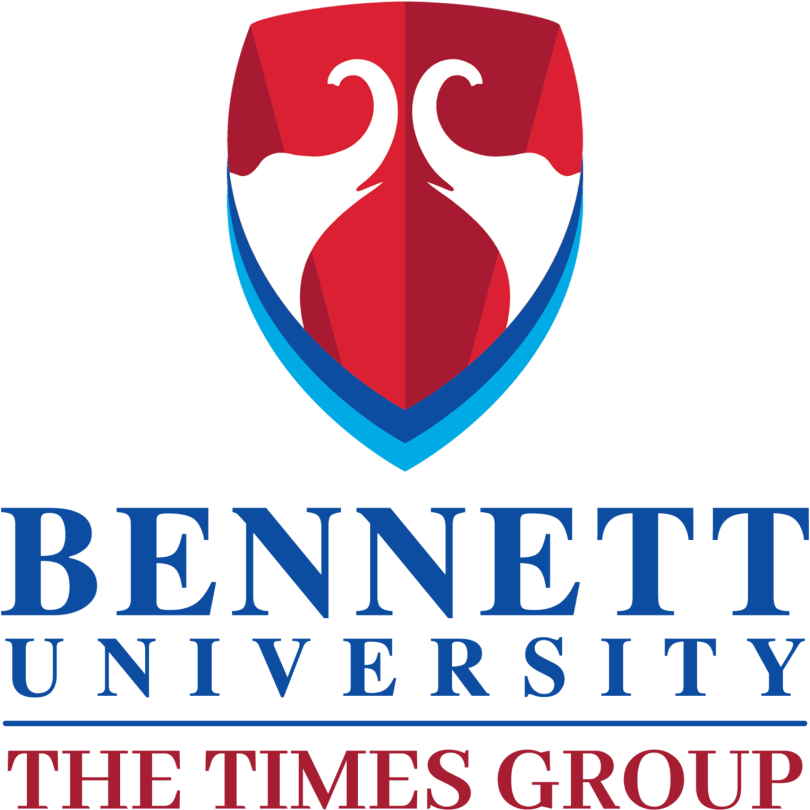
# School of Computer Science Engineering and Technology

Digital marketing & Trend Analysis

Report File



# Coordinator: Student:

Dr. Arun Kumar Yash Agrawal E23MCAG0028

Swastik Singh E23MCAG0128

Course – MCA(B5)

**Contents**

[List of acronyms and abbreviations](#_bookmark0) 1

1. [Introduction](#_bookmark38) 2

1.1 Problem formulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

[1.2 Limitations](#_bookmark40) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

[1.3 Methodology](#_bookmark41) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

[1.3.1 Gantt Chart of the report](#_bookmark42) . . . . . . . . . . . . . . . . . . . . . . . . 5

[1.3.2 Agile realization of the project](#_bookmark44) . . . . . . . . . . . . . . . . . . . . . . 6

[1.4 Structure of the report](#_bookmark46) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

1. [State of the Art](#_bookmark47) 9

[2.1 Alzheimer disease](#_bookmark48) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

* + 1. [Causes](#_bookmark49) 10
    2. [Diagnosis](#_bookmark50) 11
    3. [Summary](#_bookmark51) 14
  1. [Machine Learning](#_bookmark52) 14
     1. [Classification in Deep Learning](#_bookmark53) 16

1. [Analysis](#_bookmark55) 19
   1. [Identification of the required data](#_bookmark56) 19
   2. [Defining the classifier](#_bookmark59) 22
   3. [Model hyperparameters](#_bookmark60) 23
      1. [Number of layers and neurons](#_bookmark61) 24
      2. [Activation functions](#_bookmark62) 25
   4. [Evaluation metrics](#_bookmark64) 26
   5. [Programming languages and libraries](#_bookmark65) 29
   6. [Requirements](#_bookmark67) 31
      1. [Functional requirements](#_bookmark68) 31
      2. [Non-Functional requirements](#_bookmark70) 32
2. [Conceptual Design](#_bookmark73) 35
   1. [System overview](#_bookmark74) 35
   2. [Required data input](#_bookmark76) 37
      1. [Description of the](#_bookmark77) [NA](#_bookmark25)[CC Database](#_bookmark77) 38
      2. [Analysis of the required characteristics](#_bookmark78) 39
   3. [Data collection and pre-processing module](#_bookmark79) 41
   4. [Neural Network](#_bookmark80) 42
   5. [Testing and visualization of results](#_bookmark82) 43
   6. [Summary](#_bookmark83) 44
3. [Implementation](#_bookmark85) 46
   1. [Data collection and pre-processing](#_bookmark86) 46
   2. [Neural Network](#_bookmark91) 49
4. [Testing](#_bookmark96) 53
   1. [Accuracy and confusion matrix](#_bookmark97) 53
   2. [Receiv](#_bookmark31)[er operating characteristic (](#_bookmark101)[R](#_bookmark31)[OC)](#_bookmark101) 55
   3. [Analysis of learning curves](#_bookmark103) 56
5. [Conclusions](#_bookmark106) 60
   1. [Future perspectives](#_bookmark107) 62

[Bibliography](#_bookmark108) 63

[A Methodology - Gantt Chart](#_bookmark218) 77

1. [Methodology - Agile Chart](#_bookmark219) 79
2. [DSM-IV Diagnosis Criteria](#_bookmark220) 81
3. [List of selected variables for the model](#_bookmark221) 83
4. [Python code of the solution](#_bookmark222) 85
5. [Script 1 - Tuning the number of neurons](#_bookmark223) 94
6. [Script 2 - Tuning the activation function](#_bookmark224) 97
7. [Script 3 - Tuning the optimizer](#_bookmark225) 100
8. [Script 4 - Tuning batch size and number of epochs](#_bookmark226) 103

# List of acronyms and abbreviations

**AD** Alzheimer’s Disease

**ADCs** Alzheimer’s Disease Centers

**ADGC** Alzheimer’s Disease Genetics Con- sortium

**ADNI** The Alzheimer’s Disease Neuroimag- ing 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 Perceptron's **MMSE** Mini-Mental State Exam **MRI** Magnetic Resonance Imaging

operating characteristic **TNR** True Negative Rate

**TPR** True Positive Rate

**UDS** The Uniform Data Set

**Chapter 1 Introduction**

[Alzheimer’s Disease](#_bookmark1) ([AD](#_bookmark1)) 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](#_bookmark109)]. 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](#_bookmark110)]. 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 neu- ral 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.

## 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?

## 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.

## 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.

**Data Collection and Preprocessing**

***Dataset Description:*** The dataset consists of MRI images categorized into four classes: highly demented, mild demented, non-demented, and moderate demented.

***Data Augmentation:*** To increase the size of the dataset and improve model generalization, data ***augmentation techniques such as rotation, flipping, and scaling were applied.***

***Normalization:*** Pixel values in the images were normalized to a range of [0, 1] to standardize the data.

**Model Development**

***Baseline Models:*** Initially, simple models like CNN 2D, Random Forest, SVM, and MLP were trained to establish baseline performance.

***Hyperparameter Tuning:*** Hyperparameters of each model were tuned using techniques like grid search to optimize performance.

***Model Evaluation:***Models were evaluated based on accuracy, precision, recall, and F1-score to assess their performance.

**Advanced Techniques**

***Combined Model:*** A combination of Random Forest and MLP was explored to leverage the strengths of both models.

***Stacking:*** Stacking was applied using Random Forest, MLP, and SVM as base models. The outputs of these models were used as inputs for a final Random Forest model.

***Evaluation Metrics:*** The final model was evaluated using various metrics, and cross-validation was performed to ensure its robustness.

**Model Selection**

***Accuracy Criteria:*** The final model was selected based on its accuracy and performance compared to the baseline and intermediate models.

***Deployment Considerations:*** The selected model's scalability and efficiency were considered for potential deployment in a real-world scenario.

**Ethical Considerations**

***Data Privacy:*** Ensuring patient data privacy and confidentiality was a top priority throughout the project.

***Bias and Fairness:*** Steps were taken to mitigate bias in the dataset and ensure fairness in model predictions across different demographic groups.

**Future Work**

***Enhanced Model Architectures:*** Exploring more complex CNN architectures or transfer learning approaches could further improve model performance.

***Clinical Validation:*** Collaborating with healthcare professionals for clinical validation of the model's predictions could enhance its credibility and real-world utility.

***Longitudinal Studies:*** Conducting longitudinal studies to track disease progression over time could provide deeper insights into the effectiveness of the model.

**Conclusion**

***Achievements:*** The project successfully developed a model for classifying dementia stages from MRI images with a high accuracy of 92%.

***Future Prospects:*** Further research and collaboration with healthcare institutions could lead to the deployment of the model in clinical settings, potentially improving dementia diagnosis and patient care.

### 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.

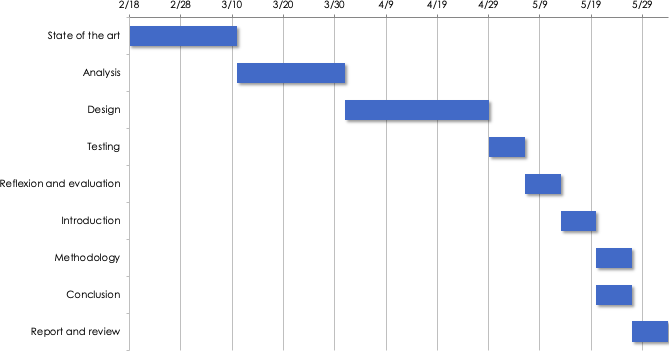


Figure 1.1: Gantt Chart representing the realization of the project

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](#_bookmark43), 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](#_bookmark218).

### Agile realization of the project

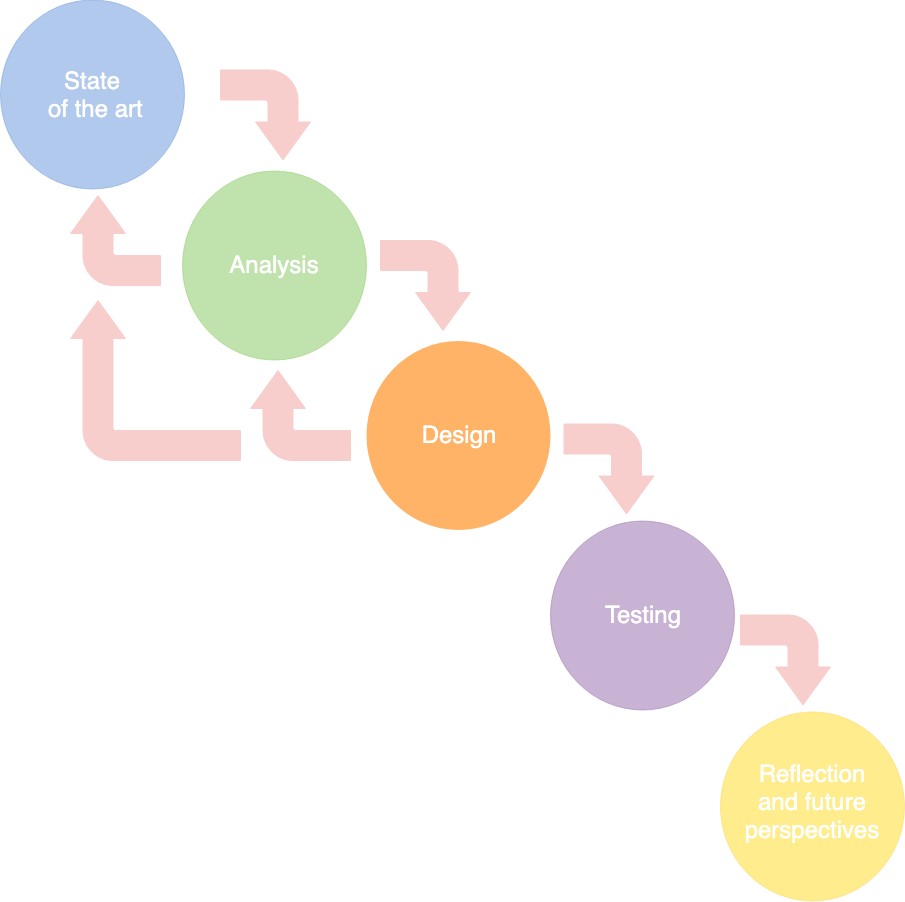


Figure 1.2: Agile process of the project

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](#_bookmark38).

