A project report

on

EARLY DETECTION AND CLASSIFICATION OF ALZHEIMER’S DISEASE

Submitted in partial fulfilment for the award of the degree of

Bachelor of Engineering

in

Computer Science and Engineering

by

MD AMJAD ALI (D/18/CS/102)

NIKITA SINGH (D/19/CS/201)

under the guidance of

Mr. Aswini Kumar Patra

(Assistant Professor)

DEPARMENT OF COMPUTER SCIENCE AND ENGINEERING
North Eastern Regional Institute of Science and Technology
(Deemed to be University)
Under Ministry of Education, Govt. of India
Nirjuli, Arunachal Pradesh-791109
June 2022

Contents

Page no

Candidate's Declaration **4**

Certificate of Approval **5**

Acknowledgement **6**

Abstract **7**

CHAPTER 1: INTRODUCTION

1.1 **Introduction** **8**

1.2 **Existing System** **8**

1.3 **Problem Statement** **8**

1.4 **Proposed system** **9**

CHAPTER 2: Literature

2.1 **Literature survey** **10**

CHAPTER 3: System architecture and design

3.1 System design **11**

3.2 Flow chart diagram **11**

3.3 Use case diagrams **12**

3.4 Data flow diagram **12**

3.5 Modules Description **13**

CHAPTER 4: IMPLEMENTATION TECHNOLOGIES **16**

CHAPTER 5: TESTING AND IMPLEMENTATION

5.1 Software testing introduction **21**

5.2 Explanation for SDLC & STLC **21**

5.3 PHASES OF SOFTWARE DEVELOPMENT 22

5.4 SDLC Models 22

5.5 STLC (Software Testing Life Cycle) 22

5.6. TYPES OF TESTING 23

5.7 LEVEL OF TESTING USED IN PROJECT 23

CHAPTER-6 RESULT AND ANALYSIS

6.1. Code Screen shots 24

6.2. Result and Analysis 32

6.3 Conclusion and Future Scope 33

References 34

Bibliography (8%) 35

CANDIDATE'S DECLARATION

This is to certify that project entitled "**Early Detection and Classification of Alzheimer's Disease**" which is submitted by us in partial fulfilment of the requirement for the award of degree in Computer Science & Engineering at **North Eastern Regional Institute of Science and Technology**, Nirjuli, Arunachal Pradesh comprises only our own work and due acknowledgement has been made in the text to all other material used.

Date:

Name of the Student:

Md Amjad Ali (D/18/CS/102)	Nikita Singh (D/19/CS/201)

पूर्वोत्तर क्षेत्रीय वर्ववज्ञान एर्ववम प्रौद्योगिकी सं स्थान
North Eastern Regional Institute of Science & Technology
(Under the Ministry of Education, Govt. of India)
(Deemed to be University u/s 3 of the UGC Act, 1956)

ननर्ऱीजु - ७९१ १०९ (ईटागनर)
अरुणाचर्ऱ प्रदेश, भारत

Nirjuli-791 109 (Itanagar)
Arunachal Pradesh, India

CERTIFICATE OF APROVAL

It is to certify that **Md Amjad Ali (D/18/CS/102), Nikita Singh (D/19/CS/201)** had carried out the project work presented in this project entitled "**Early Detection and Classification of Alzheimer's Disease**" for the award of **Degree** in **Computer Science & Engineering at North Eastern Regional Institute of Science and Technology**, Nirjuli, Itanagar, Arunachal Pradesh (Deemed to be University), under my supervision. The items in the undertaking don't frame the reason for the honour of some other degree to the applicant or to any other person.

Date:

(Mr. Aswini Kumar Patra)
Project Guide

(Mr. Aswini Kumar Patra)
Project Coordinator

Dr. Moirangthem Marjit Singh

Head of Department

Computer Science and Engineering

ACKNOWLEDGEMENT

An effective and palatable consummation of any task is the result of important and total commitment of various individual pulls in spiral bearings, expressly or verifiably.
Like each enormous thing, this task isn't just our work. A ton of other people who upheld us in different ways, while some in specialized way, and some, in moral way.
We might want to offer our genuine thanks to our Project guide Mr. Aswini Kumar Patra, Assistant Professor, Dept. of Computer Science and Engineering, NERIST who has helped us a great deal in giving all offices required and legitimate direction and coordination in finishing the undertaking inside determined time for the culmination of this venture. The ideas and analysis of these individuals altogether worked on the venture.

MD Amjad Ali

Roll No: D/18/CS/102

And

Nikita Singh

Roll no: D/19/CS/201

ABSTRACT

Alzheimer is a neurological problem. For the Alzheimer, there is no particular treatment. Early location of Alzheimer's sickness can assist patients with getting the right fix. Many investigations utilize measurable and AI strategies to analyse AD. The human-level execution of Deep Learning calculations has been actually displayed in various disciplines. In the proposed approach, the MRI information is utilized to distinguish the AD and Deep Learning procedures are utilized to arrange the current sickness. The arrangement of Alzheimer’s illness utilizing profound learning techniques has shown promising outcomes, however effective application in clinical settings requires a mix of high precision, short handling time, and generalizability to different population.

CHAPTER-1

1.1 Introduction

This project will contain 2 main sections. Alzheimer Detection and Alzheimer Classification. Detection models (PCA, SVM, LDA, VGG16) will be making judgements about the test data to see whether this person's brain image shows Alzheimer signs.
On the other hand, Alzheimer Classification will be done through training the data with different severity for Alzheimer. Actually, this is how the original data is made, and this model can be used as a detector as well. As ML model will provide what stage the patient is, it may be used as a supportive tool for doctors to diagnose Alzheimer's Disease. It brings some light in the practical world as it may find early stages of AD, which can increase life expectancy as well as increased life quality with supportive treatments.

1.2 Existing System

In the past methodology, different shape and surface elements are extricated from the hippocampus for location of Alzheimer's sickness. The surface elements are separated utilizing Grey-Level Co-Occurrence Matrix (GLCM). GLCM is a technique utilized for extricating second request measurable surface highlights. The shape highlights are extricated utilizing the seven second invariants. The Moment invariants characterize set properties of district that can be utilized for ID of shape. In view of the elements separated from the hippocampus, AD can be characterized into different stages utilizing a prepared classifier. GLCM is an old technique, and accuracy also varies.

1.3 Problem Statement

Alzheimer is known the most for the causes of dementia. It takes 2/3 of the whole dementia population, while the cause is still unknown. According to papers from Lancet neurology, even the old theory such as neural inflations, which was disregarded as the cause compared to tau-protein, beta-amyloid’s, and genetic factors (Apo E4).
Data contains 4 classes of Alzheimer that depends on the severity of dementia.

There are degrees of severity in Alzheimer.

1. Very mildly demented: This is the stage where patient starts to forget where they put their stuff, other people's names recently, etc. It is hard to detect through cognitive ability test.
2. Mildly demented: This is the stage where patients don't remember the words, can't find their way to the destination, loss of focus and work-abilities. This is also the stage where patients even forget that they are losing memory. From this stage, with cognitive testing, it can be found.
3. Moderately demented: Starts to forget the recent activities, important old histories, have hard time calculating the budget, hard to go outside alone, and loss of empathy.

There are 3 more stages in the moderately dementia, which in the terminal stage, the patient can't move on their own, while they lose the ability to speak. But I assume that the current dataset from Kaggle is considering all these stages merged inside 'Moderately demented' or not even considered.

Knowing these stages are important because the faster the stage the patient is at, the treatment will have higher effect in terms of slowing the process. If the dementia is found during the moderately demented stage, it is known that the patient will pass away in 3 years. (One of the reported cases is a rhythm guitarists Malcom from the band AC/DC was diagnosed severe dementia at 2018.)
Thus, having an AI that detects Alzheimer dementia in the early stage can allow longer life expectancy from the patient as well as higher life quality overall from the slowdown of dementia.
As Alzheimer can not only be found with cognitive ability testing, but also through MRI or CT by looking at the ventricles of the brain and cortical atrophy, the theoretical foundation on this project is solid. Doctors find the patient with Alzheimer's have a brain that have enlarged ventricles (that lies in the centre of the brain) as well as thinner cortical grey area of the brain.

1.4 Proposed system

In the proposed approach, this project will go through various ML models from basic PCA, LDA, Support Vector Machine with 3 different kernels starting with linear to rbf kernel, and finally Convolutional Neural Network models VGG16 for Detection and EfficientNetB0 for Classification) and see which type works the best.
This project will contain 2 main sections. Alzheimer Detection and Alzheimer Classification. Detection models (PCA, SVM, LDA, VGG16) will be making judgements about the test data to see whether this person's brain image shows Alzheimer signs.
On the other hand, Alzheimer Classification will be done through training the data with different severity for Alzheimer. Actually, this is how the original data is made, and this model can be used as a detector as well. As ML model will provide what stage the patient is, it may be used as a supportive tool for doctors to diagnose Alzheimer's Disease. It brings some light in the practical world as it may find early stages of AD, which can increase life expectancy as well as increased life quality with supportive treatments.
Summary: I will try to implement diverse models for Alzheimer detection (non vs with Alzheimer) as well as Alzheimer Classifier (non vs very mild vs mild vs moderate)
Prepare the data in 2 different ways.

1. Alzheimer Detection: Whether the patient has the Alzheimer or not (non vs all other categories)
2. Alzheimer Classifier: Define what stage the patient is in the Alzheimer.

The models that are going to be tested here are

1. PCA for Alzheimer Detection
2. LDA for Alzheimer Detection
3. SVM for Alzheimer Detection and Alzheimer Classifier
4. CNN for Alzheimer Detection (VGG16) and Alzheimer Classifier (EfficientNetB0)

Advantages of Proposed System

- **PCA, SVM, LDA, VGG16, CNN is a new upcoming technology.**
- **Accuracy is high**.

CHAPTER-2

2.1 Literature survey

1 Alzheimer's sickness finding in light of physically separated surface examination of the hippocampus in underlying MRI

In this paper, we propose an original technique to coordinate surface and shape examination in a physically educated manner to demonstrate the hippocampus from primary MRI. In particular, Poisson map is utilized to delineate the fragmented hippocampus into layers, and a pack of-words (Bow) model on the picture force is produced to break down the surface data implanted over the layers. One more degree of Bow model is utilized to additionally encode the surface over subjects and across layers. A Bayesian non-parametric (BNP) technique is proposed to derive and adapt to the changeability of various layers. We approve our model on hippocampus-based Alzheimer's illness (AD) and gentle mental hindrance (MCI) analysis. An intensive correlation of related highlights is performed. Results exhibit that our strategy accomplishes cutting edge execution while utilizing data from the hippocampus as it were.

2. Early Detection of Alzheimer’s Disease using Image Processing

Alzheimer’s disease (AD) is an irreversible, progressive brain disorder that slowly destroy memory and thinking skills. Alzheimer’s is one of the most common cause of Dementia.
Dementia means loss of Cognitive functioning – thinking, remembering and reasoning – and behavioural ability to such an extent that it interferes with Daily life. The image processing is widely used in medical field in order to detect disease and help doctor in decision making based on observation. The paper aim to detect the Alzheimer’s disease at earliest so that patient can be prevented before irreversible changes occur in brain. We propose the image processing technique to process the Magnetic Resonance Imaging (MRI) of brain from axial plane, coronal plane and sagittal plane. The image segmentation is used to highlight the affected region in brain MRI. The diagnosed region in brain MRI include hippocampus and volume of brain. The comparative identification of person affected with the Alzheimer’s disease, Healthy cohort and Mild Cognitive impairment is done.

CHAPTER-3

System Architecture & Design

3.1 System design

A framework design chart would be utilized to show the connection between various parts. Generally, they are made for frameworks which incorporate equipment and programming and these are addressed in the outline to show the communication between them.

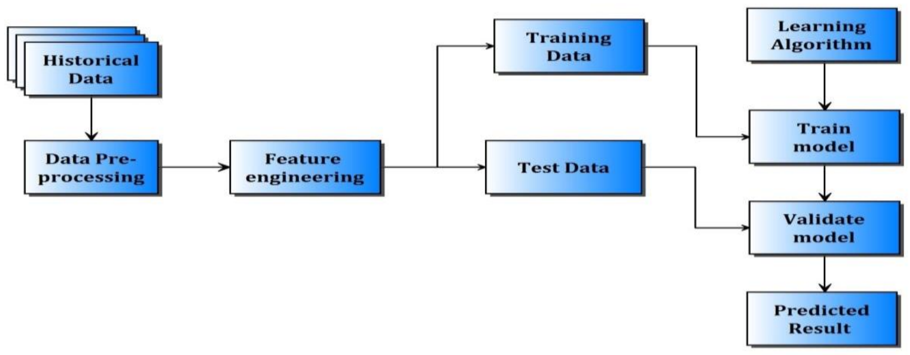


Fig 3.1 System Architecture

3.2 Flow chart diagram

Following through with all jobs and fulfil time constraints is significant. There are many undertaking the executive's devices that are accessible to assist with projecting directors deal with their assignments and timetable and one of them is the flowchart.

