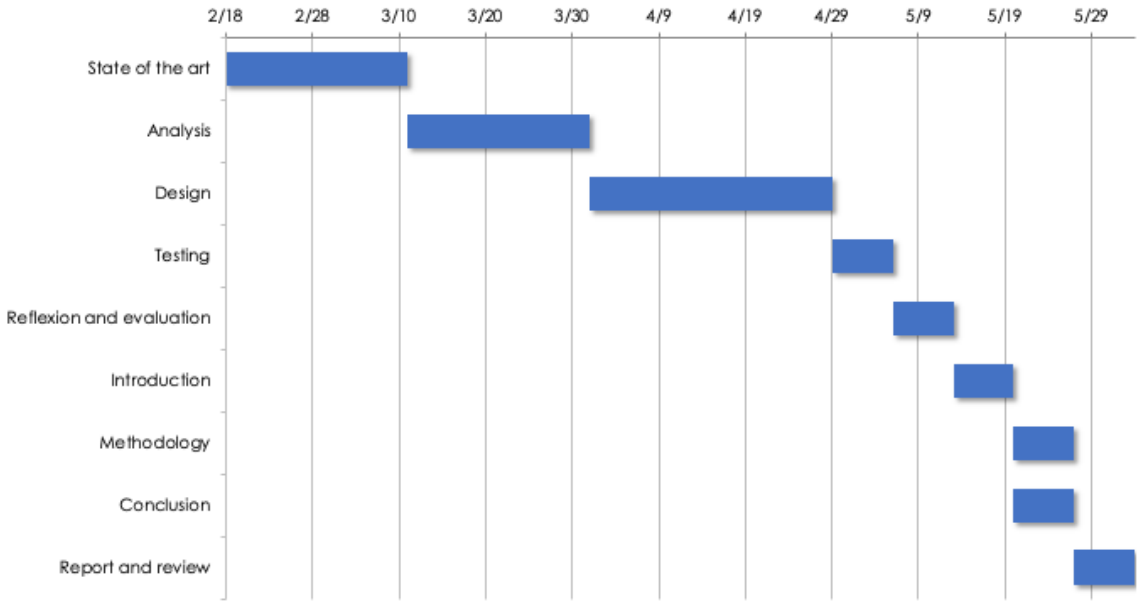
Appendix A

Methodology - Gantt Chart

REPORT PLAN - GANTT CHART

START DATE	END DATE
18-feb	3-jun

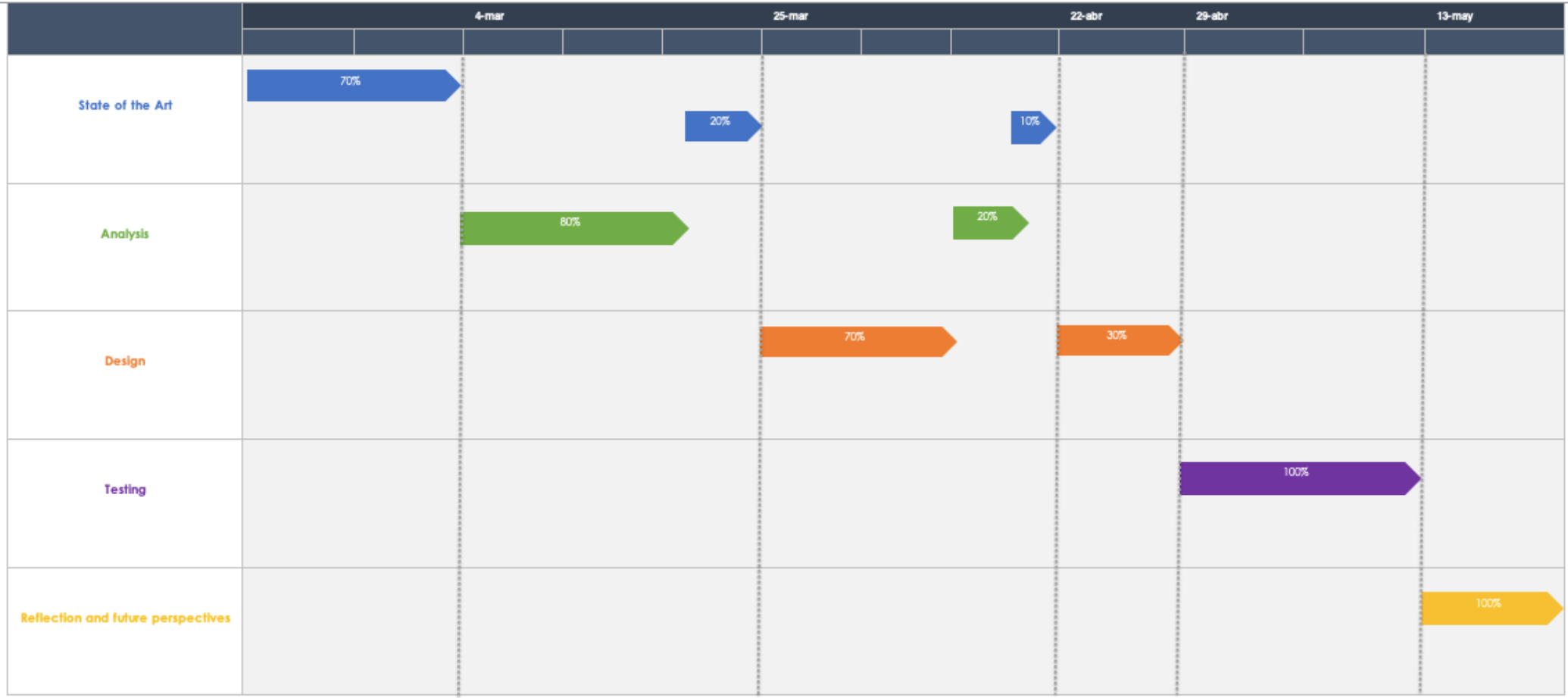
TASKS	START	END	DAYS
State of the art	2/18	3/11	21
Analysis	3/11	4/1	21
Design	4/1	4/29	28
Testing	4/29	5/6	7
Reflexion and evaluation	5/6	5/13	7
Introduction	5/13	5/20	7
Methodology	5/20	5/27	7
Conclusion	5/20	5/27	7
Report and review	5/27	6/3	7



Appendix B

Methodology - Agile Chart

AGILE CHART


4-mar
25-mar
22-abr
29-abr
13-may
State of the Art
70%
20%
10%
Analysis
80%
20%
Design
70%
30%
Testing
100%
Reflection and future perspectives
100%

Appendix C

DSM-IV Diagnosis Criteria

DSM-IV Criteria for the Diagnosis of Alzheimer's Disease

	Yes	No
A. The development of multiple cognitive deficits manifested by both:		
1. Memory impairment (impaired ability to learn new information or to recall previously learned information).	☐	☐
2. One (or more) of the following cognitive disturbances:		
a. Aphasia (language disturbance).	☐	☐
b. Apraxia (impaired ability to carry out motor activities despite intact motor function.	☐	☐
c. Agnosia (failure to recognize or identify objects despite intact sensory function).	☐	☐
d. Disturbance in executive functioning (i.e., planning, organizing, sequencing, abstracting).	☐	☐
B. The cognitive deficits in Criteria A1 and A2 each cause significant impairment in social or occupational functioning and represent a significant decline from a previous level of functioning.	☐	☐
C. The course is characterized by gradual onset and continuing cognitive decline.	☐	☐
D. The cognitive deficits in Criteria A1 and A2 are not due to any of the following:		
1. Other central nervous systems, conditions that cause progressive deficits in memory and cognition (e.g., cerebrovascular disease, Parkinson's disease, Huntington's disease, subdural hematoma, normal-pressure hydrocephalus, brain tumor).	☐	☐
2. Systemic conditions that are known to cause dementia (e.g., hypothyroidism, vitamin B_{12} or folic acid deficiency, neurosyphilis, HIV infection).	☐	☐
3. Substance-induced conditions.	☐	☐
E. The deficits do not occur exclusively during the course of a delirium.	☐	☐
F. The disturbance is not better accounted for by another disorder (e.g., major depressive disorder, schizophrenia).	☐	☐

Reference: Diagnostic and Statistical Manual of Mental Disorders, Fourth Edition, Text Revision. Washington, D.C., American Psychiatric Association, 2000.