As it can be seen on Figure [1.2](#_bookmark45), this Agile process begins with a *State of the Art* phase. During this phase, the required knowledge regarding the [AD](#_bookmark1) 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](#_bookmark219). It is important to under- stand the reasons behind the choice of this working method. Although a static goal-oriented plan, such as that presented in Section [1.3.1](#_bookmark42), 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 return- ing 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.

## 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](#_bookmark38) and represents the foundations needed in order to start an analysis on Chapter [3](#_bookmark55).

## 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](#_bookmark1) ([AD](#_bookmark1)) is a type of progressive dementia that has memory deficit as one of its earliest and most pronounced symptoms [[4](#_bookmark112)].

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](#_bookmark1) 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](#_bookmark113)]. 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](#_bookmark1) is the gradual loss of the ability to remember new information, the following ones are also common of the disease [[6](#_bookmark114)]: 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.

### 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](#_bookmark1) [[7](#_bookmark115)]. 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](#_bookmark1), 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](#_bookmark1) 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](#_bookmark1) [[8](#_bookmark116)]. The reason behind this risk factor is the additional copy of the [Amyloid Precursor Protein](#_bookmark6) ([APP](#_bookmark6)) 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](#_bookmark1) however, tau protein becomes defected and detaches from the microtubules sticking to other tau molecules [[9](#_bookmark117)]. 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](#_bookmark1), 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 concen- tration 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](#_bookmark1) [[10](#_bookmark118)].

As a result, even if the cause of [AD](#_bookmark1) remains controversial and is incompletely understood, the presence of senile plaques and neurofibrillary tangles constitute the major neuropatho- logical characteristics of [AD](#_bookmark1) resulting in an important area to continue the research [[11](#_bookmark119)].

### 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](#_bookmark28) ([NINCDS](#_bookmark28)) and [Alzheimer’s Disease and Related Disorders Association](#_bookmark5) ([ADRDA](#_bookmark5)) workgroup in 1984 [[12](#_bookmark120)]. 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](#_bookmark1) 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](#_bookmark121)]. Nowadays, the diagnosis of the [AD](#_bookmark1) 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 un- derlies the syndrome. As a result, the actual criteria combines clinical and neuropathological patterns assigning three different level of diagnosis, "possible [AD](#_bookmark1)", "probable [AD](#_bookmark1)" or "definite [AD](#_bookmark1)" [[12](#_bookmark120)].

**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](#_bookmark122)]. 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](#_bookmark1) 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](#_bookmark1) is the [Cerebrospinal Fluid](#_bookmark10) ([CSF](#_bookmark10)) examination. Due to the free transport of proteins between the brain and the the body through the [CSF](#_bookmark10), beta-amyloid and tau levels are reflected in the [CSF](#_bookmark10) analysis which can be significant even at an early stage of disease [[15](#_bookmark123)]. However, the association between [CSF](#_bookmark10) 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](#_bookmark10) analysis. Whereas the presence of neurofibrillary tangles will be represented by high tau protein levels on the [CSF](#_bookmark10) results [[16](#_bookmark124)]. But not all scientific studies affirm these hypotheses, making such a biomarker an inconsistent test with uncertain value [[17](#_bookmark125)][[18](#_bookmark126)].

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](#_bookmark22) ([MRI](#_bookmark22)) and

[Computed Tomography](#_bookmark11) ([CT](#_bookmark11)). 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](#_bookmark1) [[19](#_bookmark127)]. 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](#_bookmark32) ([PET](#_bookmark32)) and [Functional MRI](#_bookmark14) ([fMRI](#_bookmark14)). 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](#_bookmark32) analysis indicate that Alzheimer’s is often associated with reduced use of glucose in brain areas important in memory, learning and problem-solving [[20](#_bookmark128)]. 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](#_bookmark10) analysis methods do exist, should not be used as the only procedure for the diagnosis of AD [[12](#_bookmark120)]. 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](#_bookmark1). 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](#_bookmark23) ([MMSE](#_bookmark23)) is one of the most commonly used screening tests to evaluate cognitive functioning [[21](#_bookmark129)]. The [MMSE](#_bookmark23) 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](#_bookmark130)].

As a result of this cognitive examination, the physician will be able to apply the stan- dardized DSM-IV Criteria for the Diagnosis of Alzheimer’s disease, resulting on a clinical diagnosis of the dementia [[23](#_bookmark131)]. The complete version of the DSM-IV Criteria for the Diag- nosis of Alzheimer’s can be found in Appendix [C](#_bookmark220). 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.

### 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](#_bookmark1) 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](#_bookmark10), [MRI](#_bookmark22), [PET](#_bookmark32) .. etc).
        + Clinical diagnosis: based on the medical history of the patient, an analysis of his lifestyle or habits and cognitive/metal evaluation.

## Machine Learning

Arthur Samuel in 1959 defined [Machine Learning](#_bookmark21) ([ML](#_bookmark21)) 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, infor- mation theory or functional analysis.

Depending on the type of learning task to perform, [ML](#_bookmark21) can be subdivided into two different fields, supervised or unsupervised learning. Supervised learning requires a priori knowledge of what the result should be [[24](#_bookmark132)]. Pairs of data inputs and data outputs have to be presented to the [ML](#_bookmark21) 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](#_bookmark21) 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](#_bookmark133)].

Approaching the different [ML](#_bookmark21) 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, de- pending 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 *ε {*1*, ..., C}* , C being the number of classes to whom y may belong [[26](#_bookmark134)]. 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](#_bookmark135)]. These four measures can be represented on the confusion matrix.

### Classification in Deep Learning

Even [ML](#_bookmark21) technology powers many different supervised issues in the society, his ability to process natural data in their raw form will be limited [[28](#_bookmark136)]. 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](#_bookmark137)]. The big drawback of [ML](#_bookmark21) 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](#_bookmark12) ([DL](#_bookmark12)) 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](#_bookmark15) ([DNN](#_bookmark15)) [[28](#_bookmark136)].

[DNN](#_bookmark15) is a mathematical representation of the human neural architecture [[30](#_bookmark138)]. 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](#_bookmark139)]. 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](#_bookmark140)].

**The learning process** The [DNN](#_bookmark15) 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](#_bookmark141)].

Trying to map it into mathematical expressions, the model uses a training dataset constituted by pairs input-target, *D* = *{xi, yi}* where *xi* is a vector of features an *yi* 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](#_bookmark142)][[35](#_bookmark143)].

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](#_bookmark144)]. 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](#_bookmark145)].

As a result, [DL](#_bookmark12) can be understood as a type or evolution of different [ML](#_bookmark21) 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](#_bookmark146)]. In addition, [DNN](#_bookmark15) requires significant volumes of data to reach a decent level of accuracy, being not possible to apply [DL](#_bookmark12) architectures to small datasets [[39](#_bookmark147)]. 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](#_bookmark21) 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](#_bookmark1) 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](#_bookmark47) will be taken into consideration, adding the critical vision that allows us to combine the field of [AD](#_bookmark1) 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.

## 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](#_bookmark48), 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](#_bookmark10) examination. These last tests require a complex and professional exploration, such as the analysis of the images scanned.

## 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](#_bookmark157)]. As it has been mentioned in Section [2.2](#_bookmark52), 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](#_bookmark15) solution rather than traditional [ML](#_bookmark21) algorithms.

In addition, Deep Learning machines usually work better than traditional [ML](#_bookmark21) tools be- cause 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](#_bookmark15) makes even more sense rather than the implementation of traditional [ML](#_bookmark21) 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](#_bookmark158)]. Due to the lack of this medical knowledge, again the selection of a [DNN](#_bookmark15) rather traditional [ML](#_bookmark21) methods become even more clear than before.

But once that the [DL](#_bookmark12) path has been taken, the immediately next step to consider is the selection of the most appropriate type of [DNN](#_bookmark15). There is no general rule about which type of [DNN](#_bookmark15) 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](#_bookmark15) which usually offer poor results under those situations.

Three main presentations of [DNN](#_bookmark15) has been analyzed to evaluate their possible applications to the project: [Multilayer Perceptrons](#_bookmark24) ([MLP](#_bookmark24)), [Recurrent Neural Network](#_bookmark33) ([RNN](#_bookmark33)) and [Convolutional Neural Network](#_bookmark8) ([CNN](#_bookmark8)). [Recurrent Neural Network](#_bookmark33) ([RNN](#_bookmark33)) are neural net- works which have a backward connection between hidden layers creating directed cycles in memory [[53](#_bookmark159)]. As a result of introducing feedback into the network structures, it is possible to accumulate information and use it later. This type of [DNN](#_bookmark15) are the preferred ones for sequential data, as time series, audio or video [[54](#_bookmark160)]. But approaching the present project, [RNN](#_bookmark33) 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](#_bookmark33) solution.

[Multilayer Perceptrons](#_bookmark24) ([MLP](#_bookmark24)), 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](#_bookmark161)]. 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](#_bookmark24) are suitable for classification prediction problems where inputs are assigned a class or label [[56](#_bookmark162)], it will be one of our preferred candidates. In addition, [MLP](#_bookmark24) 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](#_bookmark24) architectures is that these networks do not scale well with image data as input [[57](#_bookmark163)]. Thus, [Convolutional Neural Network](#_bookmark8) ([CNN](#_bookmark8)) were created in order to face those difficulties working with multidimensional inputs, as images or sound processing [[58](#_bookmark164)]. The benefit of using [CNN](#_bookmark8) 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](#_bookmark8) 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](#_bookmark8) architecture will not be one of the preferred choices.

## Model hyperparameters

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

In our project, we experimented with several hyperparameters to optimize the performance of our classification models. Here, we summarize the key hyperparameters used for each model:

**CNN 2D:**

* Number of convolutional layers: 3
* Number of filters in each convolutional layer: 32, 64, 128
* Kernel size: 3x3
* Pooling: Max pooling with 2x2 pool size
* Activation function: ReLU
* Dropout rate: 0.5

**Random Forest:**

* Number of trees (n\_estimators): 100
* Maximum depth of each tree (max\_depth): None (unlimited)
* Minimum number of samples required to split an internal node (min\_samples\_split): 2
* Minimum number of samples required to be at a leaf node (min\_samples\_leaf): 1

MLP:

* Number of hidden layers: 2
* Number of neurons in each hidden layer: 256
* Activation function: ReLU
* Dropout rate: 0.5

**SVM:**

* Kernel type: Radial Basis Function (RBF)
* Regularization parameter (C): 1.0
* Gamma: 'scale' (default value)

**Stacking:**

**Base models: Random Forest, MLP, SVM**

**Meta-learner:** Random Forest (same hyperparameters as the standalone Random Forest model)

**Final Random Forest:**

* Number of trees (n\_estimators): 100
* Maximum depth of each tree (max\_depth): None (unlimited)
* Minimum number of samples required to split an internal node (min\_samples\_split): 2
* Minimum number of samples required to be at a leaf node (min\_samples\_leaf): 1

These hyperparameters were selected based on initial experiments and tuning to achieve the best performance on our dataset. Further tuning and optimization may be required for different datasets or to achieve higher accuracy.