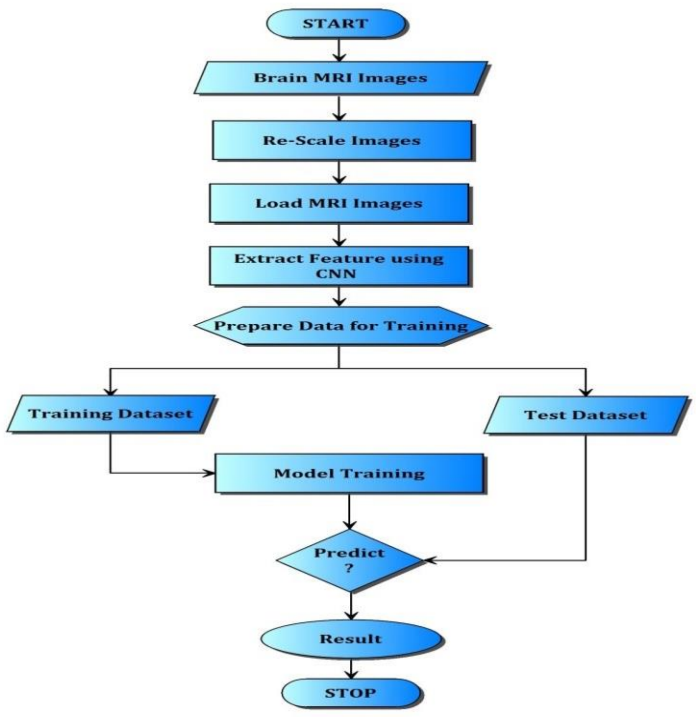
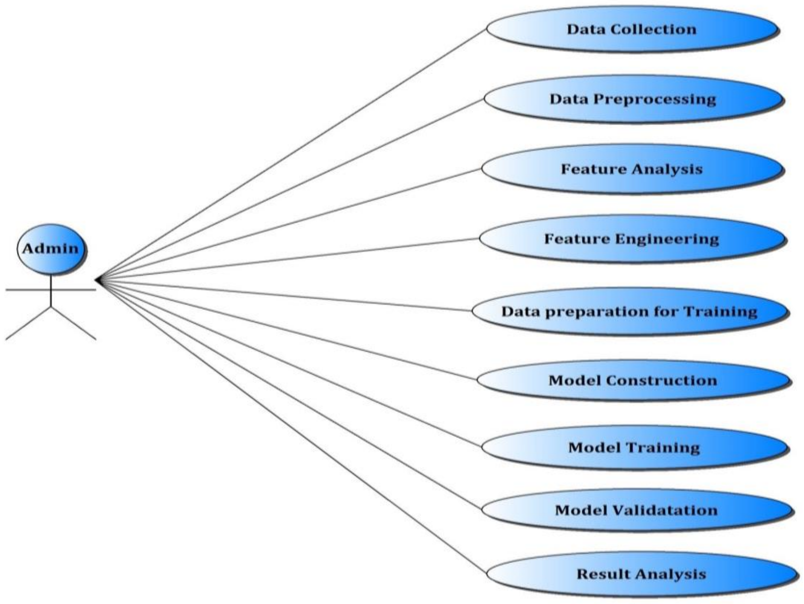


Fig 3.2 Flow Chart Diagram

3.3 Use case diagram:

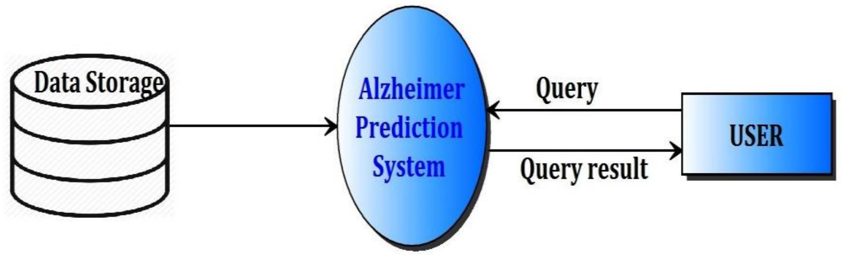
A usage case is a lot of circumstances that portraying a participation between a source and a goal. A use case diagram shows the relationship among performers and use cases. The two essential pieces of a use case dig, are use cases and performers. shows the use case graph.

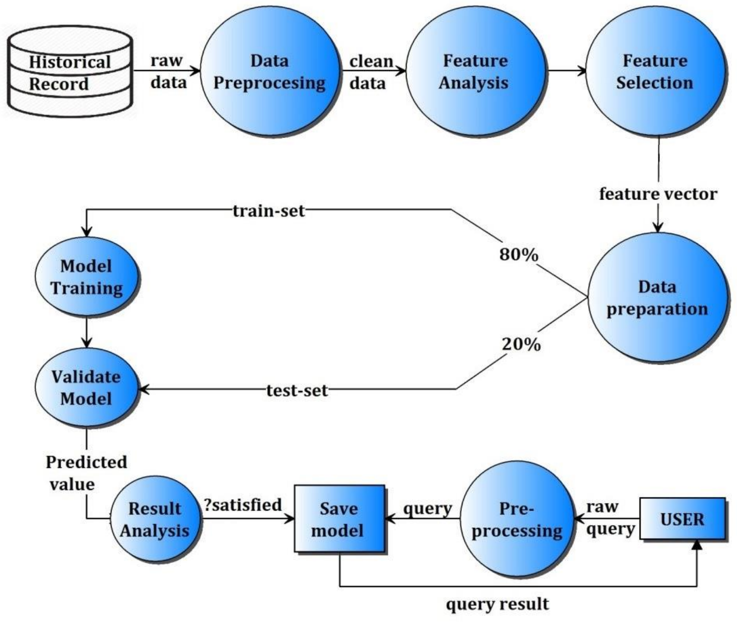


3.3.1 Use case diagram user

3.4 Data flow diagram

An information stream outline (DFD) is realistic portrayal of the "stream" of information through a data framework. An information stream outline can likewise be utilized for the representation of information handling (organized plan). It is normal practice for a creator to draw a setting level DFD first which shows the collaboration between the framework and outside elements.
The flat three-sided rectangle is representing data stores should both receive information for storing and provide it for further processing.





3.5 Modules Description

1. Data Collection

2. Image Acquisition and Pre-processing

3. Data Preparation & Model construction

4. Model training

5. Model testing and evaluation

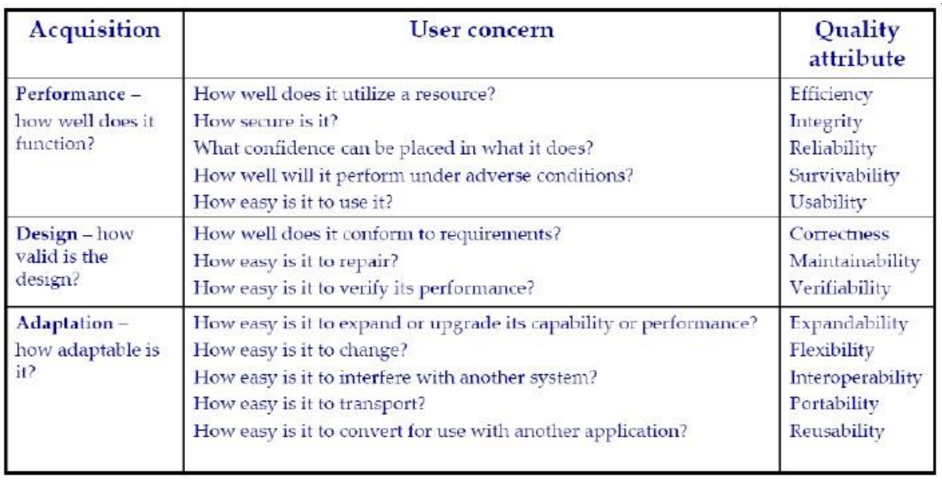
- **Data Collection**

- Alzheimer MRI Pre-processed Dataset (128 x 128)
- The Data is collected from several websites/hospitals/public repositories.
- The Dataset is consisting of Pre-processed MRI (Magnetic Resonance Imaging) Images.
- All the images are resized into 128 x 128 pixels.
- The Dataset has four classes of images.
- The Dataset is consisting of total 6400 MRI images.
  Class - 1: Mildly Demented (896 images)
  Class - 2: Moderately Demented (64 images)
  Class - 3: Non-Dementedly (3200 images)
  Class - 4: Very Mild Dementedly (2240 images)

- **Image Acquisition and Pre-processing**

In this module we will get the information from the web-based source. Further we will resize the picture for some time later. Picture resizing, or picture scaling, is a mathematical picture change which adjusts the picture size in light of a picture addition calculation. This picture scaling cycle can increment or decline the goal of an objective picture so the outright size of picture information is changed.

PCs can perform calculations on numbers and can't decipher pictures in the manner that we do. We need to some way or another believer the pictures to numbers for the PC to comprehend. The picture will be changed over completely to grayscale (scope of dim conceals from white to dark) the PC will dole out every pixel a worth in light of how dim it is. Every one of the numbers are placed into an exhibit and the PC does calculations on that cluster. We then feed the subsequent exhibit for following stage.



Acquisition	User concern	Quality attribute
Performance – how well does it function?	How well does it utilize a resource? How secure is it? What confidence can be placed in what it does? How well will it perform under adverse conditions? How easy is it to use it?	Efficiency Integrity Reliability Survivability Usability
Design – how valid is the design?	How well does it conform to requirements? How easy is it to repair? How easy is it to verify its performance?	Correctness Maintainability Verifiability
Adaptation – how adaptable is it?	How easy is it to expand or upgrade its capability or performance? How easy is it to change? How easy is it to interfere with another system? How easy is it to transport? How easy is it to convert for use with another application?	Expandability Flexibility Interoperability Portability Reusability

- **Data Preparation and Model construction**

Various on different occasions, people initially split their dataset into 2 — Train and Test. After this, they keep to the side the Test set, and randomly pick X% of their Train dataset to be the veritable Train set and the overabundance (100-X)% to be the Validation set, where X is a fixed number(say 80%), the model is then iteratively arranged and supported on these different sets. In this way, we will follow a comparable system to design data for getting ready and testing stage.

We are building our model by using Convolutional cerebrum association. Convolutional cerebrum associations (CNN) are a remarkable plan of phony mind associations, proposed by Yann LeCun in 1988. CNN uses a couple of features of the visual cortex. Now that we're done pre-dealing with, we can start completing our cerebrum association. We will have 3 convolution layers with 2 x 2 max-pooling.

Max-pooling: A method used to diminish the parts of an image by taking the best pixel worth of an organization. This in like manner decreases over fitting and makes the model more traditional. Starting then and into the foreseeable future, we add 2 totally related layers. Since the commitment of totally related layers should be two layered, and the consequence of convolution layer is four layered, we need an evening out layer between them. Toward the completion of the totally related layers is a SoftMax layer.

Not with standing, since there was close to 100% precision, there ought to be a method for supporting the exactness to no less than 90%. The justification for having low precision might be the issue that the state of the given information input is 128x128 as opposed to 224x224, which is the information that is normal for EfficientNetB0. In this way, in beneath, I will resize the picture and have a go at running CNN again with the changed size.

- **CNN model for Alzheimer Classifier (EfficientNetB0)**

Rather than using the typical and basic models, here, I use a bit more state-of-the-art model EfficientNet. Depending on the input data's shape, EfficientNet changes a little bit. However, for the smallest shape (usual input is 224x224) EfficientNetB0 is used.

- **Model training**

After model development it is the ideal opportunity for model preparation. We had the option to assemble a counterfeit convolutional brain network that can perceive pictures. Part the dataset into train and test dataset. At long last, we will construct and prepare the model utilizing preparing dataset.

- **Model testing and evaluation**

At the point when the model has been arranged finishing model, testing is possible. During this stage a test set of data is stacked. This instructive assortment has never been seen by the model and thus its real accuracy will be checked. Finally, the saved model can be used actually. The name of this stage is model appraisal. This suggests that the model can be used to survey new data.

CHAPTER-4

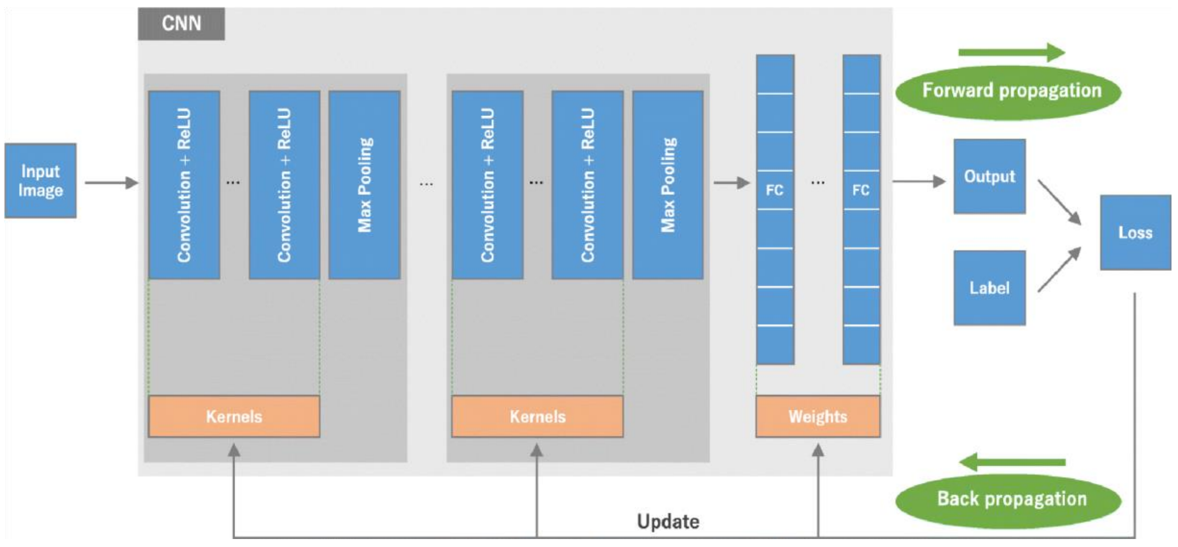
IMPLEMENTATION TECHNOLOGIES

Python Introduction:

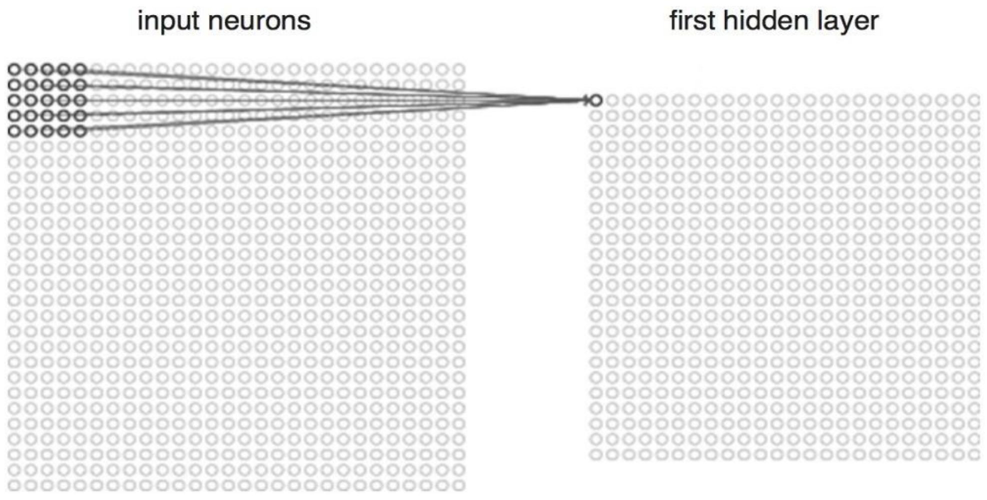
Python is a simple to learn, strong programming language. It has proficient significant level information structures and a straightforward however viable way to deal with object-situated programming. Python's exquisite grammar and dynamic composing, along with its deciphered nature, make it an optimal language for prearranging and fast application improvement in numerous areas on most stages.