Appendix D

List of selected variables for the model

Variable	Meaning
NACCVNUM	Unique patient id
NACCAGE	Age of the patient
SEX	Female or male
HISPANIC	Hispanic race or not
EDUC	Years of education
TOBAC1OO	Smoker patient or not
CVHATT	Hearth attack
CVAFIB	Atrial fibrillation
CVANGIO	Angioplasty/endarterectomy/stent
CVBYPASS	Cardiac bypass procedure
CVPACE	Pacemaker
CVCHF	Congestive heart failure
CVOTHR	Other cardiovascular disease
CBSTROKE	Stroke
PD	Parkinson's disease
SEIZURES	Seizures
NACCTBI	History of traumatic brain injury (TBI)
DIABETES	Diabetes
HYPERTEN	Hypertension
B12DEF	Vitamin B12 deficiency
THYROID	Thyroid disease
ABUSOTHR	Abused substances
NACCMMSE	MMSE score
DOWNS	Down syndrome
DEP	Depression
NACC_ALZ	Final Alzheimer diagnosis

Appendix E

Python code of the solution

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import itertools
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import type_of_target
from sklearn.metrics import roc_curve, auc, roc_auc_score


# fix random seed for reproducibility
seed = 7
np.random.seed(seed)



# # MODULE 1: DATA COLLECTION AND PRE-PROCESSING

# Reading the dataset in a dataframe using Pandas
```

```
df = pd.read_csv("dataset.csv", sep=';')

# A previous selection of which fields include has been made. This
   selection has been done with the help of an specialist in the area.
   Also, fields with a lot of null values has been discarted.
df.head(10)
df.describe()


# - 31245 patients (NACCVNUM = unique id)
# - No missing values in any of the fields
# - NACCAGE: Min age of 18, max of 109 and mean of 72
# - Sex: 1=Male 2=Female
# - HISPANIC: 0=No 1=Yes 9=Unknown
# - EDUC: years of education, between 0 and 36 99=Unknown
# - TOBAC100: smoked more than 100 cigarretes in life 0=No 1=Yes 9=Unknown
# - CVHATT: heart attack 0=Absent 1=Recent/Active 2=Remote/Inactive 9=
   Unknown
# - CVAFIB: atrial fibrillation 0=Absent 1=Recent/Active 2=Remote/Inactive
    9=Unknown
# - CVANGIO: angioplasty/endarterectomy/stent 0=Absent 1=Recent/Active 2 =
   Remote/Inactive 9=Unknown
# - CVBYPASS: Cardiac bypass procedure 0=Absent 1=Recent/Active 2=Remote/
   Inactive 9=Unknown
# - CVPACE: Pacemaker 0=Absent 1=Recent/Active 2=Remote/Inactive 9=Unknown
# - CVCHF: Congestive heart failure 0=Absent 1=Recent/Active 2=Remote/
   Inactive 9=Unknown
# - CVOTHR: Other cardiovascular disease 0=Absent 1=Recent/Active 2=Remote
   /Inactive 9=Unknown
# - CBSTROKE: stroke 0=Absent 1=Recent/Active 2=Remote/Inactive 9=Unknown
# - PD: Parkinsons disease (PD) 0=Absent 1=Recent/Active 9=Unknown
# - SEIZURES: seizures 0=Absent 1=Recent/Active 2=Remote/Inactive 9=
   Unknown
# - NACCTBI: history of traumatic brain injury (TBI) 0=No 1=Yes 9=Unknown
# - DIABETES: diabetes 0=Absent 1=Recent/Active 2=Remote/Inactive 9=
   Unknown
```