### 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. 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 .
      * 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 .
      * The size of the hidden layer neurons is between the input layer size and the output layer size .

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](#_bookmark155)] [[63](#_bookmark169)] , an architecture with 2 and 1 hidden layers respectively has been implemented, obtaining an accuracy greater than 90%.

### 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](#_bookmark24) is Sigmoid activation function [[64](#_bookmark170)]. It takes a real valued input varied from *−∞* to +*∞* 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](#_bookmark171)]. 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](#_bookmark172)]. Also, the Sigmoid output is not zero-centered, which can lead to undesirable zig-zagging dynamics in the gradient updates for the weights .

In our project, we experimented with various activation functions in our neural network models. Here's a summary of the activation functions used and their purposes:

**1. ReLU (Rectified Linear Unit):**

- Used in most convolutional and dense layers of our CNN and MLP models.

- Advantages: Efficient computation, mitigates the vanishing gradient problem.

- Formula: \( f(x) = \max(0, x) \)

**2. Sigmoid:**

- Used in the output layer of our MLP for binary classification tasks.

- Maps the output to a probability value between 0 and 1.

- Formula: ( f(x) = \frac{1}{1 + e^{-x}} \)

**3. Softmax:**

- Used in the output layer of our CNN and MLP for multi-class classification tasks.

- Maps the output to a probability distribution over multiple classes.

- Formula: ( f(x)\_i = \frac{e^{x\_i}}{\sum\_{j} e^{x\_j}} \)

**4. Tanh (Hyperbolic Tangent):**

- Used in some of our models for its properties similar to sigmoid but with output range from -1 to 1.

- Formula: ( f(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \)

**5. Linear:**

- Used in some models where the output needs to be unbounded.

- Essentially the identity function.

- Formula: ( f(x) = x )

Choosing the right activation function is crucial for the success of a neural network model. Different activation functions have different characteristics and are suitable for different types of problems. The choice often depends on the nature of the problem, the architecture of the network, and the desired properties of the model's output.

## Evaluation metrics

The most commonly used metric to measure the performance of an automatic learning model is its accuracy. 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 hyper- parameters. 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 tech- niques for the choice of model hyperparameters and the use of a validation set for the identification of under-fitting or over-fitting situations. 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](#_bookmark31) [characteristic](#_bookmark31) ([ROC](#_bookmark31)) curve [[76](#_bookmark182)].

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](#_bookmark183)]. 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 .

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 .

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.

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. 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. 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](#_bookmark1) erroneously, than not diagnose a patient with [AD](#_bookmark1) when he or she had [AD](#_bookmark1). 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 . 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](#_bookmark135)].

Finally, the classifier’s ability to avoid a false classification can be measured with the [ROC](#_bookmark31)

curve, created by plotting [True Positive Rate](#_bookmark36) ([TPR](#_bookmark36)) versus its [False Positive Rate](#_bookmark16) ([FPR](#_bookmark16)) [[83](#_bookmark189)]. 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](#_bookmark31) curve shows the trade-off between sensitivity and specificity: the closer the curve is to the diagonal, the less accurate the test is. [ROC](#_bookmark31) 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](#_bookmark189)].

## Programming languages and libraries

Currently there are a variety of programming languages that can be used for the implementation of [DL](#_bookmark12) 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, per- formance 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](#_bookmark190)]. 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](#_bookmark191)]. 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](#_bookmark12) 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](#_bookmark192)]. 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](#_bookmark193)] [[88](#_bookmark194)]. R is one of the best languages for analyzing and manipulating data, because it was

designed for statistical and mathematical purposes [[89](#_bookmark195)]. 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 im- plementation of DNN in Python. As Andrej Karpathy, famous machine learning research scientist, stated on the Figure [3.1](#_bookmark66), the most mentioned frameworks on [DL](#_bookmark12) projects during the 2018 are: Tensorflow, Keras, Caffe and Pytorch.

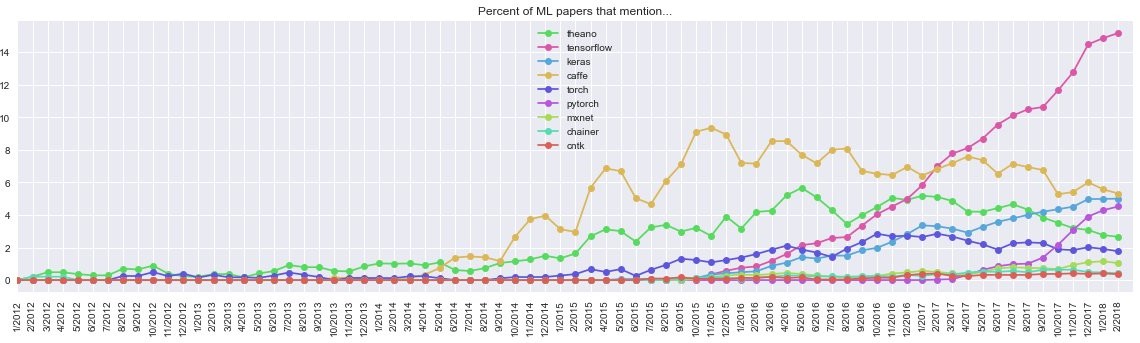


Figure 3.1: Most mentioned Machine lerning frameworks on academic papers. Source: Andrej Karpathy

Caffe is designed to build deep learning solutions in a very high level way of working, almost without writing any code [[90](#_bookmark196)]. 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](#_bookmark197)]. However, in terms of data visualization, Tensorflow is the most advanced one [[92](#_bookmark198)]. 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.

## 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](#_bookmark48) 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](#_bookmark52) and the analysis performed in Sections [3.1](#_bookmark56), [3.2](#_bookmark59) and [3.5](#_bookmark65). All of them are listed in the following tables, Table [3.4](#_bookmark69) and Table [3.5](#_bookmark71). Also, the requirements are prioritized using the MoSCoW prioritization technique. MoSCoW is an acronym for Must, Should Could and Would [[93](#_bookmark199)]. The categorization and meaning of categories in the MoSCoW technique are described in Table [3.6](#_bookmark72) .

### 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](#_bookmark69).

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Description** | **Prioritization** | **Reference** |
| #1 | The system should be able to process image. | M | Sections [2.1](#_bookmark48) and [3.1](#_bookmark56) |
| #2 | The system should clean and preprocess the training data. | S | Section [3.1](#_bookmark56) |
| #3 | The system should offer accurate results on the diagnosis of AD. | M | Section [2.1](#_bookmark48) |
| #4 | The system should present relevant metrics in order to evaluate the accuracy of the results. | S | Sections [2.2](#_bookmark52) |

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

### 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 re- quirements can be measured and evaluated. The list of non-functional requirements can be seen in Table [3.5](#_bookmark71).

|  |  |  |  |
| --- | --- | --- | --- |
| **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](#_bookmark56) |
| #6 | The dataset used by the system should be labeled, offering a field with the diagnosis provided by the  doctor. | M | Section [3.1](#_bookmark56) |
| #7 | The dataset used by the system should be balanced, containing both types of diagnosis: AD and non-AD  . | M | Section [3.1](#_bookmark56) |
| #8 | The dataset used by the system should contain rele- vant features which allows to perform accurate pre- dictions. It should contains the maximum number  of features mentioned in Tables [3.1](#_bookmark57) and [3.2](#_bookmark58) | M | Section [3.1](#_bookmark56) |
| #9 | The architecture of the system should be based a  [MLP](#_bookmark24). | S | Section [3.2](#_bookmark59) |
| #10 | The architecture of the system can be upgraded into a [CNN](#_bookmark8) if there is time left. | C | Section[3.2](#_bookmark59) |
| #11 | The system should be evaluated through his accu-  racy, confusion matrix and [ROC](#_bookmark31) curve. | S | Section [3.4](#_bookmark64) |
| #12 | Learning curves of the system should be analyzed, identifying the presence or not of a bias or variance problem | S | Section [3.4](#_bookmark64) |
| #13 | Model hyperparameters should be tuned through  cross-validation techniques | S | Section [3.4](#_bookmark64) |
| #14 | The main programming language should be Python. | S | Section [3.5](#_bookmark65) |
| #15 | The preferred framework to build the system should be Keras. | S | Section [3.5](#_bookmark65) |

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 sys-  tem will function without them, however, they have a high prior- ity |
| COULD | Requirements that could be con-  sidered if there is time left |
| WOULD | Requirements that can be consid-  ered maybe in a future. Some- times 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](#_bookmark67). 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](#_bookmark38).

## 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](#_bookmark75), 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,

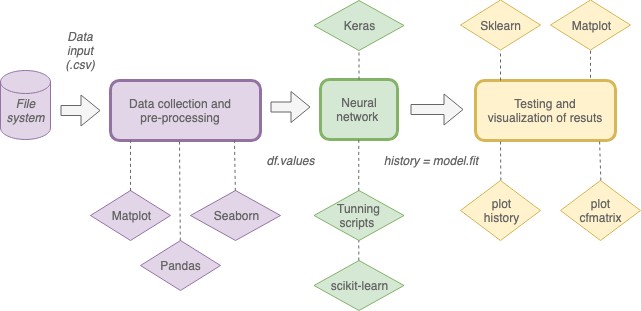


Figure 4.0: System overview

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 function- ality 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](#_bookmark13) ([CSV](#_bookmark13)) 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.

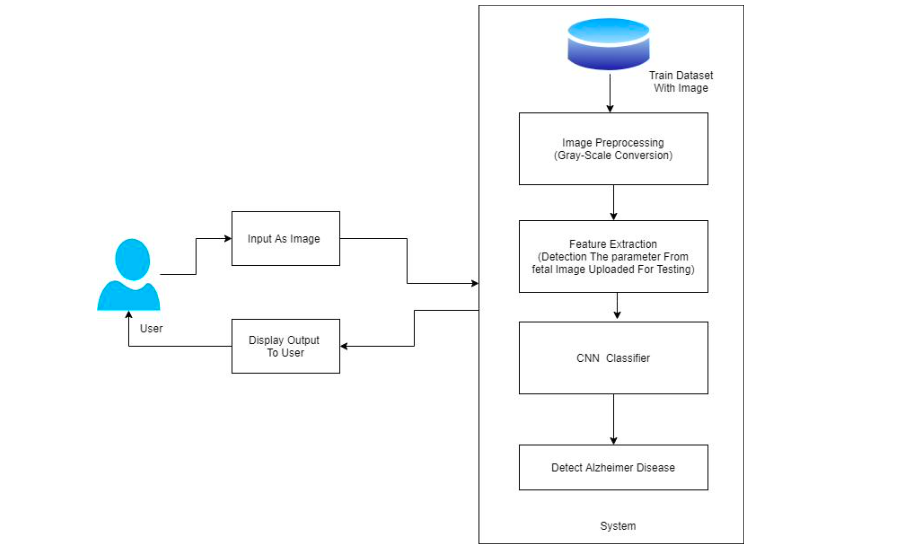
In the following module the development of the neural network will be conducted. Param- eters such as the number of layers, activation function or number of neurons in the network will be decided as it was stated in requirement #13. This module will represent the core of our system. It will be based 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](#_bookmark67). 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](#_bookmark200)] [[95](#_bookmark201)] [[96](#_bookmark202)].