Convolutional Neural Network

CNN Convolution is the essential layer to eliminate features from a data picture. Convolution protects the association between pixels by learning picture features using little squares of data. A mathematical action takes two wellsprings of data like picture lattice and a channel or touch. The principal benefit of CNN contrasted with its ancestors is that it consequently identifies the significant elements with no human management. CNN is likewise computationally productive. It utilizes exceptional convolution and pooling activities and performs boundary sharing. This empowers CNN models to run on any gadget, making them all around appealing. There is an info picture that we're working with. We play out a series convolution + pooling tasks, trailed by various completely associated layers. Assuming that we are performing multiclass grouping, the result is SoftMax.

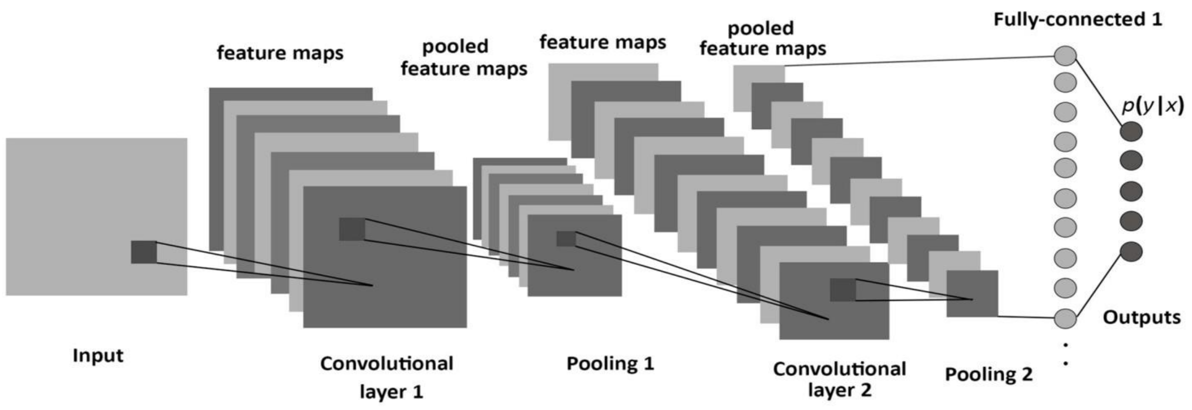


The Convolution layer is reliably the first. The image (framework with Instead of the image, the PC sees different pixels. For example, accepting picture size is 300 x 300. For this present circumstance, the size of the show will be 300x300x3. Where 300 is width, next 300 is level and 3 is RGB channel values. The PC is given out a value from 0 to 255 to all of these numbers.

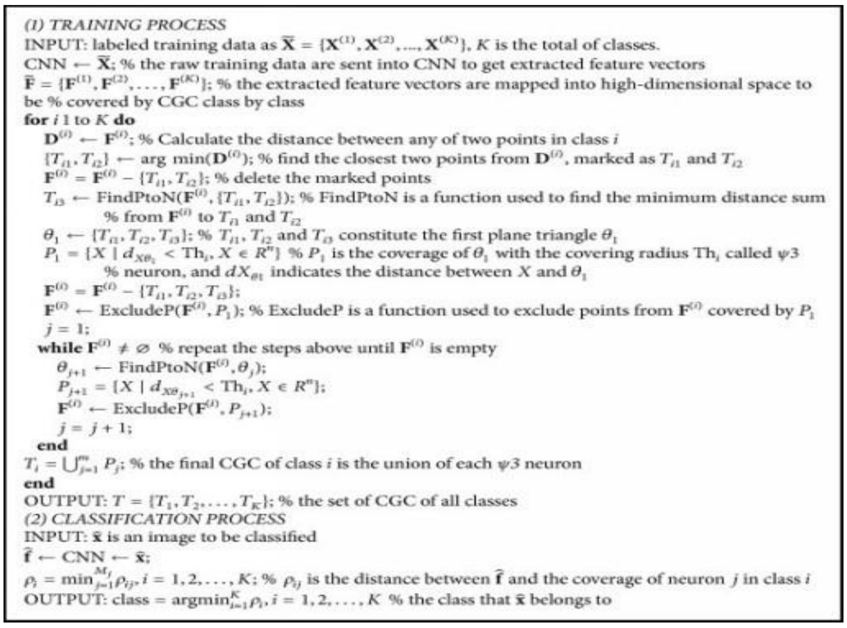


This activity, according to a human point of view, is closely resembling recognizing limits and basic varieties on the picture. However, to perceive the properties of a more elevated level, for example, the storage compartment or huge ears the entire organization is required.

The pooling layer follows the nonlinear layer. It works with width and level of the image and plays out a down inspecting methodology on them. In this way, the image volume is reduced. This means that if a couple of components (concerning model cut off points) have proactively been separated in the past convolution movement, then, a quick and dirty picture isn't for the most part expected for extra taking care of, and it is compacted to less unmistakable pictures.



After fulfilment of series of convolutional, nonlinear and pooling layers, it is essential to join a totally related layer. This layer takes the outcome information from convolutional networks. Joining a totally related layer to the farthest furthest reaches of the association achieves a N layered vector, where N is the number of classes from which the model picks the best class.



```
(1) TRAINING PROCESS
INPUT: labeled training data as $\overline{\mathbf{X}} = \{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, ..., \mathbf{X}^{(K)}\}$, $K$ is the total of classes.
CNN $\leftarrow \overline{\mathbf{X}}$; % the raw training data are sent into CNN to get extracted feature vectors
$\overline{\mathbf{F}} = \{\mathbf{F}^{(1)}, \mathbf{F}^{(2)}, \ldots, \mathbf{F}^{(K)}\}$; % the extracted feature vectors are mapped into high-dimensional space to
be % covered by CGC class by class
for $i$ 1 to $K$ do
    $\mathbf{D}^{(i)} \leftarrow \mathbf{F}^{(i)}$; % Calculate the distance between any of two points in class $i$
    $\{T_{i1}, T_{i2}\} \leftarrow \arg\min(\mathbf{D}^{(i)})$; % find the closest two points from $\mathbf{D}^{(i)}$, marked as $T_{i1}$ and $T_{i2}$
    $\mathbf{F}^{(i)} = \mathbf{F}^{(i)} - \{T_{i1}, T_{i2}\}$; % delete the marked points
    $T_{i3} \leftarrow$ FindPtoN($\mathbf{F}^{(i)}, \{T_{i1}, T_{i2}\}$); % FindPtoN is a function used to find the minimum distance sum
            % from $\mathbf{F}^{(i)}$ to $T_{i1}$ and $T_{i2}$
    $\theta_1 \leftarrow \{T_{i1}, T_{i2}, T_{i3}\}$; % $T_{i1}$, $T_{i2}$ and $T_{i3}$ constitute the first plane triangle $\theta_1$
    $P_1 = \{X \mid d_{X\theta_1} < \mathrm{Th}_i, X \in R^n\}$ % $P_1$ is the coverage of $\theta_1$ with the covering radius $\mathrm{Th}_i$ called $\psi 3$
            % neuron, and $dX_{\theta 1}$ indicates the distance between $X$ and $\theta_1$
    $\mathbf{F}^{(i)} = \mathbf{F}^{(i)} - \{T_{i1}, T_{i2}, T_{i3}\}$;
    $\mathbf{F}^{(i)} \leftarrow$ ExcludeP($\mathbf{F}^{(i)}, P_1$); % ExcludeP is a function used to exclude points from $\mathbf{F}^{(i)}$ covered by $P_1$
    $j = 1$;
   while $\mathbf{F}^{(i)} \neq \varnothing$ % repeat the steps above until $\mathbf{F}^{(i)}$ is empty
      $\theta_{j+1} \leftarrow$ FindPtoN($\mathbf{F}^{(i)}, \theta_j$);
      $P_{j+1} = \{X \mid d_{X\theta_{j+1}} < \mathrm{Th}_i, X \in R^n\}$;
      $\mathbf{F}^{(i)} \leftarrow$ ExcludeP($\mathbf{F}^{(i)}, P_{j+1}$);
      $j = j + 1$;
   end
$T_i = \bigcup_{j=1}^{m} P_j$; % the final CGC of class $i$ is the union of each $\psi 3$ neuron
end
OUTPUT: $T = \{T_1, T_2, \ldots, T_K\}$; % the set of CGC of all classes
(2) CLASSIFICATION PROCESS
INPUT: $\hat{\mathbf{x}}$ is an image to be classified
$\hat{\mathbf{f}} \leftarrow$ CNN $\leftarrow \hat{\mathbf{x}}$;
$\rho_i = \min_{j=1}^{M_i} \rho_{ij}, i = 1, 2, \ldots, K$; % $\rho_{ij}$ is the distance between $\hat{\mathbf{f}}$ and the coverage of neuron $j$ in class $i$
OUTPUT: class = $\mathrm{argmin}_{i=1}^{K} \rho_i, i = 1, 2, \ldots, K$ % the class that $\hat{\mathbf{x}}$ belongs to
```

Artificial intelligence

Computerized reasoning incorporates an exceptionally wide extension. we might consider something like Dijkstra's most brief way calculation as Artificial Intelligence. Be that as it may, two classifications of AI are regularly stirred up: Machine Learning and Deep Learning.

Machine learning

AI is a technique for factual realizing where each example in a dataset is depicted by a bunch of elements or characteristics. The terms appear to be to some degree tradable, be that as it may, with Deep Learning strategy the calculation builds portrayals of the information consequently. Conversely, information portrayals are hard-coded as a bunch of elements in machine ML calculations, requiring further cycles like component choice and extraction, (like PCA, LDA,).

SDLC Project Methodology

It will cover the subtleties clarification of strategy that is being utilized to make this venture total and functioning admirably. Numerous techniques or discoveries from this field basically produced into diary for others to take benefits and work on as impending investigations. The strategy is use to accomplish the goal of the task that will achieve an ideal outcome. To assess this venture, the approach in view of System Development Life Cycle (SDLC)

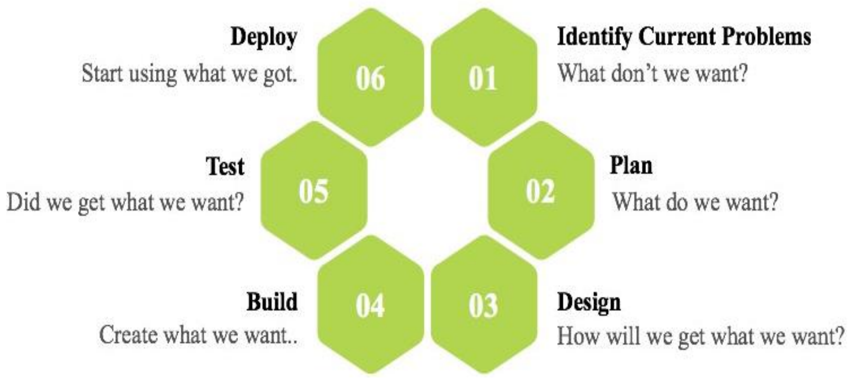


Fig 1: Software Development Life Cycle

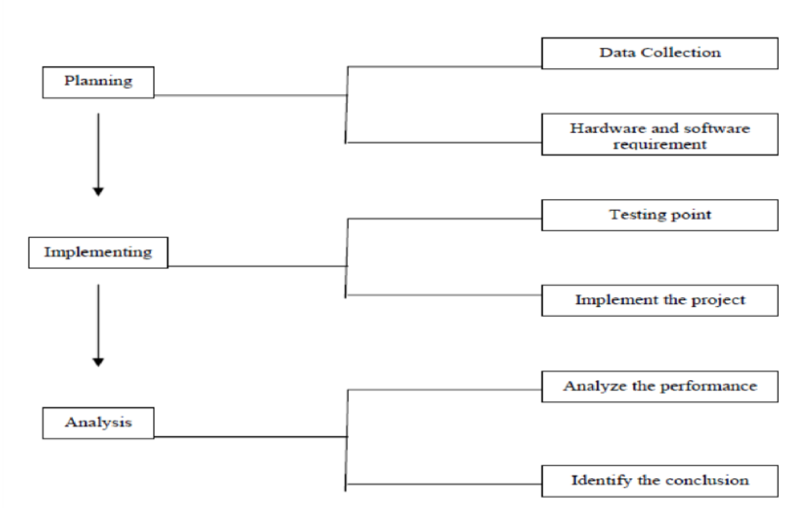


Fig 2: Steps of Methodology

Implementing

In this work, a business canny model has been created, to find human exercises, in view of a particular business structure manage human exercises utilizing a reasonable AI procedure. We are utilizing Convolutional Neural Network (CNN) to fabricate our model.

Hardware Requirements:

System	:	above i5
Hard Disk	:	500 GB.
Ram	:	12/16 GB

Requirements:

Operating system	:	Windows 8 / 10
Coding Language	:	Python
Software	:	Anaconda
IDE	:	Jupyter Notebook

CHAPTER -5

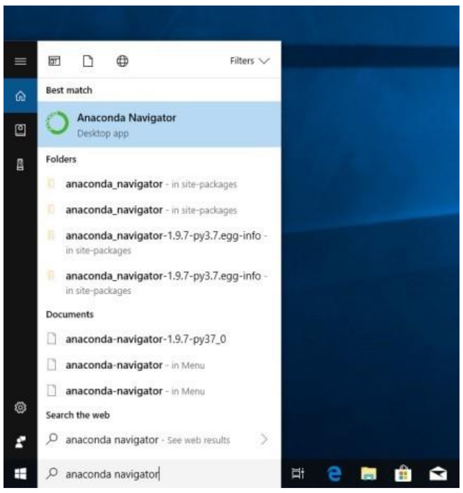
TESTING AND IMPLEMENTATION

Anaconda:

Use Anaconda Navigator to ship off an application. Then, make and run a fundamental Python program with Spyder and Jupyter Notebook.

- Open Navigator Choose the guidelines for your working framework. Windows

From the very start menu, click the Anaconda Navigator workspace application.