```
# - HYPERTEN: hypertension 0=Absent 1=Recent/Active 2=Remote/Inactive 9=
    Unknown
# - B12DEF: vitamin B12 deficiency 0=Absent 1=Recent/Active 2=Remote/
    Inactive 9=Unknown
# - THYROID: thyroid disease 0=Absent 1=Recent/Active 2=Remote/Inactive 9=
    Unknown
# - ABUSOTHR: abused substances 0=Absent 1=Recent/Active 2=Remote/Inactive
     9=Unknown
# - NACCMMSE: MMSE score between 030
# - DOWNS: down syndrome 0 = No 1 = Yes
# - DEP: Depression 0 = No (assumed assessed and found not present) 1 =
    Yes
# - NACC_ALZ: Final alzheimer diagnosis 1=Yes 0= No

df.isnull().sum()

# bar drawing function
def bar_chart(feature):
    AD = df[df['NACC_ALZ']==1][feature].value_counts()
    NonAD = df[df['NACC_ALZ']==0][feature].value_counts()
    df_bar = pd.DataFrame([AD,NonAD])
    df_bar.index = ['AD','NonAD']
    df_bar.plot(kind='bar',stacked=True, figsize=(8,5))

# Gender
bar_chart('SEX')
plt.xlabel('NACC_ALZ')
plt.ylabel('Number␣of␣patients')
plt.legend()
plt.title('Gender␣and␣AD')


# The above graph indicates that women are more likely with dementia than
    men.

# MMSE : Mini Mental State Examination
```

```
# mmse: min 0 , max 30
facet= sns.FacetGrid(df,hue="NACC_ALZ", aspect=3)
facet.map(sns.kdeplot,'NACCMMSE',shade= True)
facet.set(xlim=(0, df['NACCMMSE'].max()))
facet.add_legend()
plt.xlim(0.30)


# The chart shows Non-AD group got much more higher MMSE scores than AD
    group.

# AGE
facet= sns.FacetGrid(df,hue="NACC_ALZ", aspect=3)
facet.map(sns.kdeplot,'NACCAGE',shade= True)
facet.set(xlim=(0, df['NACCAGE'].max()))
facet.add_legend()
plt.xlim(0,110)


# There is a higher concentration of 70-85 years old in the AD patient
    group than those in the non-AD patients.

df0.describe()
df1.describe()

df0 = df[df['NACC_ALZ'] == 0]
df1 = df[df['NACC_ALZ'] == 1]

df0 = df0.sample(8500)
df1 = df1.sample(8500)

df = pd.concat([df0, df1])

#Shuffle
df = df.sample(frac=1).reset_index(drop=True)
df.describe()
```

```
# # MODULE 2: NEURAL NETWORK

dataset = df.values
X = dataset[:,1:26]
Y = dataset[:,26]

Y=Y.astype('int')

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
   random_state=1)

# create model
model = Sequential()
model.add(Dense(25, input_dim=25, activation='tanh'))
model.add(Dense(25, activation='tanh'))
model.add(Dense(1, activation='sigmoid'))

# Compile model
model.compile(loss='mean_absolute_error', optimizer='Adamax', metrics=['
   accuracy'])
model.summary()

# Fit the model
history = model.fit(X_train, Y_train, validation_split=0.1, epochs=1600,
   batch_size=5000, verbose=2)


# # MODULE 3: TESTING AND VISUALIZATION OF RESULTS

def plot_history(history):

    loss_list = [s for s in history.history.keys() if 'loss' in s and 'val'
       not in s]
```

```
    val_loss_list = [s for s in history.history.keys() if 'loss' in s and '
        val' in s]

    if len(loss_list) == 0:
        print('Loss is missing in history')
        return