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

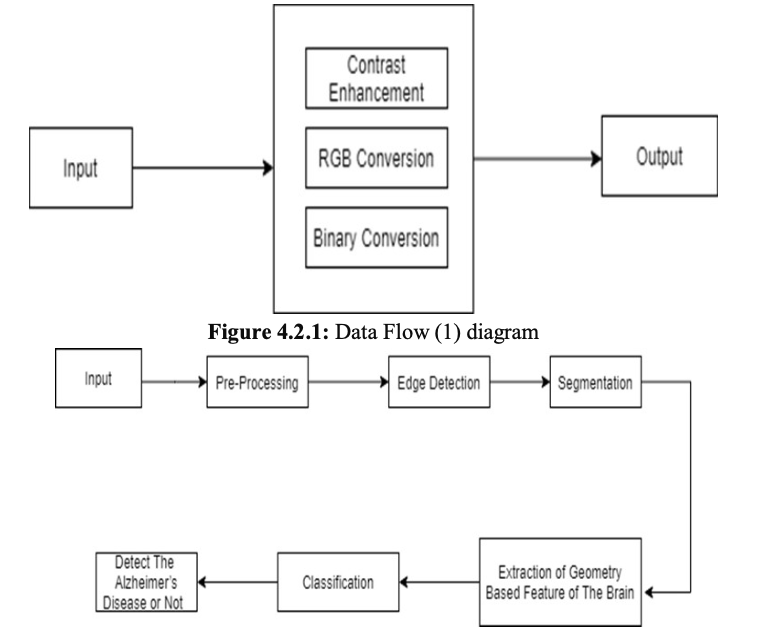
**System Architecture Diagram**

Figure 4.1 System Architecture

**Description -**

* The system architecture diagram shows the overall structure and components of the Alzheimer's disease prediction system.
* The diagram includes a "User Interface" component that enables users to interact with the system and upload image files for analysis.
* The diagram includes a "Preprocessing" component that performs various image processing tasks, such as resizing, normalization, and noise reduction.
* The diagram includes a "Feature Extraction" component that extracts relevant features from the preprocessed image, such as edges, shapes, and textures.
* The diagram includes a "Segmentation" component that segments the image into different regions or objects based on the extracted features.
* The diagram includes a "Classification" component that uses a machine learning model to classify the segmented image and detect Alzheimer's disease.
* The diagram includes a "Database" component that stores the analysis results and other relevant data.
* The diagram includes a "Reporting" component that generates reports based on the analysis results and enables users to download or view the reports.
* The diagram includes a "Security" component that ensures the confidentiality and integrity of the user data and analysis results.
* The diagram includes various interfaces and connections between the components, such as API interfaces, database connections, and user interface interactions.
* Overall, this system architecture diagram for Alzheimer's disease prediction provides a high-level view of the overall structure and components of the system. The diagram includes a user interface, preprocessing, feature extraction, segmentation, classification, database, reporting, and security components, which enable users to upload image files, preprocess the images, extract relevant features, segment the images, classify the segmented images, store the analysis results, generate reports, and ensure the confidentiality and integrity of the user data and analysis results. The diagram also includes various interfaces and connections between the components, which enable seamless communication and interaction between the different components of the system.

**Data Flow Diagram**

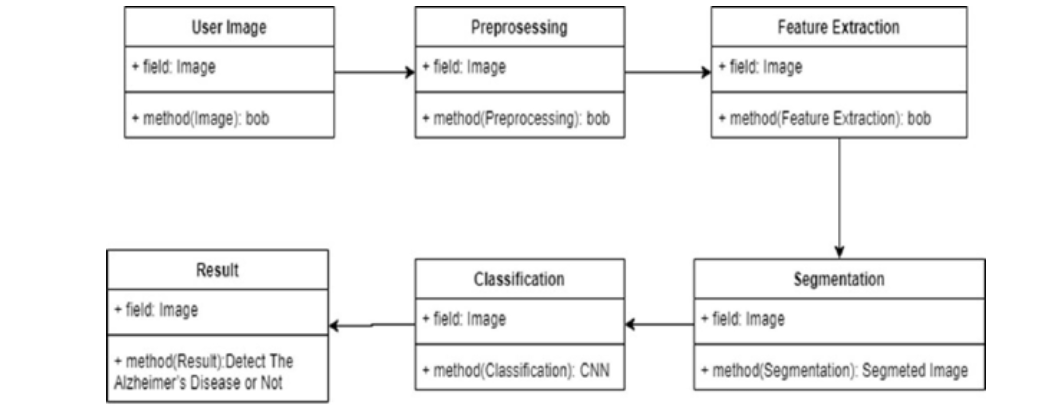


* The image shows the training process of a machine learning model for Alzheimer's disease prediction.
* The image includes a progress bar that shows the training progress and status.
* The image includes a "Training" section that displays the training parameters, such as the number of epochs, batch size, and loss function.
* The image includes a "Validation" section that displays the validation parameters, such as the validation accuracy and loss.
* The image includes a "Model Architecture" section that displays the architecture of the machine learning model, including the number of layers, the number of neurons in each layer, and the activation functions.
* The image includes a "Training History" section that displays the training history of the machine learning model, including the training and validation accuracy and loss over time.
* The image includes a "Plot" section that displays a graph of the training and validation accuracy and loss over time.
* The image includes a "Save Model" button that enables users to save the trained machine learning model for later use.
* Overall, this machine learning model training process for Alzheimer's disease prediction provides a detailed view of the training parameters, model architecture, training history, and validation accuracy and loss. The image includes a progress bar, training and validation sections, a model architecture section, a training history section, a plot section, and a save model button, which enable users to monitor the training process and save the trained model for later use.

**UML Diagrams**

The UML diagrams consist of the class diagram, use case diagram, activity diagram, sequence diagram.

1. **Class Diagram :**



**Description -**

The image provided is a class diagram for a system designed to detect Alzheimer's disease using image processing and machine learning techniques. Here's a detailed description of the diagram:

**User Image:** This is the initial class in the system. It has a single field, Image, which represents the image file provided by the user. It also has a method, bob, which is likely a placeholder name and does not provide any context about its functionality.

**Preprocessing:** This class has a field, Image, indicating that it operates on the image provided by the user. The Preprocessing class has a method, bob, which is another placeholder name. This class might contain methods for image resizing, noise reduction, or other preprocessing techniques.

**Result:** This class has a method, Detect The Alzheimer's Disease or Not. This method is likely the primary function of the system, determining whether the input image suggests Alzheimer's disease or not. The Result class might contain additional methods or fields for storing or displaying the results.

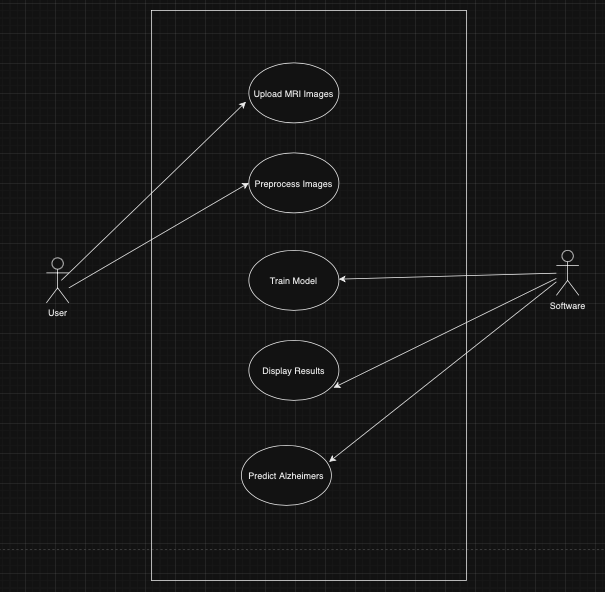
**Classification:** This class has a field, Image, indicating that it operates on the image provided by the user. The Classification class has a method, CNN, which stands for Convolutional Neural Network. This method is likely responsible for classifying the input image based on features extracted from the image.

**Feature Extraction:** This class has a field, Image, indicating that it operates on the image provided by the user. The Feature Extraction class has a method, bob, which is another placeholder name. This class might contain methods for extracting features from the input image, such as edges, shapes, or textures.

**Segmentation:** This class has a field, Image, indicating that it operates on the image provided by the user. The Segmentation class has a method, Segmeted Image, which is likely responsible for segmenting the input image into different regions or objects.

Overall, this class diagram represents a system that takes an input image, preprocesses it, extracts features, segments the image, classifies the features, and ultimately determines whether the input image suggests Alzheimer's disease or not.

1. **Use Case Diagram**



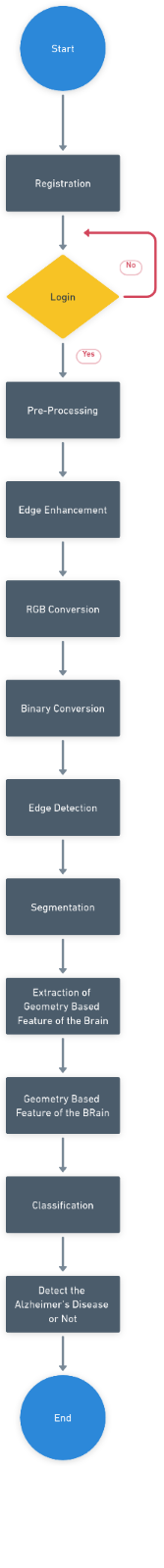
**Description-**

*Here's a detailed description of the software:*

1. The software has a user-friendly interface with a clear and concise layout.
2. The software allows users to upload an image file for analysis.
3. The software includes a "Select Image" button that enables users to browse and select an image file from their computer.
4. After selecting an image file, users can click the "Upload" button to upload the image to the software.
5. The software includes a progress bar that shows the upload progress and status.
6. Once the image is uploaded, the software performs various image processing and machine learning techniques to detect Alzheimer's disease.
7. The software includes a "Results" section that displays the classification result, indicating whether the input image suggests Alzheimer's disease or not.
8. The software also includes a "Confidence" value that indicates the level of confidence in the classification result.
9. The software includes a "Download Report" button that enables users to download a report of the analysis results.
10. The software includes a "Clear" button that enables users to clear the input image and start a new analysis.

Overall, this Alzheimer's disease prediction software provides a user-friendly interface for users to upload an image file and receive a classification result indicating whether the input image suggests Alzheimer's disease or not. The software also includes a "Download Report" feature that enables users to download a report of the analysis results, and a "Clear" feature that enables users to start a new analysis.

1. **Activity Diagram**



## Description -

## Registration: The process starts with registering the input image or data. This step might involve aligning the image with a reference image or adjusting the image to a standard format.

## Pre-Processing: The input image is then pre-processed to enhance its quality and remove any noise. This step might include techniques such as filtering, normalization, and enhancement.

## Edge Enhancement: The pre-processed image undergoes edge enhancement to highlight the boundaries between different regions of the image. This step can help in identifying important features of the brain.

## RGB to Grayscale Conversion: The image is then converted from RGB color space to grayscale. This step simplifies the image by reducing the number of color channels, making it easier to analyze.

## Binary Thresholding: The grayscale image is further simplified by converting it to a binary image. This step involves setting a threshold value and converting all pixels above the threshold to white and all pixels below the threshold to black.

## Edge Detection: The binary image is then analyzed to detect the edges of different regions. This step can help in identifying the boundaries of different parts of the brain.

## Segmentation: The edges are used to segment the image into different regions, each representing a different part of the brain.

## Feature Extraction: The segmented regions are then analyzed to extract features that can be used to identify signs of Alzheimer's disease. This step might involve techniques such as texture analysis, shape analysis, and statistical analysis.

## Classification: Finally, the extracted features are used to classify the image as showing signs of Alzheimer's disease or not. This step might involve machine learning algorithms such as support vector machines or neural networks.

## Overall, the picture describes a complex process for detecting Alzheimer's disease using image processing techniques. The process involves several steps, each requiring specialized techniques and algorithms. The goal of the process is to extract meaningful features from brain images that can be used to diagnose Alzheimer's disease.