Filters

Best match

Anaconda Navigator
Desktop app

Folders

anaconda_navigator - in site-packages
anaconda_navigator - in site-packages
anaconda_navigator-1.9.7-py3.7.egg-info - in site-packages
anaconda_navigator-1.9.7-py3.7.egg-info - in site-packages

Documents

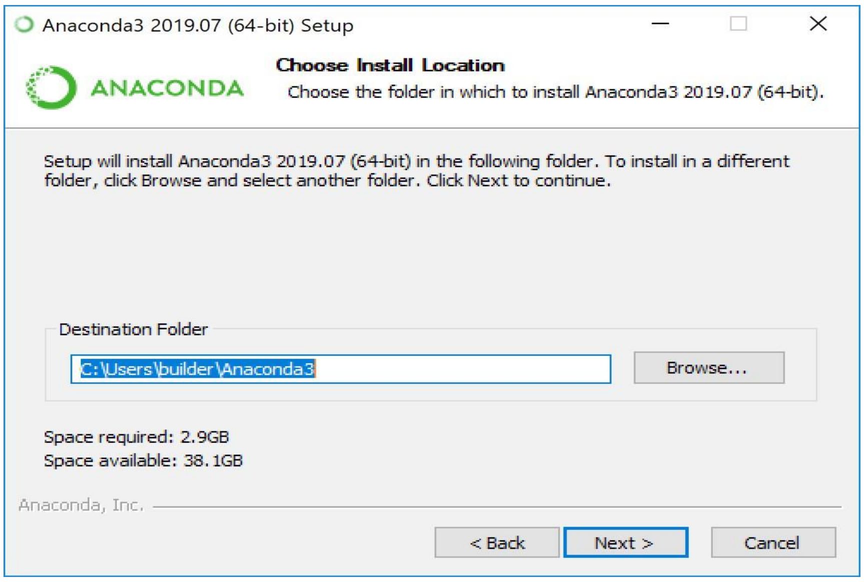
anaconda-navigator-1.9.7-py37_0
anaconda-navigator - in Menu
anaconda-navigator - in Menu

Search the web

anaconda navigator - See web results

anaconda navigator

Introducing on Windows



Anaconda3 2019.07 (64-bit) Setup

ANACONDA

Choose Install Location

Choose the folder in which to install Anaconda3 2019.07 (64-bit).

Setup will install Anaconda3 2019.07 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

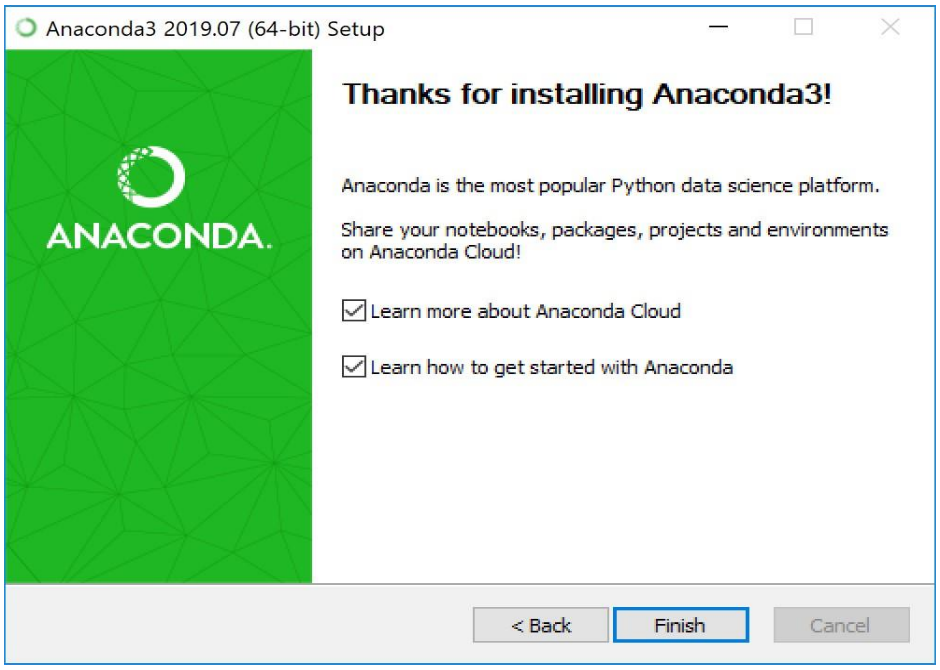
C:\Users\builder\Anaconda3

Browse...

Space required: 2.9GB
Space available: 38.1GB

Anaconda, Inc.

< Back | Next > | Cancel


Anaconda3 2019.07 (64-bit) Setup

Thanks for installing Anaconda3!

Anaconda is the most popular Python data science platform.

Share your notebooks, packages, projects and environments on Anaconda Cloud!

☑ Learn more about Anaconda Cloud

☑ Learn how to get started with Anaconda

5.1 Software testing introduction

Programming testing is a cycle used to assist with recognizing the rightness, fulfilment and nature of created program. Programming testing is the interaction used to quantify the nature of created programming. Testing is the most common way of executing a program with the purpose of tracking down blunders. Programming testing is frequently alluded to as check and approval.

5.2 Explanation for SDLC & STLC

SDLC: The product improvement life cycle is a reasonable model utilized in project the board that depicts the stages engaged with a data framework improvement project, from an underlying practicality concentrate on through support of the finished application.

5.3 PHASES OF SOFTWARE DEVELOPMENT

- Requirement Analysis
- Software design
- Development or Coding
- Testing
- Maintenance

5.3.1 Requirement analysis

The necessities of an ideal programming item are separated. Based the business situation the SRS (Software Requirement Specification) record is ready in this stage.

5.3.2 Design

Plans are spread out concerning the actual development, equipment, working frameworks, programming, correspondences, and security issues for the product.
There are 2 phases in plan,
HLD - High Level Design
LLD - Low Level Design
HLD - gives the plan of the item thing to be made and is done by organizers and senior specialists.
LLD - done by senior architects. It depicts how each and every component in the thing should work and how each part should work. Here, just the plan will be there and not the code.

5.3.3 Testing

Testing is assessing the product to check for the client prerequisites. Here the product is assessed with purpose of tracking down surrenders.

5.3.4 Maintenance

When the new framework is ready for some time, it ought to be comprehensively assessed. Support should be kept up thoroughly consistently. Clients of the framework ought to be kept upto-date concerning the most recent changes and strategies

5.4 SDLC Models

5.4.1 Water fall model

It will execute individually of the SDLC interaction. The plan Starts subsequent to finishing the prerequisites investigation coding starts after plan. It is a standard model It is a progressive arrangement process, oftentimes used in SDLC, in which the headway is seen as streaming reliably downwards (like an outpouring), through the different stages.

5.4.2 Spiral model

This mechanism is updating the application version by version. All the SDLC process will update version by version**.**

5.5. STLC (**Software Testing Life Cycle):** STLC is important for SDLC Test Plan.

- Test Development
- Test Execution
- Analyse Results
- Defect Tracking
- Summaries Report

5.5.1. TEST PLAN

It is a report which depicts the testing environment, reason, scope, targets, test strategy, plans, accomplishments, testing instrument, occupations and commitments, bets, planning, staffing and who will test the application, what kind of tests should be performed and the way that it will follow the flaws.

5.6. TYPES OF TESTING:

White Box Testing.
Black Box Testing.
Grey box testing.

5.6.1WHITE BOX TESTING

White box testing as the name proposes gives the inside perspective on the product. This sort of testing is otherwise called primary testing or glass box testing too, as the interest lies in what lies inside the container.

5.6.2BLACK BOX TESTING

It's moreover called as friendly testing. It revolves around the down to earth necessities of the item. Testing either common sense or non-utilitarian without reference to the internal development of the part or structure is called black box testing.

5.6.3 GREY BOX TESTING

Dark box testing is the blend n of black box and white box testing. Expectation of this testing is to figure out abandons connected with awful plan or terrible execution of the framework.

5.7 LEVEL OF TESTING USED IN PROJECT

5.7.1 Alpha testing

Alpha testing is last trying before the product is delivered to the overall population. This testing is led at the designer site and in a controlled climate toward the end client of the product.

5.7.2 Beta testing

The beta test is aimed no less than one client objections around the end client of the item. The beta test is driven no less than one client regions close to the end client of the item. 5.8-unit testing cases. Instatement testing is the essential level of dynamic testing and is first the commitment of creators and a short time later that of the test engineers. Unit testing is performed after the typical exploratory results are met or differentiates are sensible/recognize

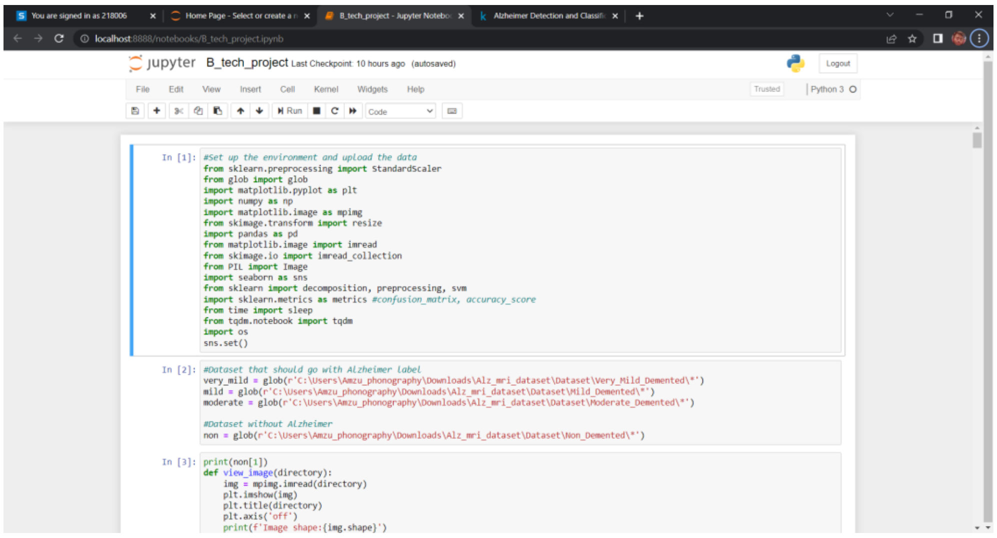
CHAPTER-6

RESULT AND ANALYSIS

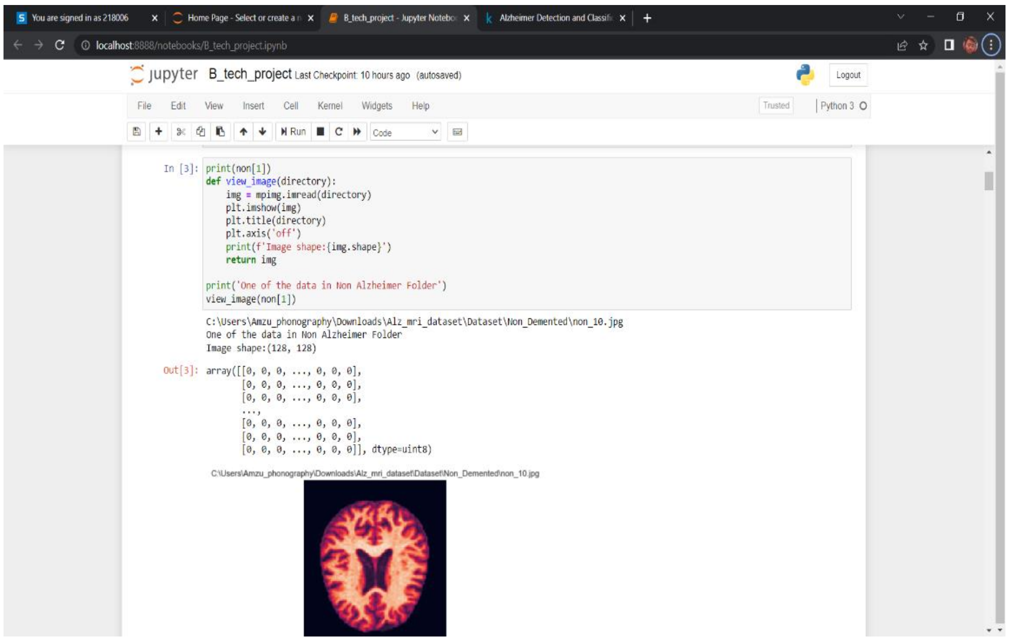
An outcome is the last result of activities or occasions communicated subjectively or quantitatively. Execution examination is a functional investigation, is a bunch of fundamental quantitative connection between the presentation amounts.

6.1. Screen shots

Add your screen shot of your code here.