    ## As loss always exists
    epochs = range(1,len(history.history[loss_list[0]]) + 1)

    ## Loss
    plt.figure(1)
    for l in loss_list:
        plt.plot(epochs, history.history[l], 'b', label='Training loss (' +
            str(str(format(history.history[l][-1],'.5f'))+')'))
    for l in val_loss_list:
        plt.plot(epochs, history.history[l], 'g', label='Validation loss ('
            + str(str(format(history.history[l][-1],'.5f'))+')'))

    plt.title('Loss')
    plt.ylim(0, 1)
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()


def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.YlOrRd):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
```

```
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                horizontalalignment="center",
                color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()


plot_history(history)


# evaluate the model
scores = model.evaluate(X_test, Y_test)

print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))


for i,value in enumerate(Y_test):
```

```
    print(Y_test[i])


y_pred = model.predict(X_test)
y_pred2 = np.empty([len(y_pred)], dtype=int)

for i,value in enumerate(y_pred):

    #Threshold
    if value>0.5:
        y_pred2[i] = 1
    else:
        y_pred2[i] = 0

for i,value in enumerate(y_pred2):
    print(y_pred2[i])


print(type_of_target(Y_test))
print(type_of_target(y_pred2))

# Output of predictions

for i,value in enumerate(y_pred):
    print(y_pred[i])

cnf_matrix = confusion_matrix(Y_test, y_pred2)
plot_confusion_matrix(cnf_matrix, classes=[0,1], normalize=False,
                      title='Confusion␣matrix')

false_positive_rate, true_positive_rate, thresholds = roc_curve(Y_test,
   y_pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

plt.xlabel('False␣Positive␣Rate')
plt.ylabel('True␣Positive␣Rate')
```

```
plt.title('Receiver␣operating␣characteristic␣example')
plt.legend(loc="lower␣right")

plt.title('Receiver␣Operating␣Characteristic')

plt.plot(false_positive_rate, true_positive_rate, 'darkorange',
label='Deep␣neural␣network␣␣ROC␣(AUC␣=␣%0.2f)' % roc_auc)

plt.plot([0,1],[0,1],color='slategray',linestyle='--', label = 'Random␣guess
    ')
plt.legend(loc='lower␣right')
plt.show()
```

Appendix F

Script 1 - Tuning the number of neurons

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier

# fix random seed for reproducibility
seed = 7
np.random.seed(seed)
df = pd.read_csv("dataset.csv", sep=';') # Reading the dataset in a
    dataframe using Pandas

df0 = df[df['NACC_ALZ'] == 0]
df1 = df[df['NACC_ALZ'] == 1]

df0 = df0.sample(8500)
df1 = df1.sample(8500)
```

```
df = pd.concat([df0, df1])

#Shuffle
df = df.sample(frac=1).reset_index(drop=True)

df.head(100)
df.shape

dataset = df.values
X = dataset[:,1:26]
Y = dataset[:,26]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
   random_state=1)

def create_model(neurons=1):
    # create model
    model = Sequential()
    model.add(Dense(neurons, input_dim=25, activation='relu'))
    model.add(Dense(neurons, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    # compile model
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['
       accuracy'])
    return model

model = KerasClassifier(build_fn=create_model, epochs=300, batch_size=800)

# define the grid search parameters
neurons = [10, 15, 20, 25, 30, 35, 40, 45, 50]
param_grid = dict(neurons=neurons)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
grid_result = grid.fit(X, Y, verbose=0)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.
   best_params_))
```

```
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f␣(%f)␣with:␣%r" % (mean, stdev, param))
```

Appendix G

Script 2 - Tuning the activation function

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier

# fix random seed for reproducibility
seed = 7
np.random.seed(seed)
df = pd.read_csv("dataset.csv", sep=';') # Reading the dataset in a
    dataframe using Pandas

df0 = df[df['NACC_ALZ'] == 0]
df1 = df[df['NACC_ALZ'] == 1]

df0 = df0.sample(8500)
df1 = df1.sample(8500)
```

```
df = pd.concat([df0, df1])