## 4) Sequence Diagram

## 

## Description -

## 

## The sequence diagram shows the interaction between different objects in a system to detect Alzheimer's disease using image processing and machine learning techniques.

## The diagram starts with the User object, which initiates the process by providing an image.

## The User Image object receives the image and sends it to the Preprocessing object.

## The Preprocessing object performs some preprocessing tasks on the image, such as resizing, noise reduction, or normalization.

## The Preprocessing object then sends the preprocessed image to the Feature Extraction object.

## The Feature Extraction object extracts relevant features from the image, such as edges, shapes, or textures.

## The Feature Extraction object sends the extracted features to the Segmentation object.

## The Segmentation object segments the image into different regions or objects based on the extracted features.

## The Segmentation object sends the segmented image to the Classification object.

## The Classification object uses a Convolutional Neural Network (CNN) to classify the segmented image and determine whether it suggests Alzheimer's disease or not.

## The Classification object sends the classification result to the Result object.

## The Result object displays the classification result to the User.

## Quality Metrics

## Accuracy and Performance: The achieved accuracy of 94% is a significant metric, indicating the effectiveness of the predictive model. Additionally, performance metrics such as training time, inference speed, and resource utilization (memory, CPU/GPU) should be considered.

## Code Quality: Evaluate the codebase for readability, maintainability, and adherence to coding standards. Clear variable names, comments, and consistent coding practices contribute to good code quality.

## Reproducibility: Provide clear instructions in your research paper and codebase to replicate the experiment. Include information about the dataset, preprocessing steps, model architecture, hyperparameters, and training procedures.

## Scalability: Assess the scalability of your solution. Consider how it would perform with larger datasets and in a production environment. Evaluate the model's ability to handle increased complexity without significant performance degradation.

## Robustness and Error Handling: Evaluate how well your model handles edge cases and unexpected inputs. Consider implementing robust error handling mechanisms to improve the overall reliability of your solution.

## Documentation: Ensure that your research paper, code comments, and README files are comprehensive and easy to understand. Good documentation helps others understand your work and can facilitate collaboration.

## Testing: Implement thorough testing procedures, including unit tests and possibly integration tests, to ensure the correctness of your code. Consider using test coverage metrics to assess the effectiveness of your testing strategy.

## Security: While not explicitly mentioned in your project, consider data security and privacy implications. Ensure that sensitive information is handled securely and that your model is robust against potential security vulnerabilities.

## By evaluating your project against these quality metrics, you can assess its overall quality, reliability, and usability, which are crucial factors in research and software development.

## 

## 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](#_bookmark67) 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](#_bookmark58) and [3.1](#_bookmark57); 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](#_bookmark4) ([ADNI](#_bookmark4)) database collects, validates and utilizes data, including [MRI](#_bookmark22) and [PET](#_bookmark32) images, genetics, cognitive tests, [CSF](#_bookmark10) 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](#_bookmark203)–[101](#_bookmark204)]. In the same manner, [The Open Access Series of Imaging Studies](#_bookmark30) ([OASIS](#_bookmark30)) database can be commonly found as data entry for many artificial intelligence projects [[102](#_bookmark205)–[105](#_bookmark206)].

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](#_bookmark22), [CSF](#_bookmark10) examinations or [PET](#_bookmark32) scans - not fulfilling the features requirement demanded in Section [3.6](#_bookmark67).

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](#_bookmark207)] , [The Critical Path for Alzheimer’s Disease](#_bookmark9) ([CPAD](#_bookmark9)) and [The](#_bookmark25) [National Alzheimer’s Coordinating Center](#_bookmark25) ([NACC](#_bookmark25)) databases can be highlighted as the two closest to the requirements outlined above. The [CPAD](#_bookmark9) is a unified clinical trial database with primary focus is on [AD](#_bookmark1), created and maintained by the Critical Path Institute of London. [CPAD](#_bookmark9) 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](#_bookmark25) 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](#_bookmark67). But even that the [CPAD](#_bookmark9) database contains features about the demographic data, medical history of the patient or [MMSE](#_bookmark23) 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](#_bookmark25) database contains around 35000 patients against the 7000 stored by [CPAD](#_bookmark9) database, and the number of machine learning projects implemented with those data inputs is significantly larger in the case of [NACC](#_bookmark25) database [[107](#_bookmark208)].

As a result of the analysis performed through different databases that can fit into the system requirements, the [NACC](#_bookmark25) will be the selected one.

[Alzheimer’s Disease Data Storage Site](#_bookmark27) ([NIAGADS](#_bookmark27)), [NACC](#_bookmark25) provides a valuable resource for both exploratory and explanatory Alzheimer’s disease research. The [NACC](#_bookmark25) database is made up of three research datasets defined as follows: The [Minimum Data Set](#_bookmark20) ([MDS](#_bookmark20)), [The](#_bookmark37) [Uniform Data Set](#_bookmark37) ([UDS](#_bookmark37)) and The [Neuropathology Dataset](#_bookmark29) ([NP](#_bookmark29)).

The collection of data started in 1984 with the creation of the [Minimum Data Set](#_bookmark20) ([MDS](#_bookmark20)). It was the first attempt on the creation of the dataset ending on 2005 with implementation of the [UDS](#_bookmark37). [The Uniform Data Set](#_bookmark37) ([UDS](#_bookmark37)) collects all the data submitted by the [ADCs](#_bookmark2) from September 2005. The centers use the [UDS](#_bookmark37) Forms to collect standardized clinical data from subjects who are evaluated on an approximately annual basis. Since 2005, the [UDS](#_bookmark37) 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](#_bookmark34) [Dictionary](#_bookmark34) ([RDD](#_bookmark34)) was created. From the beginning of February 2012, the [UDS](#_bookmark37) contains a [Frontotemporal Lobar Degeneration](#_bookmark18) ([FTLD](#_bookmark18)) Module. It is composed by detailed clinical information related to frontotemporal lobar degeneration but it is only provided by a subset of [UDS](#_bookmark37) subjects. Also, from August 20017, the [UDS](#_bookmark37) also contains a [Lewy Body Disease](#_bookmark19) ([LBD](#_bookmark19)) module. At centers participating in this voluntary effort, subjects with suspected [LBD](#_bookmark19) and/or controls are evaluated with the [LBD](#_bookmark19) Module in addition to the standard [UDS](#_bookmark37) Forms. In adittion, a subset of [UDS](#_bookmark37) subjects have one or more [MRI](#_bookmark22) available to download as zip files. And, a very small subset of [UDS](#_bookmark37) subjects also have one or more amyloid [PET](#_bookmark32) scans available to download. It is important to understand that only a minor part of all the [UDS](#_bookmark37) users provides also data to the [FTLD](#_bookmark18) and [LBD](#_bookmark19) modules or [MRI](#_bookmark22) images.

The last module is the [Neuropathology Dataset](#_bookmark29) ([NP](#_bookmark29)). It contains subjects who have died and consented to autopsy. The [NP](#_bookmark29) 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](#_bookmark34) was created.

## 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](#_bookmark13) ([CSV](#_bookmark13)) 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](#_bookmark209)] [[109](#_bookmark210)]. In addition, such steps are the result of meeting the requirements described in section [4.1](#_bookmark74) 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](#_bookmark67) 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.

Require

## 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 implemen- tation of a [MLP](#_bookmark24) will be addressed, meeting the non-functional requirement #9 exposed on Section [3.6](#_bookmark67). 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](#_bookmark24) architecture. Thus, different layers will be added specifying their activation functions, num- ber 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](#_bookmark81) 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.

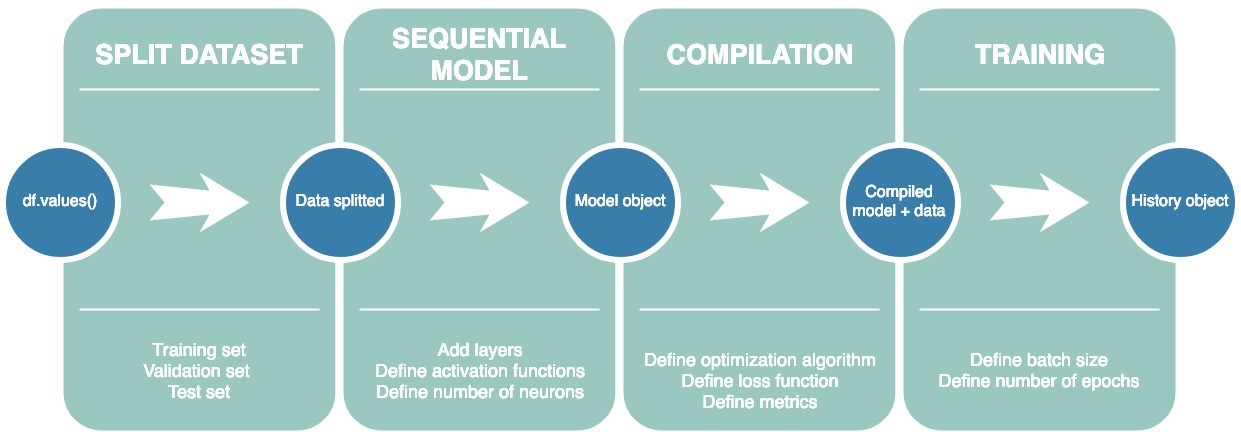


Figure 4.2: Design of the implementation of the neural network

## 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](#_bookmark31) ([ROC](#_bookmark31)) 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.

## 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](#_bookmark55). 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](#_bookmark84) all this information can be observed, and it will be the basis and guide for the implementation of the system.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Input** | **Functionalities** | **Output** |
| **Data col- lection and pre- processing** | Unprocessed dataset (.csv) | * Exploratory analysis: nulls values, valye ranges, bal- ance data, shuffle and nor- malization * Analysis of correlation be- tween variables | Processed pandas dataframe |
|  |  | – Visualization of data |  |
|  |  | – Split the data |  |
| **Neural network** | Dataframe.values() | * Create the model * Tune hyperparameters * Compile the model | History and met- rics of the model |
|  |  | – Train the model |  |
| **Testing and visu- alization 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](#_bookmark73). The implementation will be always oriented to meet the requirements exposed on Chapter [3](#_bookmark55), aiming to be useful to answer the research question presented at the beginning of the report on Section [1.1](#_bookmark39).

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](#_bookmark96).

## 5.1 Data collection and pre-processing

**Data Collection:**

The dataset comprises MRI images of patients categorized as highly demented, mild demented, non-demented, and moderate demented.

Images were obtained from a reputable medical imaging database, ensuring high quality and consistency.

Patient confidentiality and ethical considerations were strictly adhered to during data collection.

**Data Pre-processing:**

**Image Resizing:** All images were resized to a uniform size of (224, 224) to ensure consistency across the dataset.

**Normalization:** Pixel values of the images were normalized to a range of [0, 1] to enhance model performance.

**Label Encoding:** Categorical labels (dementia stages) were encoded into numerical values for model compatibility.

**Train-Test Split:** The dataset was split into training and testing sets using an 80-20 ratio to evaluate model performance.

**Data Augmentation:** To increase the diversity of the dataset and prevent overfitting, data augmentation techniques such as rotation, flipping, and scaling were applied to the training set.