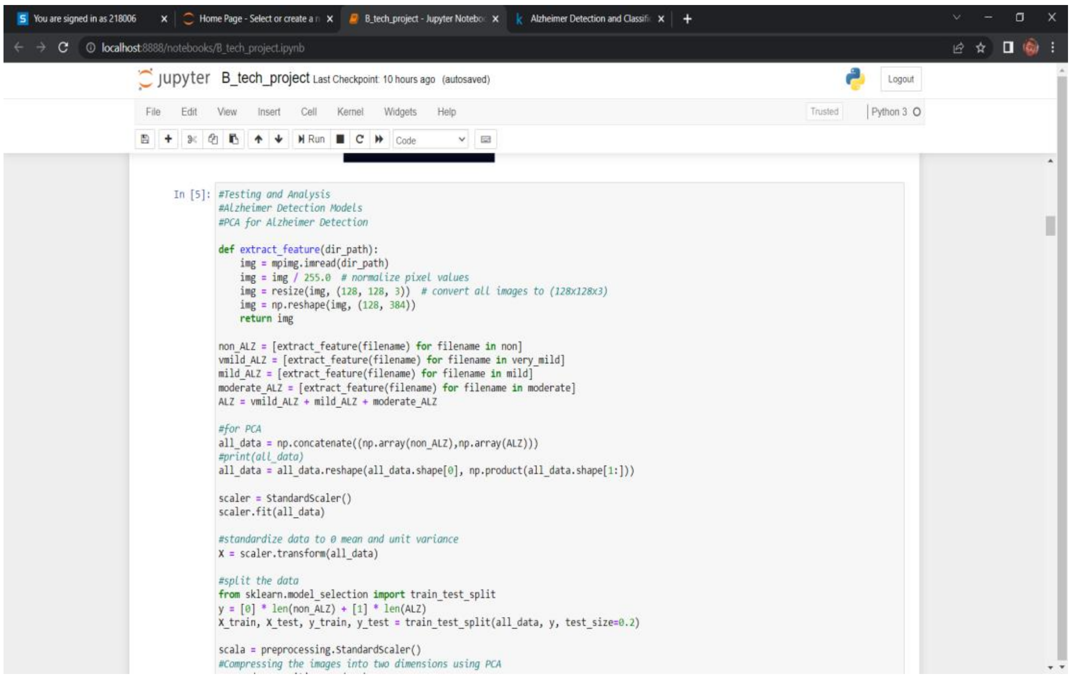
```
In [1]: #Set up the environment and upload the data
        from sklearn.preprocessing import StandardScaler
        from glob import glob
        import matplotlib.pyplot as plt
        import numpy as np
        import matplotlib.image as mpimg
        from skimage.transform import resize
        import pandas as pd
        from matplotlib.image import imread
        from skimage.io import imread_collection
        from PIL import Image
        import seaborn as sns
        from sklearn import decomposition, preprocessing, svm
        import sklearn.metrics as metrics #confusion_matrix, accuracy_score
        from time import sleep
        from tqdm.notebook import tqdm
        import os
        sns.set()

In [2]: #Dataset that should go with Alzheimer label
        very_mild = glob(r'C:\Users\Amzu_phonography\Downloads\Alz_mri_dataset\Dataset\Very_Mild_Demented\*')
        mild = glob(r'C:\Users\Amzu_phonography\Downloads\Alz_mri_dataset\Dataset\Mild_Demented\*')
        moderate = glob(r'C:\Users\Amzu_phonography\Downloads\Alz_mri_dataset\Dataset\Moderate_Demented\*')

        #Dataset without Alzheimer
        non = glob(r'C:\Users\Amzu_phonography\Downloads\Alz_mri_dataset\Dataset\Non_Demented\*')

In [3]: print(non[1])
        def view_image(directory):
            img = mpimg.imread(directory)
            plt.imshow(img)
            plt.title(directory)
            plt.axis('off')
            print(f'Image shape:{img.shape}')
```



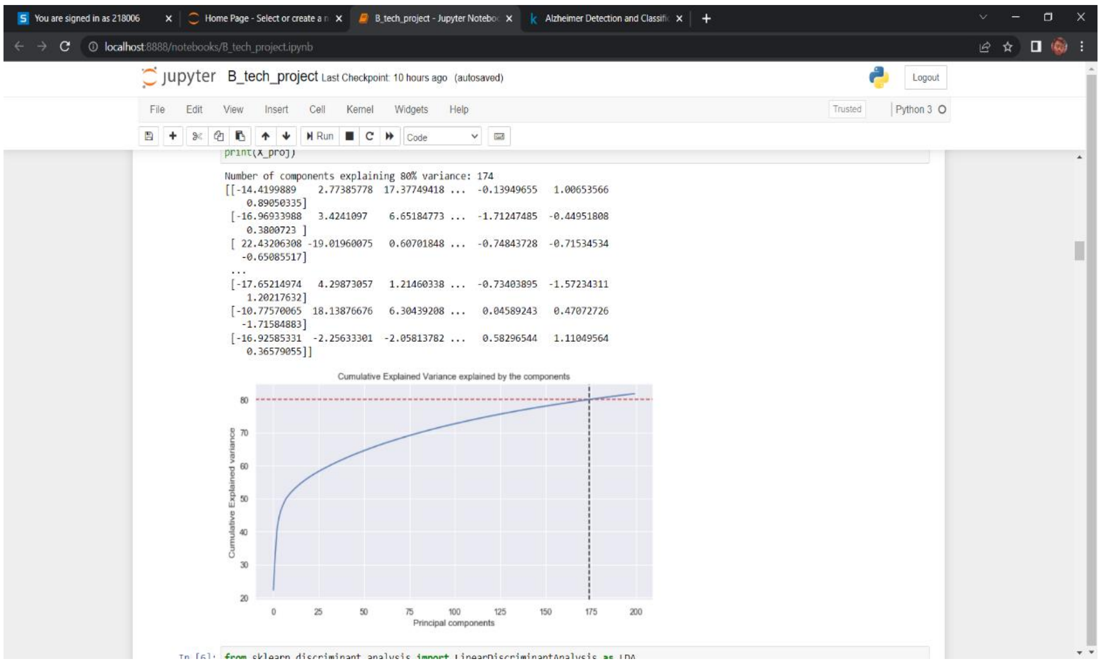
```
In [3]: print(non[1])
        def view_image(directory):
            img = mpimg.imread(directory)
            plt.imshow(img)
            plt.title(directory)
            plt.axis('off')
            print(f'Image shape:{img.shape}')
            return img

        print('One of the data in Non Alzheimer Folder')
        view_image(non[1])

        C:\Users\Amzu_phonography\Downloads\Alz_mri_dataset\Dataset\Non_Demented\non_10.jpg
        One of the data in Non Alzheimer Folder
        Image shape:(128, 128)

Out[3]: array([[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)

        C:\Users\Amzu_phonography\Downloads\Alz_mri_dataset\Dataset\Non_Demented\non_10.jpg
```

Alzheimer Detection Models

PCA for Alzheimer Detection



```
In [5]: #Testing and Analysis
        #Alzheimer Detection Models
        #PCA for Alzheimer Detection

        def extract_feature(dir_path):
            img = mpimg.imread(dir_path)
            img = img / 255.0  # normalize pixel values
            img = resize(img, (128, 128, 3))  # convert all images to (128x128x3)
            img = np.reshape(img, (128, 384))
            return img

        non_ALZ = [extract_feature(filename) for filename in non]
        vmild_ALZ = [extract_feature(filename) for filename in very_mild]
        mild_ALZ = [extract_feature(filename) for filename in mild]
        moderate_ALZ = [extract_feature(filename) for filename in moderate]
        ALZ = vmild_ALZ + mild_ALZ + moderate_ALZ

        #for PCA
        all_data = np.concatenate((np.array(non_ALZ),np.array(ALZ)))
        #print(all_data)
        all_data = all_data.reshape(all_data.shape[0], np.product(all_data.shape[1:]))

        scaler = StandardScaler()
        scaler.fit(all_data)

        #standardize data to 0 mean and unit variance
        X = scaler.transform(all_data)

        #split the data
        from sklearn.model_selection import train_test_split
        y = [0] * len(non_ALZ) + [1] * len(ALZ)
        X_train, X_test, y_train, y_test = train_test_split(all_data, y, test_size=0.2)

        scala = preprocessing.StandardScaler()
        #Compressing the images into two dimensions using PCA
```

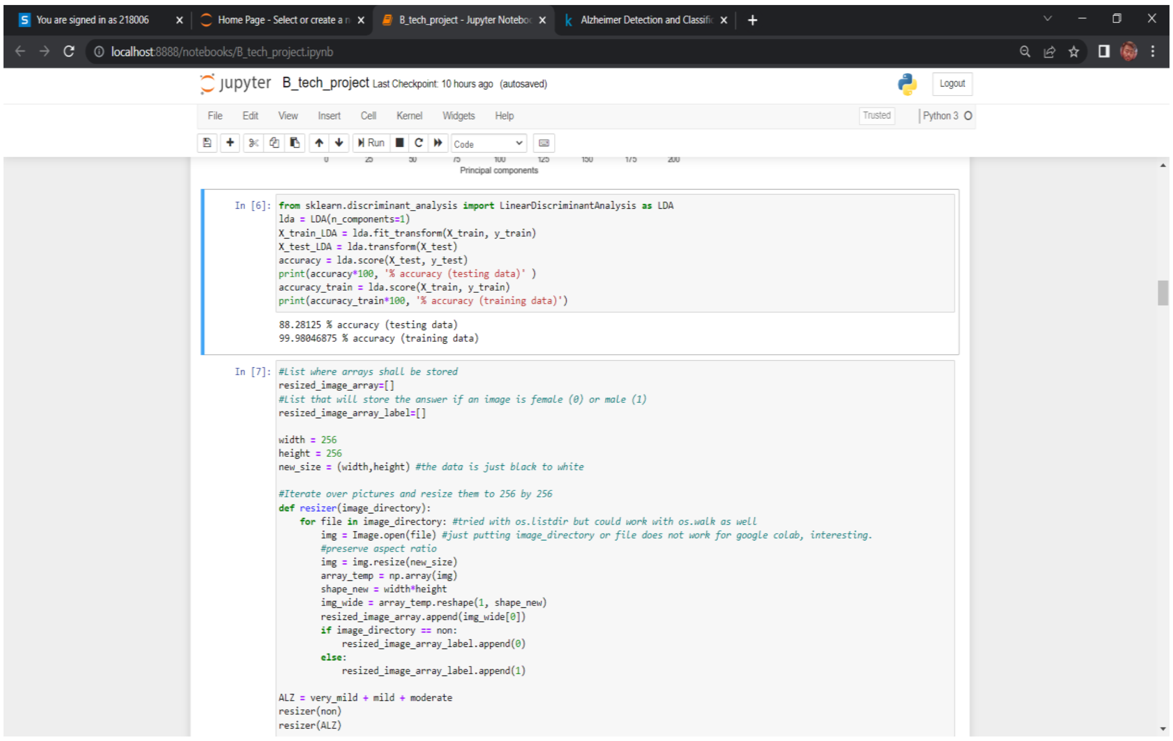


```
print(X_proj)

Number of components explaining 80% variance: 174
[[-14.4199889    2.77385778  17.37749418 ...  -0.13949655   1.00653566
    0.89050335]
 [-16.96933988   3.4241097    6.65184773 ...  -1.71247485  -0.44951808
    0.3800723 ]
 [ 22.43206308 -19.01960075   0.60701848 ...  -0.74843728  -0.71534534
   -0.65085517]
 ...
 [-17.65214974   4.29873057   1.21460338 ...  -0.73403895  -1.57234311
    1.20217632]
 [-10.77570065  18.13876676   6.30439208 ...   0.04589243   0.47072726
   -1.71584883]
 [-16.92585331  -2.25633301  -2.05813782 ...   0.58296544   1.11049564
    0.36579055]]
```

Cumulative Explained Variance explained by the components
Cumulative Explained variance
Principal components
80
70
60
50
40
30
20
0
25
50
75
100
125
150
175
200

```
In [6]: from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
```

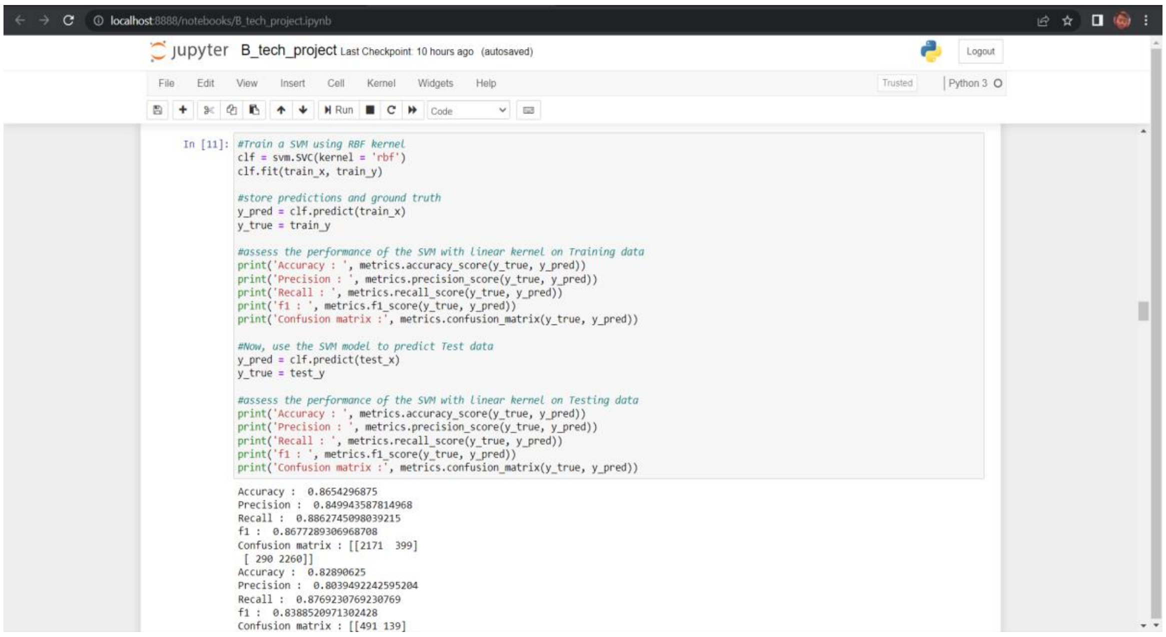
From looking at the above, PCA will differentiate the dataset into two, a brain suffering from Alzheimer or not. However, to have at least 80% accuracy of differentiation, it requires 174 principal components. This means that to differentiate the data with 80% accuracy, we will need to make a model that has 174 different axes in the data.



```
In [6]: from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
        lda = LDA(n_components=1)
        X_train_LDA = lda.fit_transform(X_train, y_train)
        X_test_LDA = lda.transform(X_test)
        accuracy = lda.score(X_test, y_test)
        print(accuracy*100, '% accuracy (testing data)' )
        accuracy_train = lda.score(X_train, y_train)
        print(accuracy_train*100, '% accuracy (training data)')

        88.28125 % accuracy (testing data)
        99.98046875 % accuracy (training data)

In [7]: #List where arrays shall be stored
        resized_image_array=[]
        #List that will store the answer if an image is female (0) or male (1)
        resized_image_array_label=[]

        width = 256
        height = 256
        new_size = (width,height) #the data is just black to white

        #Iterate over pictures and resize them to 256 by 256
        def resizer(image_directory):
            for file in image_directory: #tried with os.listdir but could work with os.walk as well
                img = Image.open(file) #just putting image_directory or file does not work for google colab, interesting.
                #preserve aspect ratio
                img = img.resize(new_size)
                array_temp = np.array(img)
                shape_new = width*height
                img_wide = array_temp.reshape(1, shape_new)
                resized_image_array.append(img_wide[0])
                if image_directory == non:
                    resized_image_array_label.append(0)
                else:
                    resized_image_array_label.append(1)

        ALZ = very_mild + mild + moderate
        resizer(non)
        resizer(ALZ)
```

First time when I saw the performance of the LDA (Linear Discriminant Analysis) model, I was surprised to see the performance of the model on the training data is so high. as well as the performance on the testing data is also significant. This gave me a lot of confidence about the data as it seems to be very well taken with MRI.