#Shuffle
df = df.sample(frac=1).reset_index(drop=True)

df.head(100)
df.shape

dataset = df.values
X = dataset[:,1:26]
Y = dataset[:,26]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
   random_state=1)

def create_model(activation='relu'):
    # create model
    model = Sequential()
    model.add(Dense(25, input_dim=25, activation=activation))
    model.add(Dense(25, activation=activation))
    model.add(Dense(1, activation='sigmoid'))
    # compile model
    model.compile(loss='binary_crossentropy', optimizer='Adamax', metrics=['
       accuracy'])
    return model

model = KerasClassifier(build_fn=create_model, epochs=300, batch_size=800)

# define the grid search parameters
activation = ['softmax', 'relu', 'tanh', 'sigmoid']
param_grid = dict(activation=activation)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
grid_result = grid.fit(X, Y, verbose=0)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.
   best_params_))
```

```
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f␣(%f)␣with:␣%r" % (mean, stdev, param))
```

Appendix H

Script 3 - Tuning the optimizer

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier

# fix random seed for reproducibility
seed = 7
np.random.seed(seed)
df = pd.read_csv("dataset.csv", sep=';') # Reading the dataset in a
    dataframe using Pandas

df0 = df[df['NACC_ALZ'] == 0]
df1 = df[df['NACC_ALZ'] == 1]

df0 = df0.sample(8500)
df1 = df1.sample(8500)

df = pd.concat([df0, df1])
```

```
#Shuffle
df = df.sample(frac=1).reset_index(drop=True)

df.head(100)
df.shape

dataset = df.values
X = dataset[:,1:26]
Y = dataset[:,26]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
   random_state=1)

def create_model(optimizer='adam'):
    # create model
    model = Sequential()
    model.add(Dense(25, input_dim=25, activation='tanh'))
    model.add(Dense(25, activation='tanh'))
    model.add(Dense(1, activation='sigmoid'))
    # compile model
    model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=[
       'accuracy'])
    return model


model = KerasClassifier(build_fn=create_model, epochs=300, batch_size=800)

# define the grid search parameters
optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', '
   Nadam']
param_grid = dict(optimizer=optimizer)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
grid_result = grid.fit(X, Y, verbose=0)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.
   best_params_))
```

```
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f␣(%f)␣with:␣%r" % (mean, stdev, param))
```

Appendix I

Script 4 - Tuning batch size and number of epochs

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier

# fix random seed for reproducibility
seed = 7
np.random.seed(seed)
df = pd.read_csv("dataset.csv", sep=';') # Reading the dataset in a
    dataframe using Pandas

df0 = df[df['NACC_ALZ'] == 0]
df1 = df[df['NACC_ALZ'] == 1]

df0 = df0.sample(8500)
df1 = df1.sample(8500)
```

```
df = pd.concat([df0, df1])

#Shuffle
df = df.sample(frac=1).reset_index(drop=True)

df.head(100)
df.shape

dataset = df.values
X = dataset[:,1:26]
Y = dataset[:,26]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
   random_state=1)

def create_model():
    # create model
    model = Sequential()
    model.add(Dense(25, input_dim=25, activation='tanh'))
    model.add(Dense(25, activation='tanh'))
    model.add(Dense(1, activation='sigmoid'))
    # compile model
    model.compile(loss='binary_crossentropy', optimizer='Adamax', metrics=['
       accuracy'])
    return model

model = KerasClassifier(build_fn=create_model)

# define the grid search parameters
batch_size = [800, 2000, 5000, 10000, 13000, 15000]
epochs = [600, 800, 1000, 1200, 1400, 1600]
param_grid = dict(batch_size=batch_size, epochs=epochs)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
grid_result = grid.fit(X, Y, verbose=0)
# summarize results
```

```
print("Best: %f using %s" % (grid_result.best_score_, grid_result.
    best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```