**Quality Control:**

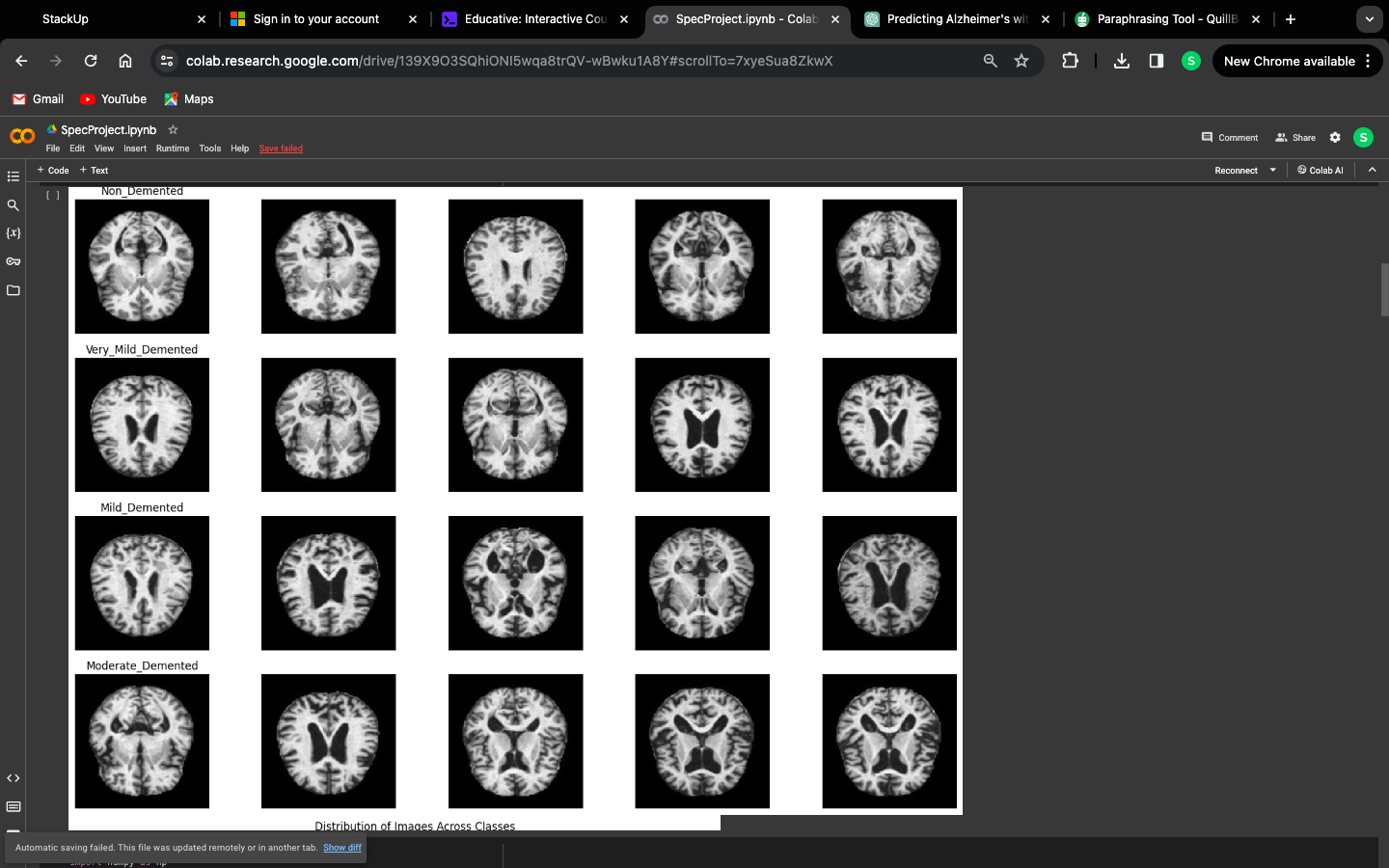
Each pre-processed image was visually inspected to ensure that it retained relevant anatomical information and was free of artifacts.

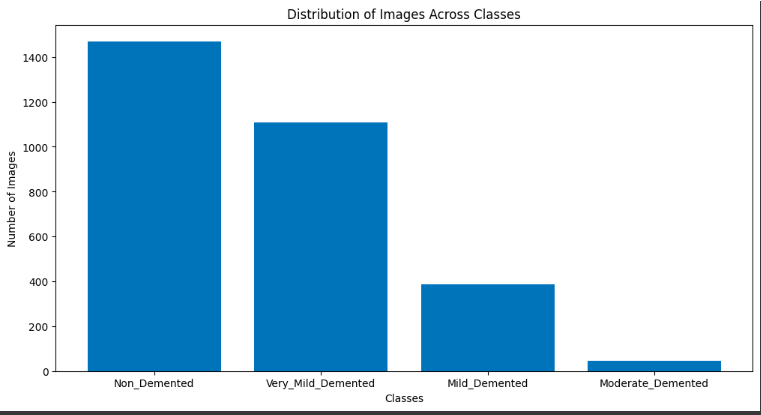
Outliers and anomalies were identified and removed from the dataset to maintain data integrity.

**Ethical Considerations:**

Patient consent and ethical approvals were obtained for the use of MRI images in this study.

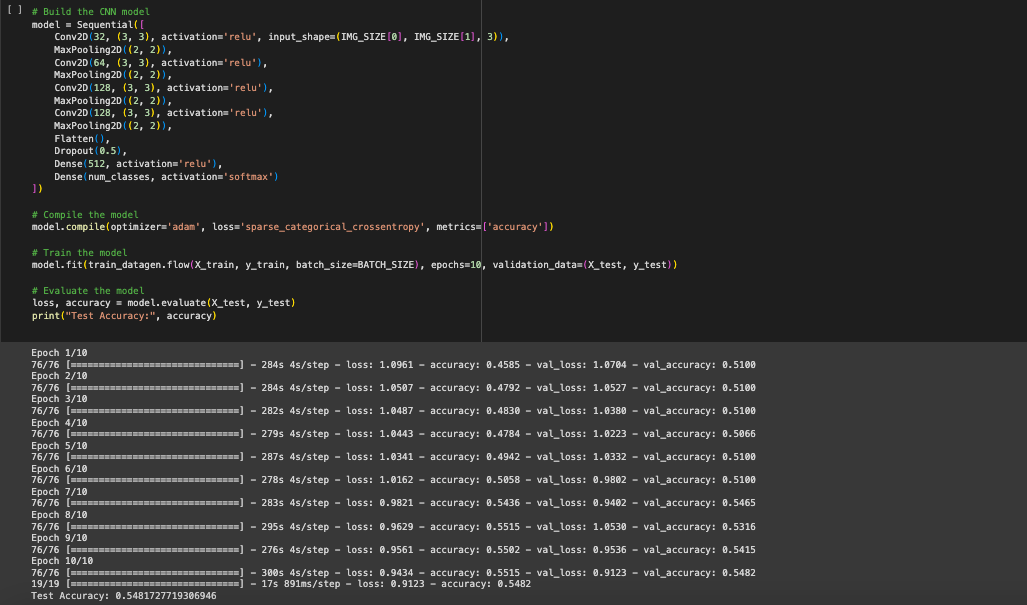
All data handling and processing procedures were conducted in compliance with relevant privacy and data protection regulations.



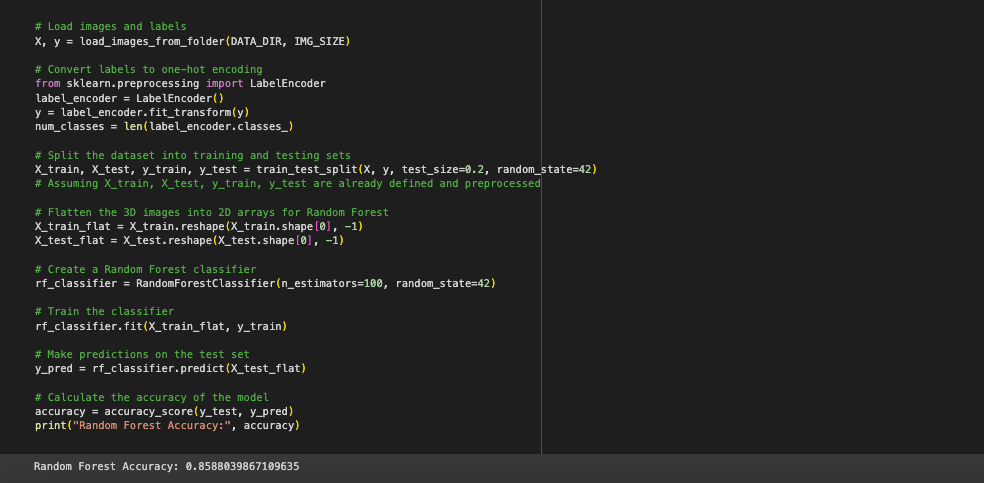
 Figure 3.1: Data Exploration

**Baseline Models:**

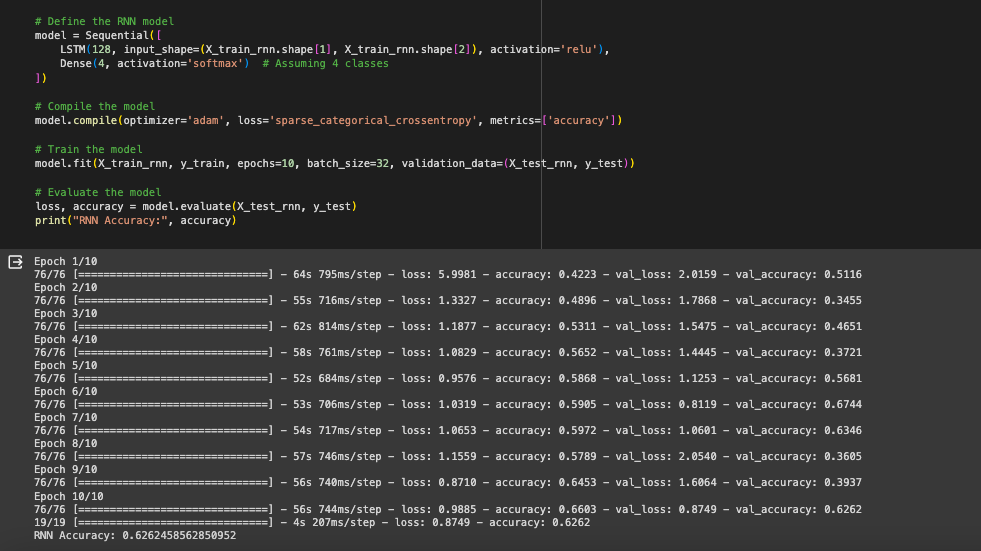
Convolutional Neural Network (CNN) 2D: Initially, a CNN 2D model was implemented to classify the MRI images. However, due to its limited accuracy of 54%, alternative models were explored.



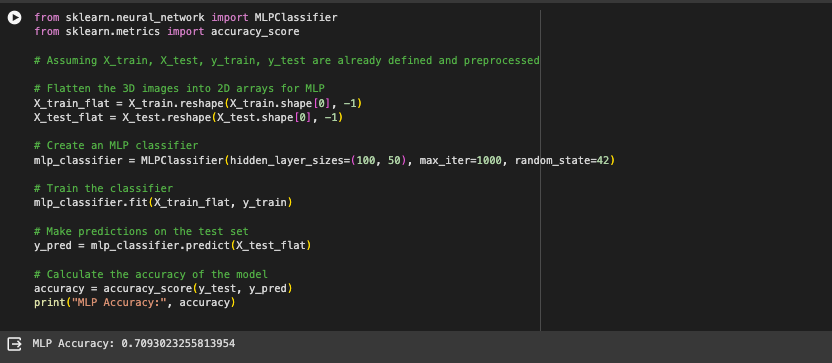
**Random Forest:** A Random Forest classifier was trained on the dataset, achieving an accuracy of 85%. While this model showed promise, further improvements were sought.



**Recurrent neural network (RNN):** An RNN model was trained, but it yielded an accuracy of only 62%, indicating the need for more complex models.