SVM for Alzheimer Detection



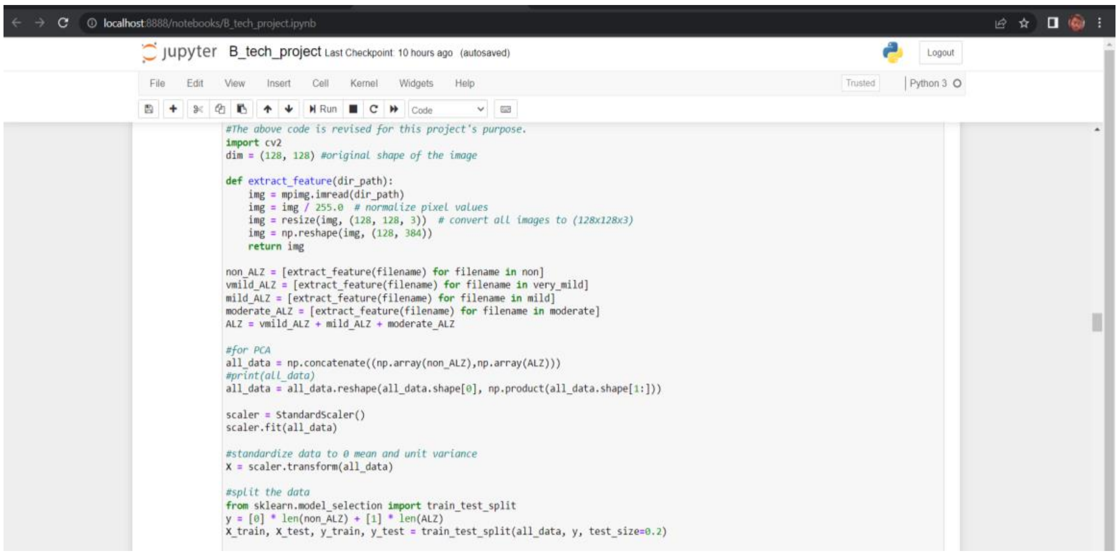
```
In [11]: #Train a SVM using RBF kernel
         clf = svm.SVC(kernel = 'rbf')
         clf.fit(train_x, train_y)

         #store predictions and ground truth
         y_pred = clf.predict(train_x)
         y_true = train_y

         #assess the performance of the SVM with linear kernel on Training data
         print('Accuracy : ', metrics.accuracy_score(y_true, y_pred))
         print('Precision : ', metrics.precision_score(y_true, y_pred))
         print('Recall : ', metrics.recall_score(y_true, y_pred))
         print('f1 : ', metrics.f1_score(y_true, y_pred))
         print('Confusion matrix :', metrics.confusion_matrix(y_true, y_pred))

         #Now, use the SVM model to predict Test data
         y_pred = clf.predict(test_x)
         y_true = test_y

         #assess the performance of the SVM with linear kernel on Testing data
         print('Accuracy : ', metrics.accuracy_score(y_true, y_pred))
         print('Precision : ', metrics.precision_score(y_true, y_pred))
         print('Recall : ', metrics.recall_score(y_true, y_pred))
         print('f1 : ', metrics.f1_score(y_true, y_pred))
         print('Confusion matrix :', metrics.confusion_matrix(y_true, y_pred))

         Accuracy :  0.8654296875
         Precision :  0.849943587814968
         Recall :  0.8862745098039215
         f1 :  0.8677289306968708
         Confusion matrix : [[2171  399]
          [ 290 2260]]
         Accuracy :  0.82890625
         Precision :  0.8039492242595204
         Recall :  0.8769230769230769
         f1 :  0.8388520971302428
         Confusion matrix : [[491 139]
```

Already from the SVM's linear kernel, the performance of the model is very promising. Considering that this model is not those of so-what-called 'State-of-the-Art', it still has a stellar performance.

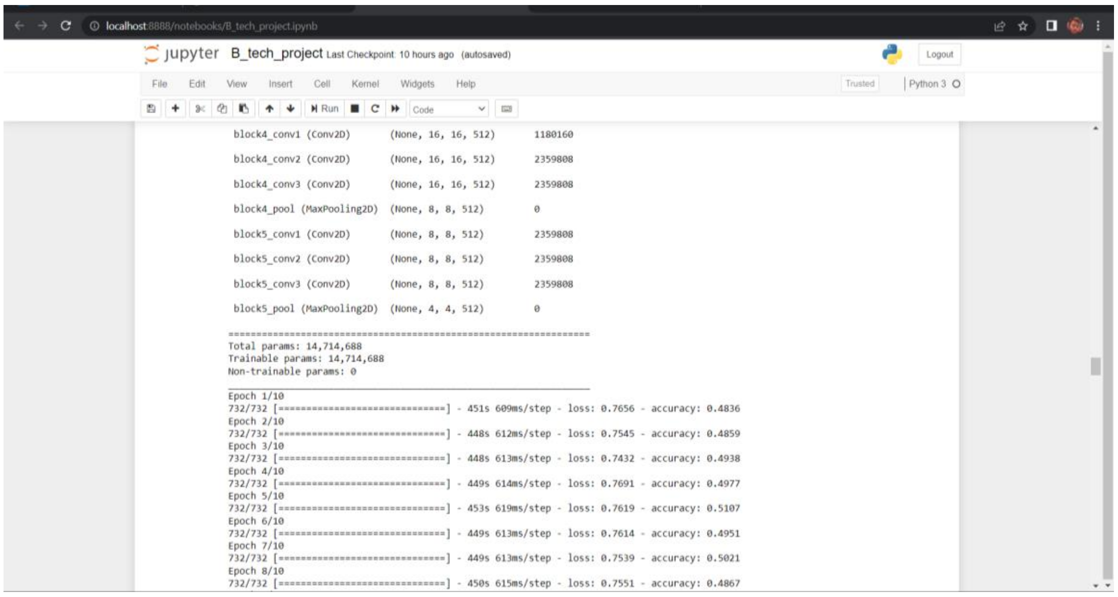
Surprisingly, as the kernel gets more complex, the overall performance does not necessarily rise. This may be the reason that 'having Alzheimer' or not is very easy to distinguish by looking at the thickness of the grey matter and the size of the ventricles. Which does not require some complex kernel tricks to be made.
However, at this point, it is just 'Alzheimer detector' not 'Alzheimer classifier' as the original data is categorical variable. So, when I start conducting the Alzheimer classifier, more complex kernel may be better rather than a simple linear kernel trick.

VGG16 (CNN) for Alzheimer Detection

VGG16 is known to take RGB colored data as the input for the model. However, the MRI images are in greyscale. So, I had to research on how to make the grayscale data to multi-channel before I feed it to VGG16. Reference for this greyscale data



```python
#The above code is revised for this project's purpose.
import cv2
dim = (128, 128) #original shape of the image

def extract_feature(dir_path):
    img = mpimg.imread(dir_path)
    img = img / 255.0  # normalize pixel values
    img = resize(img, (128, 128, 3))  # convert all images to (128x128x3)
    img = np.reshape(img, (128, 384))
    return img

non_ALZ = [extract_feature(filename) for filename in non]
vmild_ALZ = [extract_feature(filename) for filename in very_mild]
mild_ALZ = [extract_feature(filename) for filename in mild]
moderate_ALZ = [extract_feature(filename) for filename in moderate]
ALZ = vmild_ALZ + mild_ALZ + moderate_ALZ

#for PCA
all_data = np.concatenate((np.array(non_ALZ),np.array(ALZ)))
#print(all_data)
all_data = all_data.reshape(all_data.shape[0], np.product(all_data.shape[1:]))

scaler = StandardScaler()
scaler.fit(all_data)

#standardize data to 0 mean and unit variance
X = scaler.transform(all_data)

#split the data
from sklearn.model_selection import train_test_split
y = [0] * len(non_ALZ) + [1] * len(ALZ)
X_train, X_test, y_train, y_test = train_test_split(all_data, y, test_size=0.2)
```



```
 block4_conv1 (Conv2D)       (None, 16, 16, 512)       1180160

 block4_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 8, 8, 512)         0

 block5_conv1 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv2 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_conv3 (Conv2D)       (None, 8, 8, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
732/732 [==============================] - 451s 609ms/step - loss: 0.7656 - accuracy: 0.4836
Epoch 2/10
732/732 [==============================] - 448s 612ms/step - loss: 0.7545 - accuracy: 0.4859
Epoch 3/10
732/732 [==============================] - 448s 613ms/step - loss: 0.7432 - accuracy: 0.4938
Epoch 4/10
732/732 [==============================] - 449s 614ms/step - loss: 0.7691 - accuracy: 0.4977
Epoch 5/10
732/732 [==============================] - 453s 619ms/step - loss: 0.7619 - accuracy: 0.5107
Epoch 6/10
732/732 [==============================] - 449s 613ms/step - loss: 0.7614 - accuracy: 0.4951
Epoch 7/10
732/732 [==============================] - 449s 613ms/step - loss: 0.7539 - accuracy: 0.5021
Epoch 8/10
732/732 [==============================] - 450s 615ms/step - loss: 0.7551 - accuracy: 0.4867
```

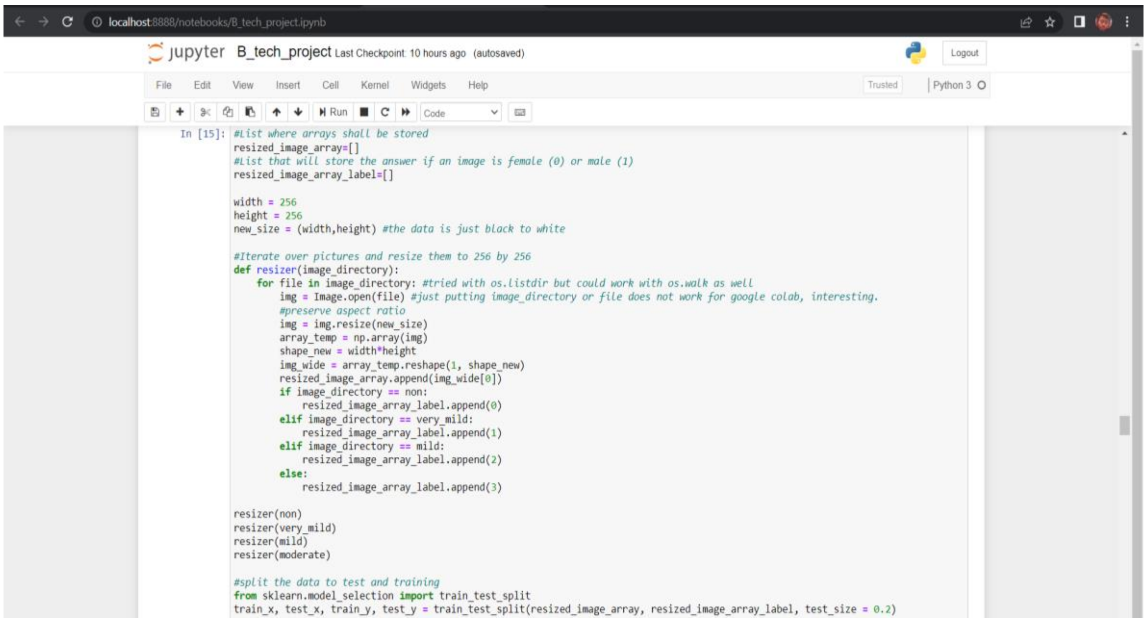
From the results, SVM with linear kernel has 98% for all performance measure on testing data. Other kernels that are more complex than this did not work well with SVM as the linear kernel. LDA on the other hand had 90% accuracy on testing data. PCA can have 80% accuracy by having 174 principle components (may change as it depends how training and

testing data are split). CNN model (VGG16) surprisingly had low performance considering that it is the most complicated model that I've used for the Alzheimer Detector.
This may be because that I've changed the greyscale to RBG (colored images) so that it fits the requirement for VGG16. (VGG16 is used for color images). Thus, making the data change might have affected the performance of VGG16. Accuracy of somewhere around 50% just means that it is not different from making random guess as there are only two choices for the AI model to decide. 'This image is an Alzheimer patient's brain' or not.
To see if this is the issue for CNN itself or from transforming the data, I will try different model in the Alzheimer Classifier. (EfficientNetB0)

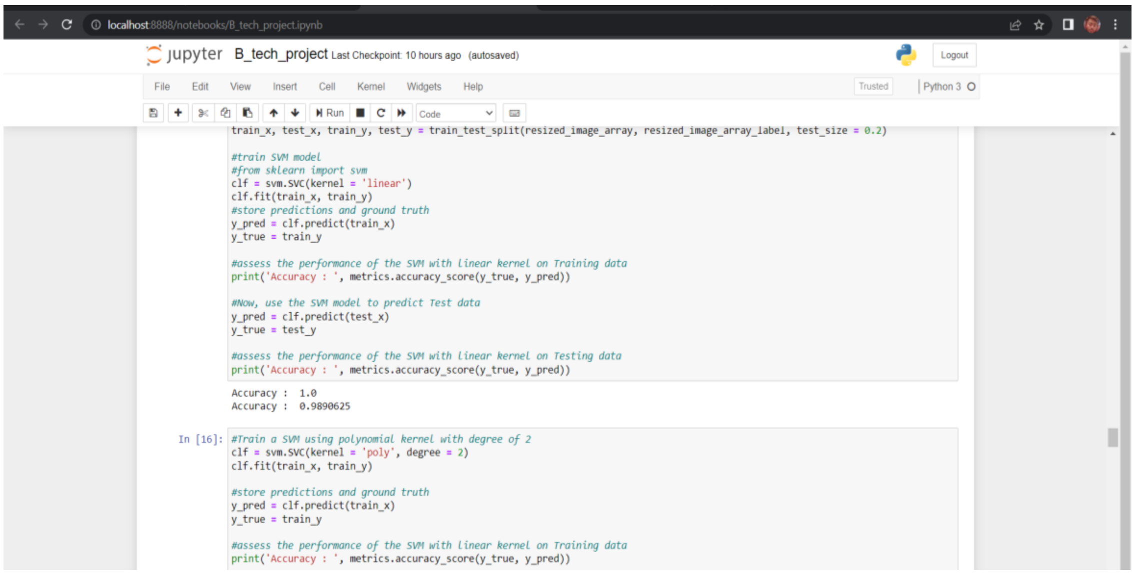
Alzheimer Classifier

Now is the time where we make SVM, LDA, and CNN models for Alzheimer classifier.