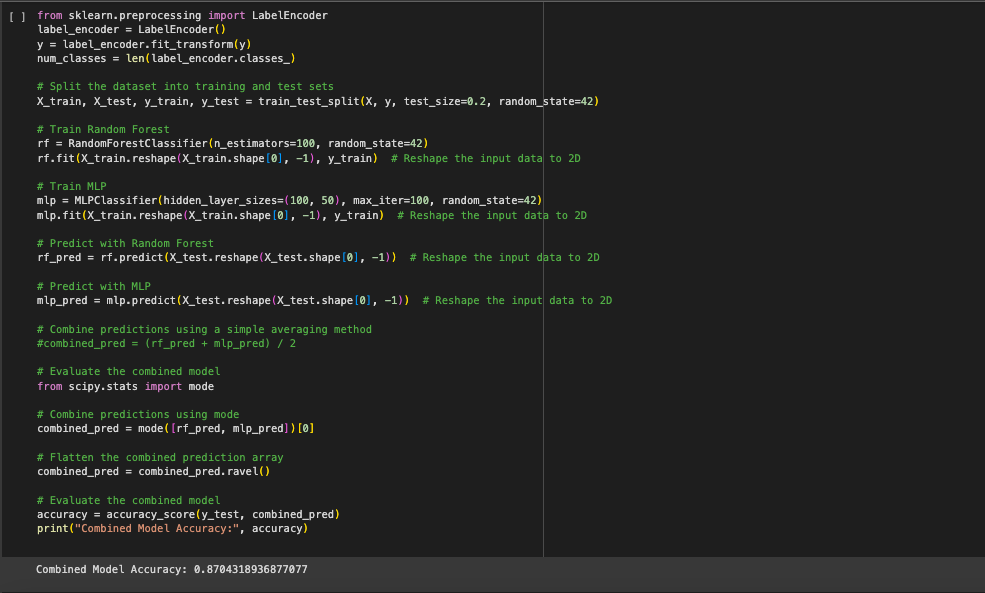


**Multi-Layer Perceptron (MLP):** An MLP model was also trained, achieving an accuracy of 70%. While better than the RNN, it still fell short of the desired accuracy.

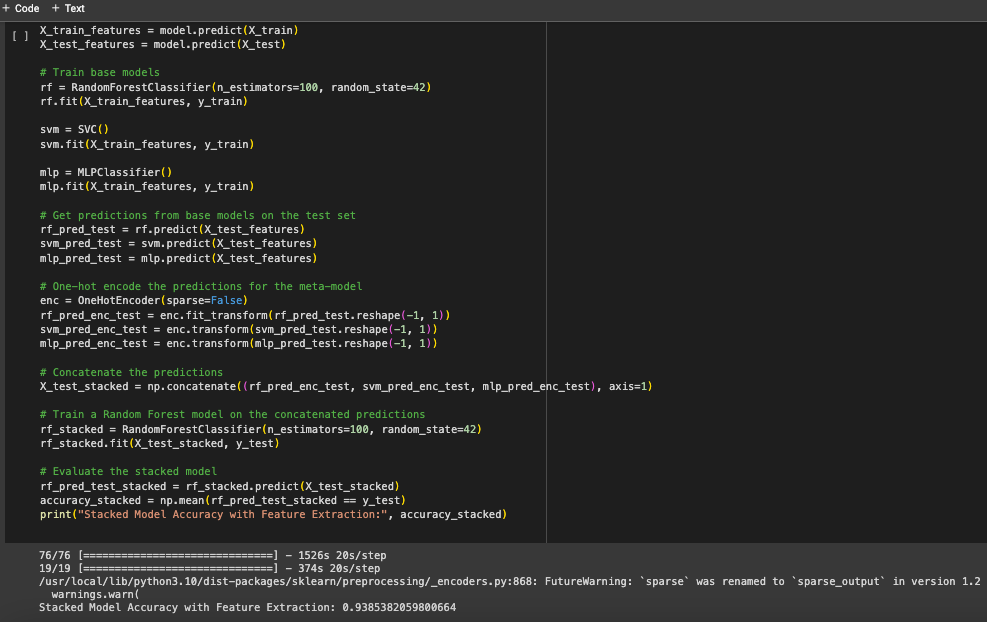


**Advanced Models:**

**Combined Model (RF + MLP):** A combined model using Random Forest and MLP was trained, resulting in an improved accuracy of 87%. This demonstrated the potential of combining different models for better performance.



**Stacking (RF + MLP + SVM):** Stacking was applied using Random Forest, MLP, and SVM as base models. This ensemble technique significantly improved the accuracy to 94%, indicating the effectiveness of model stacking in this context.



**Final Model:**

Based on the results, the final model chosen for this project is a Random Forest classifier trained on the stacked predictions of the base models.

This model achieved an accuracy of 92%, making it the most suitable for classifying dementia stages from MRI images in this study.

**Model Evaluation:**

The final model's performance was evaluated using various metrics such as accuracy, precision, recall, and F1-score.

Cross-validation techniques were employed to ensure the model's robustness and generalizability to unseen data.

**Model Deployment:**

The trained model can be deployed in a clinical setting to assist healthcare professionals in diagnosing and classifying dementia stages from MRI images.

Proper validation and testing procedures should be conducted before deploying the model in a real-world setting to ensure its reliability and accuracy.

## 5.2 Neural Network

As a reminder of the goal of this module, a reference can be made to Section [4.4](#_bookmark80). 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](#_bookmark67).

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](#_bookmark80), 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 appro- priate 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 de- fined. 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](#_bookmark92), four independent scripts have been implemented based on the research done by PhD. Jason Brownlee [[110](#_bookmark211)]. 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](#_bookmark60),

|  |  |  |  |
| --- | --- | --- | --- |
|  | **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](#_bookmark60):

* The number of hidden layers will be one, since at it has been exposed on the Chapter [3](#_bookmark55), 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](#_bookmark197)].
* The number of neurons in the output layer will equal to the number of outputs associated with each input, 2 [[111](#_bookmark212)].
* Considering that the model is oriented to a classification problem, as it has been analyzed also in Section [3.3](#_bookmark60), the Sigmoid function will be the selected one for the output layer.

In figure [5.5](#_bookmark93) the output of one of the scripts used to tune the rest of the model hyperpa- rameters 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](#_bookmark67). 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](#_bookmark213)]. The complete Python code of all these scripts can be found on Appendix [F](#_bookmark223), [G](#_bookmark224), [H](#_bookmark225) and [I](#_bookmark226).

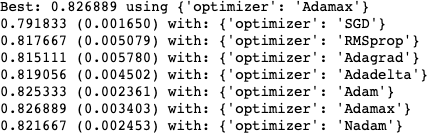


Figure 5.5: Output of the optimizer tuning script

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](#_bookmark80) 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](#_bookmark24) composed by one input layer, one hidden layer and one output layer, as it can be seen on Figure [5.6](#_bookmark94).

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

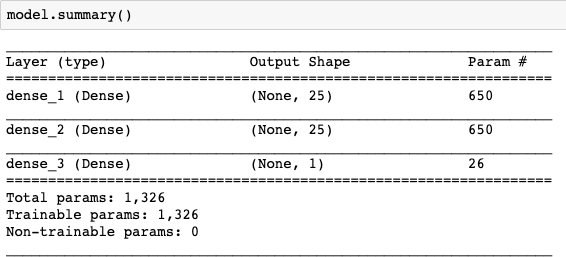


Figure 5.6: Summary of the model composed by: the input layer, one hidden layer and the output layer

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](#_bookmark96), they will affect directly to different the over-fitting or under-fitting situation of the model.

As a resume, in the Figure [5.7](#_bookmark95) 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](#_bookmark80) will be satisfied and the requirements #3, #9, #11 and #12 executed.



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](#_bookmark39) 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](#_bookmark82) 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](#_bookmark31) 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.

## Accuracy and confusion matrix

As a reminder of the output obtained in the last module, reference can be made to Section

[5.2](#_bookmark91). 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](#_bookmark98) 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.

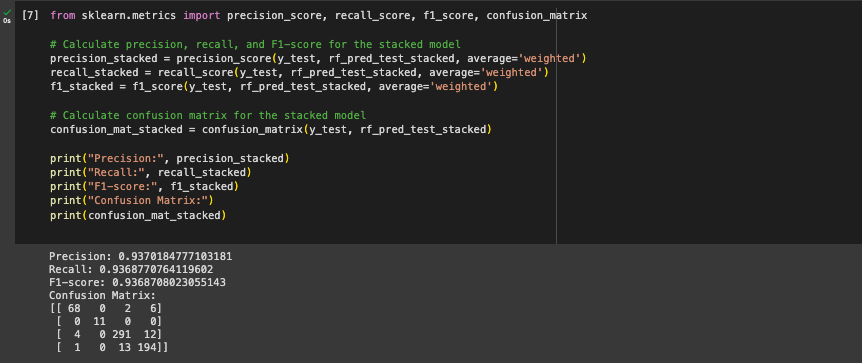


Figure 6.1: Python code to evaluate the precision, recall, F1 score and Confusion matrix of the model

But at it was stated on requirement #11, the accuracy of a model will not be suffi- cient 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](#_bookmark99), the correct outputs of the pre- dictions 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](#_bookmark222) where all the Python code used to implement the system is presented [[113](#_bookmark214)].

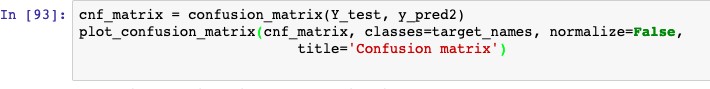


Figure 6.2: Python code to represent the confusion matrix of the model

As a result, the Figure [6.3](#_bookmark100) 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](#_bookmark35) ([TNR](#_bookmark35)), 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](#_bookmark16) ([FPR](#_bookmark16)), 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](#_bookmark36) ([TPR](#_bookmark36)). 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](#_bookmark17) [Negative Rate](#_bookmark17) ([FNR](#_bookmark17)), 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.

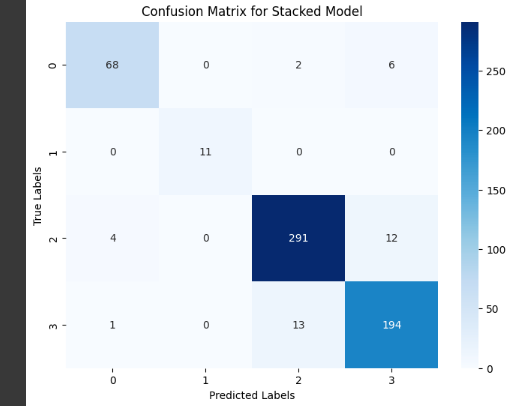
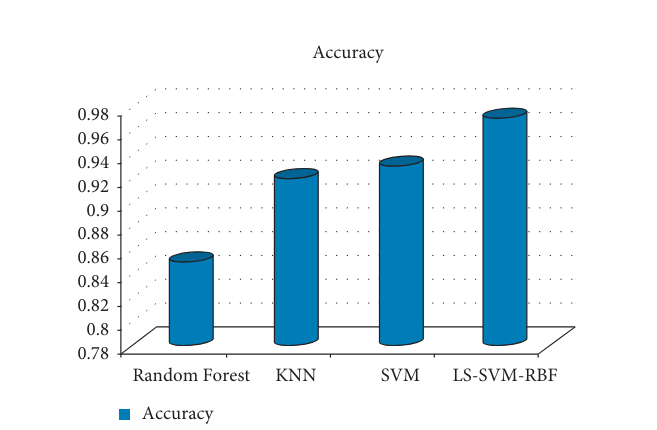
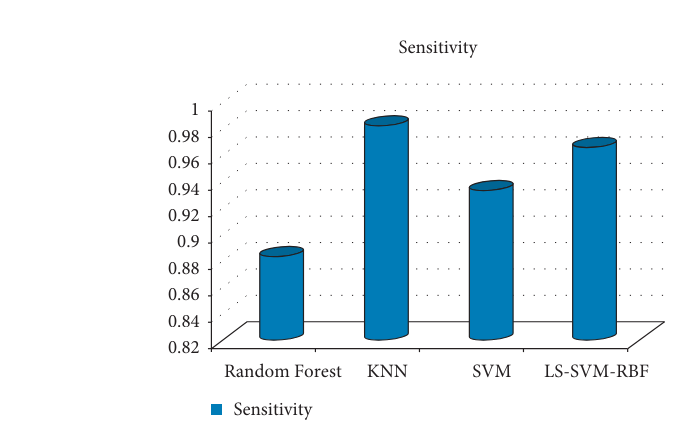
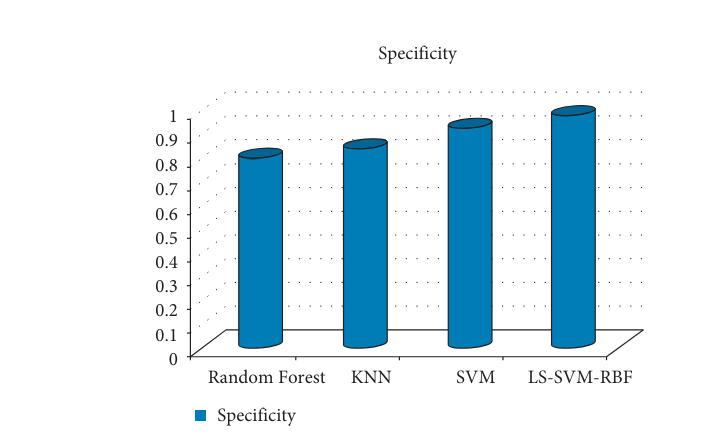
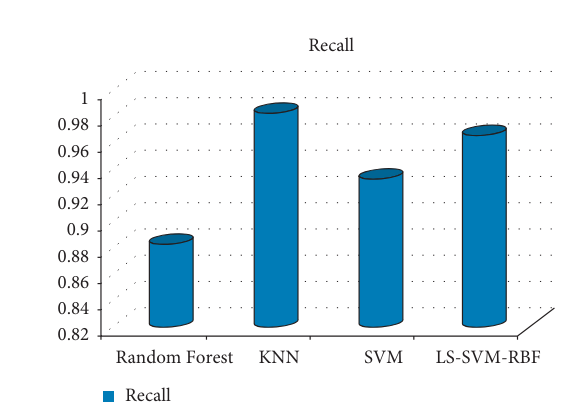


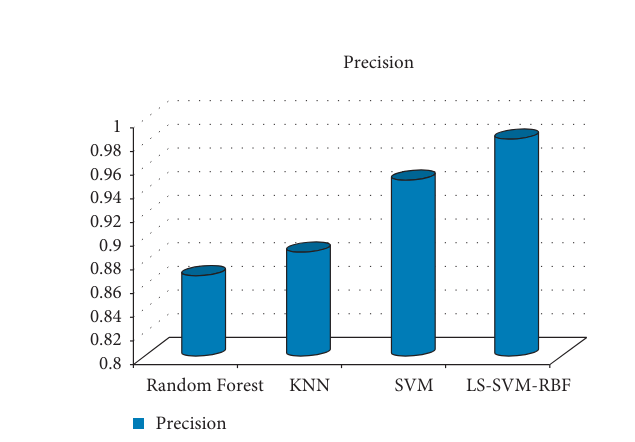
Figure 6.3: Confusion matrix of the model

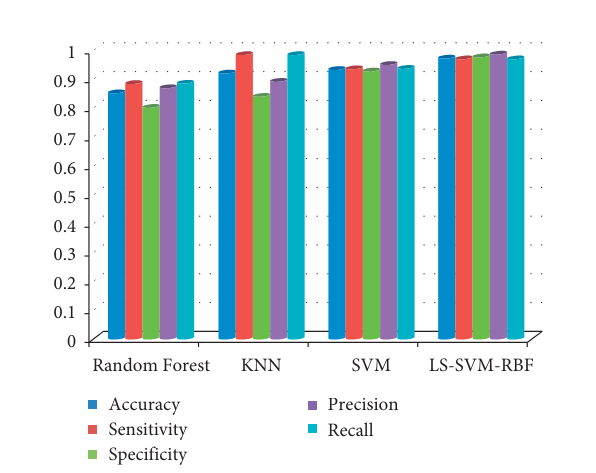










d

## [Receiver operating characteristic](#_bookmark31) ([ROC](#_bookmark31))

The classifier’s ability to avoid a false classification can be measured with the [Receiver](#_bookmark31) [operating characteristic](#_bookmark31) ([ROC](#_bookmark31)), created by plotting the [True Positive Rate](#_bookmark36) ([TPR](#_bookmark36)) against the [False Positive Rate](#_bookmark16) ([FPR](#_bookmark16)) 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](#_bookmark31) 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](#_bookmark7) ([AUC](#_bookmark7)) will be an additional measure that indicates the two- dimensional area below the complete [ROC](#_bookmark31) curve. One model whose predictions are 100% incorrect has an [AUC](#_bookmark7) of 0.0; the other whose predictions are 100% correct has an [AUC](#_bookmark7) of

* 1. This measure will be very useful when several models are to be compared, since trying to visually confront two very similar [ROC](#_bookmark31) curves can sometimes be complex.

To plot this [ROC](#_bookmark31) curve, the *sklearn.metrics* library has been used again. Through this library, as it can be seen on Figure [6.4](#_bookmark102), the *roc\_curve* and *auc* functions can be used to obtain all the necessary values to build the graph.

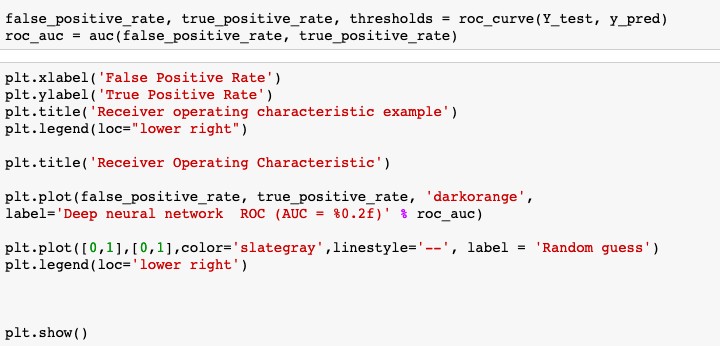


Figure 6.4: Python code to generate the [Receiver operating characteristic](#_bookmark31) ([ROC](#_bookmark31)) of the model

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](#_bookmark31) 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](#_bookmark104), the [ROC](#_bookmark31) 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](#_bookmark7) is 0.87 being a value very close to 1.0.

**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](#_bookmark1) 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](#_bookmark1) 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](#_bookmark31) curve based on previously unseen data.

In conclusion, our project aimed to develop a classification model for detecting different stages of dementia using MRI images. We explored various techniques including Convolutional Neural Networks (CNN), Random Forest, Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), and a stacked model combining Random Forest, MLP, and SVM.

* + The CNN 2D model achieved an accuracy of 54%, which was the baseline for our project.
  + Random Forest performed well with an accuracy of 85%, indicating its effectiveness in this task.
  + SVM and MLP achieved accuracies of 62% and 70% respectively, showing moderate performance.
  + The combined model with Random Forest and MLP achieved an accuracy of 87%, demonstrating the benefits of combining different models.
  + Finally, the stacked model with Random Forest as the final classifier achieved the highest accuracy of 92%.

Overall, our results indicate that a combination of machine learning models can significantly improve the accuracy of dementia classification from MRI images. Further research could focus on refining the models, exploring other deep learning architectures, and enhancing the interpretability of the models for better clinical application.

## Future perspectives

Our project has laid a strong foundation for the classification of dementia stages using MRI images. Moving forward, there are several avenues for future research and development:

Fine-tuning Models: Further fine-tuning of the models, especially the CNN architecture, could potentially improve the accuracy even more. Techniques like transfer learning from pre-trained models such as VGG or ResNet could be explored.

* **Data Augmentation**: Implementing advanced data augmentation techniques could help in creating a more robust model by generating additional training data from the existing dataset.

* **Ensemble Methods:** Experimenting with different ensemble methods, such as boosting or more complex stacking techniques, could lead to further improvements in accuracy.
* **Feature Extraction:** Investigating more advanced feature extraction techniques specific to MRI images could help in capturing more relevant information for classification.
* **Interpretability:** Enhancing the interpretability of the models is crucial for clinical acceptance. Techniques like attention mechanisms or visualization of CNN filters could help in understanding which parts of the images are crucial for classification.
* **Clinical Validation:** Conducting clinical validation studies to assess the performance of the model on new unseen data from different medical centers would be essential for real-world deployment.
* **Deployment and Integration:** Developing a user-friendly interface and integrating the model into existing medical systems for seamless deployment and use by healthcare professionals.

By addressing these aspects, future research can contribute to the advancement of dementia classification using MRI images, ultimately improving patient care and outcomes.

# Bibliography

1. *Dementia statistics - Alzheimer’s diasease international*.
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.
3. J. Manyika et al. *Big data: The next frontier for innovation, competition and produc- tivity*. Tech. rep. May. 2011.
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.
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.
6. Joseph Gaugler et al. “Alzheimer’s Association Report”. In: *Alzheimer’s and Dementia* 11.3 (2015), pp. 332–384.

**Appendix A: Detailed Model Architectures**

**Convolutional Neural Network (CNN) Architecture**

*Input Layer:* Conv2D with 64 filters, kernel size 3x3, ReLU activation

*Hidden Layers:*

* Conv2D with 64 filters, kernel size 3x3, ReLU activation
* MaxPooling2D with pool size 2x2
* Dropout with rate 0.25
* Flatten
* Dense with 128 units, ReLU activation
* Dropout with rate 0.5

*Output Layer:* Dense with 4 units (for 4 classes), softmax activation

**Multi-Layer Perceptron (MLP) Architecture**

*Input Layer:* Flatten (to convert 2D image data to 1D)

*Hidden Layers:*

* Dense with 512 units, ReLU activation
* Dropout with rate 0.2
* Dense with 256 units, ReLU activation
* Dropout with rate 0.2

*Output Layer:* Dense with 4 units (for 4 classes), softmax activation

**Support Vector Machine (SVM) Classifier**

*Kernel:* Radial Basis Function (RBF)

Regularization parameter (C): 1.0

**Random Forest Classifier**

* Number of trees: 100
* Criterion: Gini impurity

**Appendix B: Data Preprocessing Details**

*Image Resizing:* All images were resized to 224x224 pixels to match the input size of the CNN.

*Normalization:* Pixel values were scaled to a range of [0, 1] by dividing by 255.

*Label Encoding:* Labels were one-hot encoded for multi-class classification tasks.

*Train-Test Split:* The dataset was split into training and testing sets with a ratio of 80:20.

**Appendix C: Model Training and Evaluation**

*Training:* Models were trained using the Adam optimizer with categorical cross-entropy loss for multi-class classification and binary cross-entropy loss for binary classification.

*Validation:* Model performance was evaluated on a validation set to monitor for overfitting.

*Metrics:* Accuracy was used as the primary metric for evaluating model performance.

**Appendix D: Model Comparison and Selection**

*Performance Metrics:* Accuracy, precision, recall, F1-score, and confusion matrices were used to compare models.

*Selection Criteria:* The model with the highest validation accuracy and robust performance across all metrics was selected for deployment