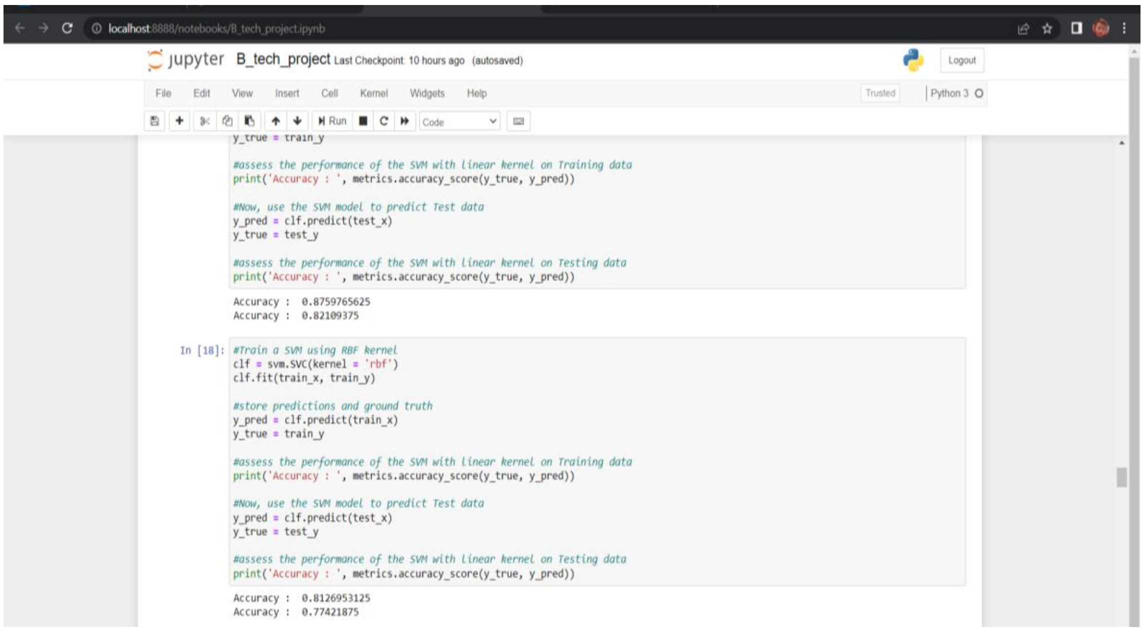
SVM for Alzheimer Classifier


localhost:8888/notebooks/B_tech_project.ipynb
jupyter B_tech_project Last Checkpoint: 10 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [15]: #List where arrays shall be stored
         resized_image_array=[]
         #List that will store the answer if an image is female (0) or male (1)
         resized_image_array_label=[]

         width = 256
         height = 256
         new_size = (width,height) #the data is just black to white

         #Iterate over pictures and resize them to 256 by 256
         def resizer(image_directory):
             for file in image_directory: #tried with os.listdir but could work with os.walk as well
                 img = Image.open(file) #just putting image_directory or file does not work for google colab, interesting.
                 #preserve aspect ratio
                 img = img.resize(new_size)
                 array_temp = np.array(img)
                 shape_new = width*height
                 img_wide = array_temp.reshape(1, shape_new)
                 resized_image_array.append(img_wide[0])
                 if image_directory == non:
                     resized_image_array_label.append(0)
                 elif image_directory == very_mild:
                     resized_image_array_label.append(1)
                 elif image_directory == mild:
                     resized_image_array_label.append(2)
                 else:
                     resized_image_array_label.append(3)

         resizer(non)
         resizer(very_mild)
         resizer(mild)
         resizer(moderate)

         #split the data to test and training
         from sklearn.model_selection import train_test_split
         train_x, test_x, train_y, test_y = train_test_split(resized_image_array, resized_image_array_label, test_size = 0.2)
```


localhost:8888/notebooks/B_tech_project.ipynb
jupyter B_tech_project Last Checkpoint: 10 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
         train_x, test_x, train_y, test_y = train_test_split(resized_image_array, resized_image_array_label, test_size = 0.2)

         #train SVM model
         #from sklearn import svm
         clf = svm.SVC(kernel = 'linear')
         clf.fit(train_x, train_y)
         #store predictions and ground truth
         y_pred = clf.predict(train_x)
         y_true = train_y

         #assess the performance of the SVM with linear kernel on Training data
         print('Accuracy : ', metrics.accuracy_score(y_true, y_pred))

         #Now, use the SVM model to predict Test data
         y_pred = clf.predict(test_x)
         y_true = test_y

         #assess the performance of the SVM with linear kernel on Testing data
         print('Accuracy : ', metrics.accuracy_score(y_true, y_pred))

         Accuracy :  1.0
         Accuracy :  0.9890625

In [16]: #Train a SVM using polynomial kernel with degree of 2
         clf = svm.SVC(kernel = 'poly', degree = 2)
         clf.fit(train_x, train_y)

         #store predictions and ground truth
         y_pred = clf.predict(train_x)
         y_true = train_y

         #assess the performance of the SVM with linear kernel on Training data
         print('Accuracy : ', metrics.accuracy_score(y_true, y_pred))
```



```
y_true = train_y

#assess the performance of the SVM with linear kernel on Training data
print('Accuracy : ', metrics.accuracy_score(y_true, y_pred))

#Now, use the SVM model to predict Test data
y_pred = clf.predict(test_x)
y_true = test_y

#assess the performance of the SVM with linear kernel on Testing data
print('Accuracy : ', metrics.accuracy_score(y_true, y_pred))
```

```
Accuracy :  0.8759765625
Accuracy :  0.82109375
```

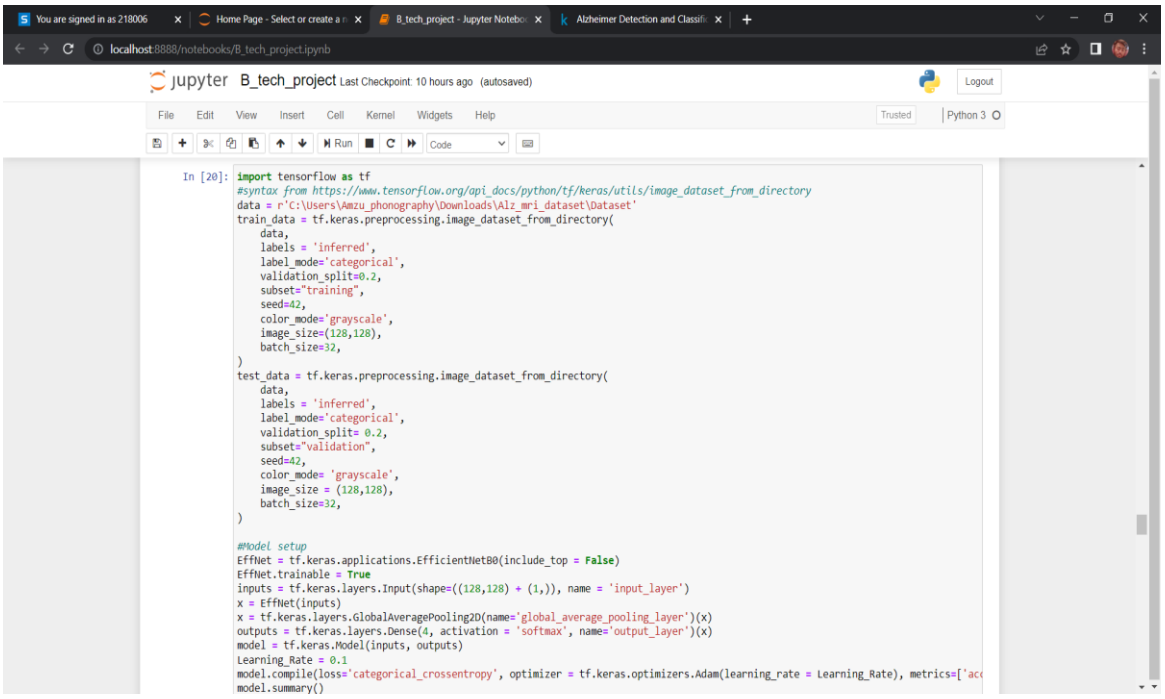
```
In [18]: #Train a SVM using RBF kernel
clf = svm.SVC(kernel = 'rbf')
clf.fit(train_x, train_y)

#store predictions and ground truth
y_pred = clf.predict(train_x)
y_true = train_y

#assess the performance of the SVM with linear kernel on Training data
print('Accuracy : ', metrics.accuracy_score(y_true, y_pred))

#Now, use the SVM model to predict Test data
y_pred = clf.predict(test_x)
y_true = test_y

#assess the performance of the SVM with linear kernel on Testing data
print('Accuracy : ', metrics.accuracy_score(y_true, y_pred))
```

```
Accuracy :  0.8126953125
Accuracy :  0.77421875
```

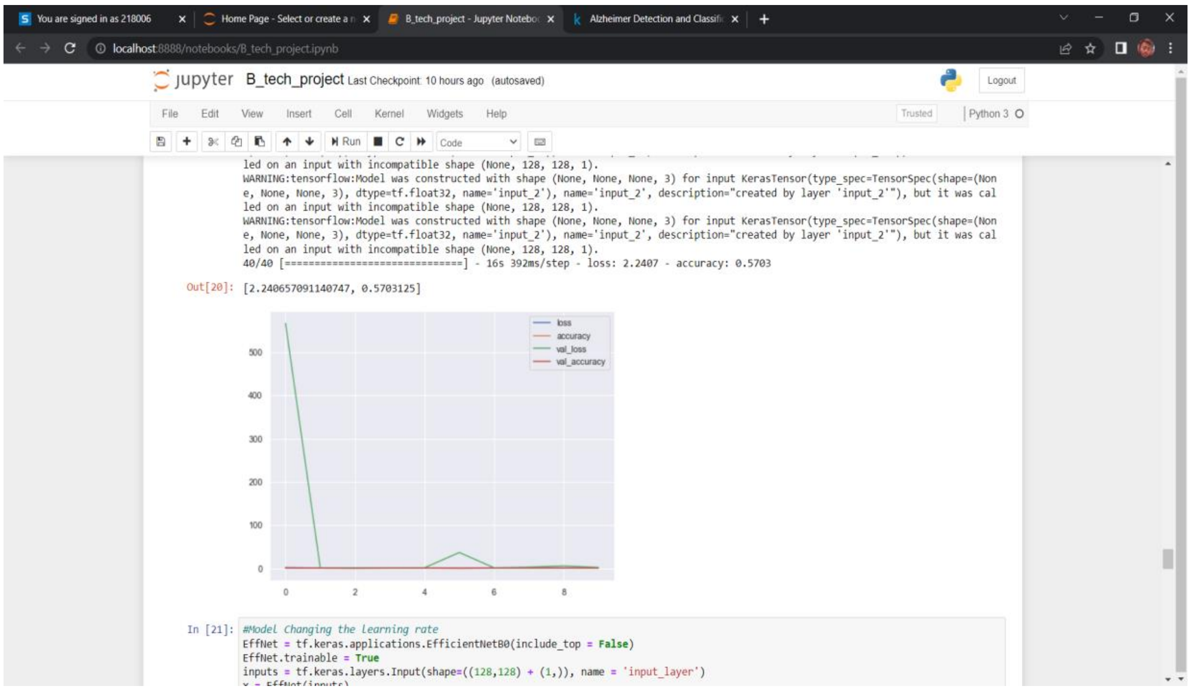
The above performance tells that the linear kernel still works the best for the classification as well compared to other complex kernels. Considering that kernel is used to separate the data with its shape, the data seems to be very well separated linearly in the upper dimension. Also, the accuracy of the linear kernel SVM model is 98.67% which is higher than just Alzheimer detection model from above.

CNN model for Alzheimer Classifier (EfficientNetB0)

Rather than using the typical and basic models, here, I use a bit more state-of-the-art model EfficientNet. Depending on the input data's shape, EfficientNet changes a little bit. However, for the smallest shape (usual input is 224x224) EfficientNetB0 is used.



```
In [20]: import tensorflow as tf
#syntax from https://www.tensorflow.org/api_docs/python/tf/keras/utils/image_dataset_from_directory
data = r'C:\Users\Amzu_phonography\Downloads\Alz_mri_dataset\Dataset'
train_data = tf.keras.preprocessing.image_dataset_from_directory(
    data,
    labels = 'inferred',
    label_mode='categorical',
    validation_split=0.2,
    subset="training",
    seed=42,
    color_mode='grayscale',
    image_size=(128,128),
    batch_size=32,
)
test_data = tf.keras.preprocessing.image_dataset_from_directory(
    data,
    labels = 'inferred',
    label_mode='categorical',
    validation_split= 0.2,
    subset="validation",
    seed=42,
    color_mode= 'grayscale',
    image_size = (128,128),
    batch_size=32,
)

#Model setup
EffNet = tf.keras.applications.EfficientNetB0(include_top = False)
EffNet.trainable = True
inputs = tf.keras.layers.Input(shape=((128,128) + (1,)), name = 'input_layer')
x = EffNet(inputs)
x = tf.keras.layers.GlobalAveragePooling2D(name='global_average_pooling_layer')(x)
outputs = tf.keras.layers.Dense(4, activation = 'softmax', name='output_layer')(x)
model = tf.keras.Model(inputs, outputs)
Learning_Rate = 0.1
model.compile(loss='categorical_crossentropy', optimizer = tf.keras.optimizers.Adam(learning_rate = Learning_Rate), metrics=['ac
model.summary()
```



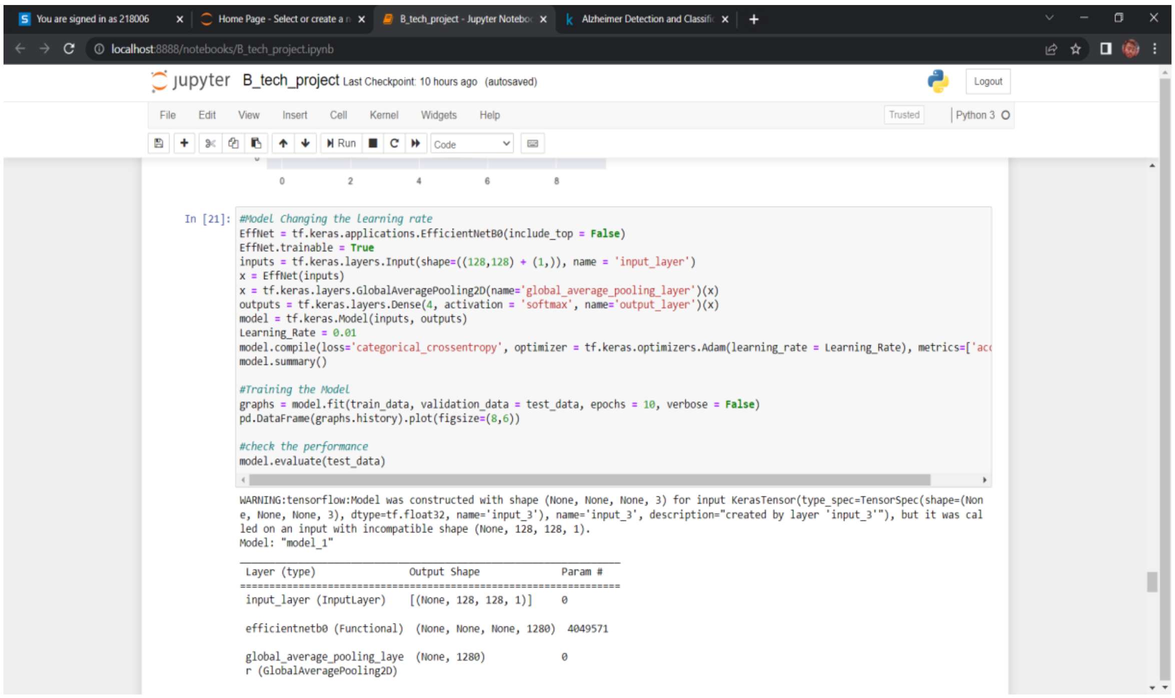
jupyter
B_tech_project Last Checkpoint: 10 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
loss
accuracy
val_loss
val_accuracy

```
led on an input with incompatible shape (None, 128, 128, 1).
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(Non
e, None, None, 3), dtype=tf.float32, name='input_2'), name='input_2', description="created by layer 'input_2'"), but it was cal
led on an input with incompatible shape (None, 128, 128, 1).
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(Non
e, None, None, 3), dtype=tf.float32, name='input_2'), name='input_2', description="created by layer 'input_2'"), but it was cal
led on an input with incompatible shape (None, 128, 128, 1).
40/40 [==============================] - 16s 392ms/step - loss: 2.2407 - accuracy: 0.5703
```

```
Out[20]: [2.240657091140747, 0.5703125]
```

```
In [21]: #Model Changing the learning rate
         EffNet = tf.keras.applications.EfficientNetB0(include_top = False)
         EffNet.trainable = True
         inputs = tf.keras.layers.Input(shape=((128,128) + (1,)), name = 'input_layer')
```

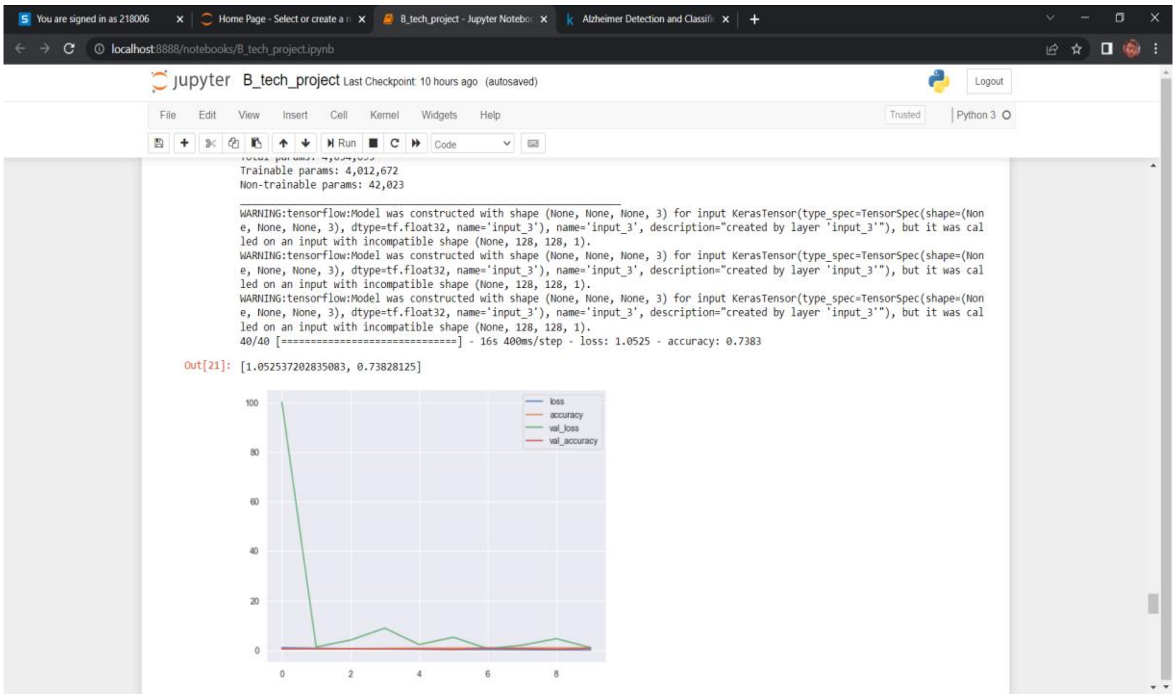
Before I make judgements, I'll try with smaller learning rate.



jupyter
B_tech_project Last Checkpoint: 10 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [21]: #Model Changing the learning rate
         EffNet = tf.keras.applications.EfficientNetB0(include_top = False)
         EffNet.trainable = True
         inputs = tf.keras.layers.Input(shape=((128,128) + (1,)), name = 'input_layer')
         x = EffNet(inputs)
         x = tf.keras.layers.GlobalAveragePooling2D(name='global_average_pooling_layer')(x)
         outputs = tf.keras.layers.Dense(4, activation = 'softmax', name='output_layer')(x)
         model = tf.keras.Model(inputs, outputs)
         Learning_Rate = 0.01
         model.compile(loss='categorical_crossentropy', optimizer = tf.keras.optimizers.Adam(learning_rate = Learning_Rate), metrics=['acc
         model.summary()

         #Training the Model
         graphs = model.fit(train_data, validation_data = test_data, epochs = 10, verbose = False)
         pd.DataFrame(graphs.history).plot(figsize=(8,6))

         #check the performance
         model.evaluate(test_data)
```

```
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(Non
e, None, None, 3), dtype=tf.float32, name='input_3'), name='input_3', description="created by layer 'input_3'"), but it was cal
led on an input with incompatible shape (None, 128, 128, 1).
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_layer (InputLayer)    [(None, 128, 128, 1)]     0

 efficientnetb0 (Functional)  (None, None, None, 1280)  4049571

 global_average_pooling_laye  (None, 1280)             0
 r (GlobalAveragePooling2D)
```

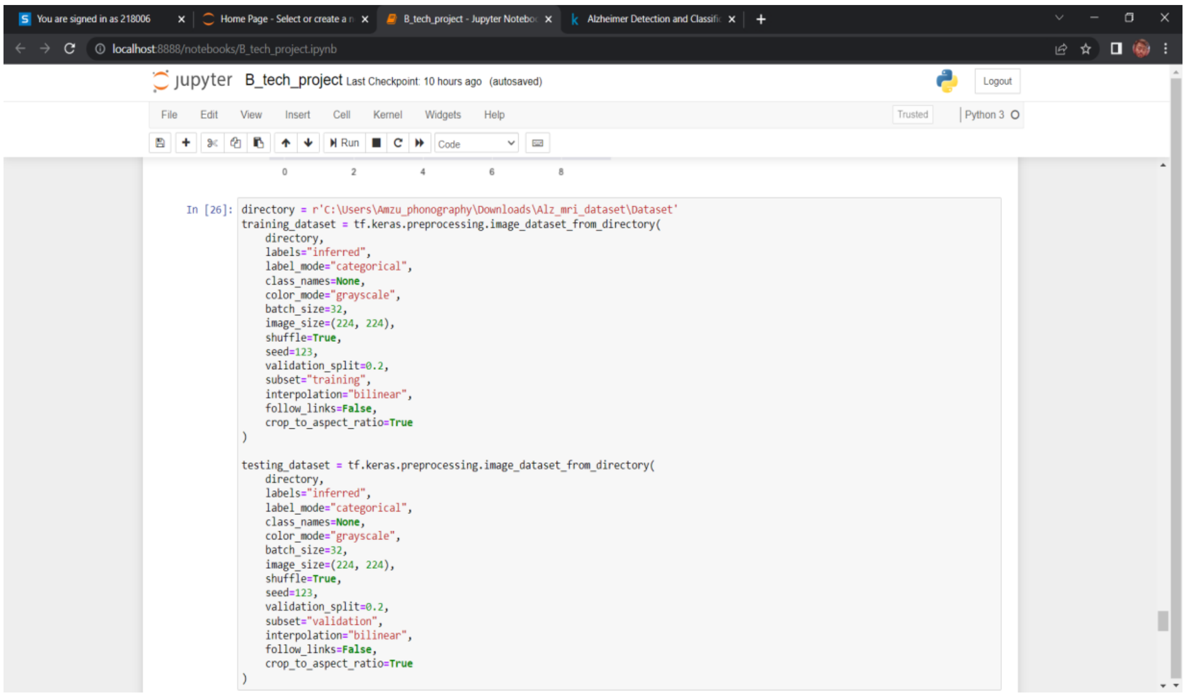


```
Trainable params: 4,012,672
Non-trainable params: 42,023
_________________________________________________________________
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(Non
e, None, None, 3), dtype=tf.float32, name='input_3'), name='input_3', description="created by layer 'input_3'"), but it was cal
led on an input with incompatible shape (None, 128, 128, 1).
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(Non
e, None, None, 3), dtype=tf.float32, name='input_3'), name='input_3', description="created by layer 'input_3'"), but it was cal
led on an input with incompatible shape (None, 128, 128, 1).
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(Non
e, None, None, 3), dtype=tf.float32, name='input_3'), name='input_3', description="created by layer 'input_3'"), but it was cal
led on an input with incompatible shape (None, 128, 128, 1).
40/40 [==============================] - 16s 400ms/step - loss: 1.0525 - accuracy: 0.7383
Out[21]: [1.052537202835083, 0.73828125]
```

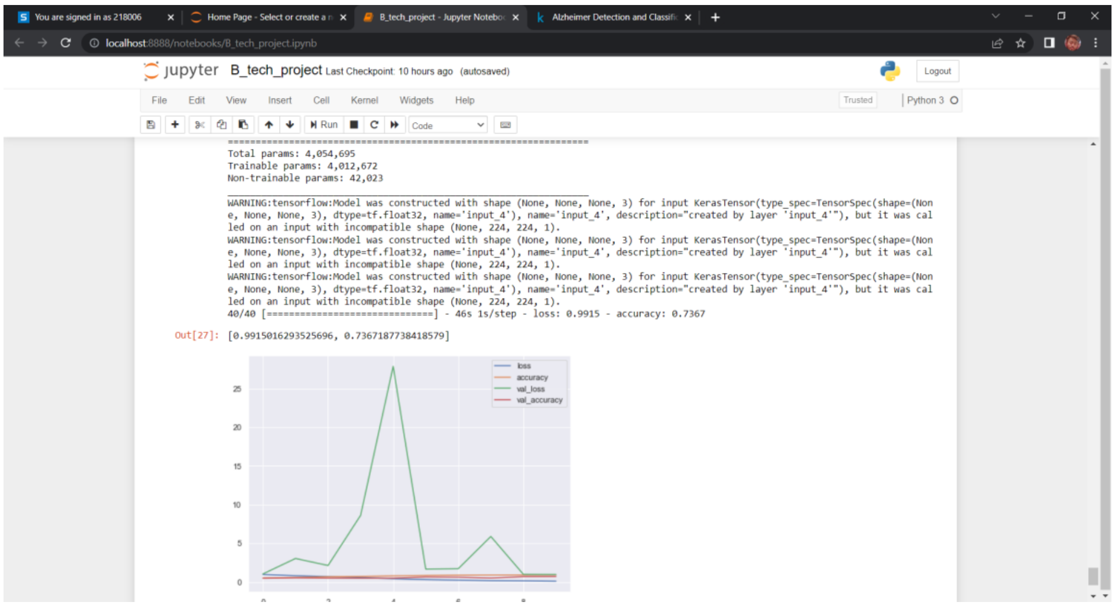
From the above, the performance of the existing code had 99% accuracy. But actually, having a different seed, the accuracy drops significantly to 58% with learning rate of 0.1 and 85% with learning rate of 0.01. No other fundamental changes in the code exists when it comes to learning rate of 0.01.

However, since there was 99% accuracy, there should be a way to boost the accuracy to at least 90%. The reason for having low accuracy may be the issue that the shape of the given data input is 128x128 rather than 224x224, which is the input that is expected for EfficientNetB0.

So, in below, I will resize the image and try running CNN again with the changed size.



```
In [26]: directory = r'C:\Users\Amzu_phonography\Downloads\Alz_mri_dataset\Dataset'
training_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    directory,
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="grayscale",
    batch_size=32,
    image_size=(224, 224),
    shuffle=True,
    seed=123,
    validation_split=0.2,
    subset="training",
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=True
)

testing_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    directory,
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="grayscale",
    batch_size=32,
    image_size=(224, 224),
    shuffle=True,
    seed=123,
    validation_split=0.2,
    subset="validation",
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=True
)
```



```
=================================================================
Total params: 4,054,695
Trainable params: 4,012,672
Non-trainable params: 42,023
_________________________________________________________________
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(Non
e, None, None, 3), dtype=tf.float32, name='input_4'), name='input_4', description="created by layer 'input_4'"), but it was cal
led on an input with incompatible shape (None, 224, 224, 1).
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(Non
e, None, None, 3), dtype=tf.float32, name='input_4'), name='input_4', description="created by layer 'input_4'"), but it was cal
led on an input with incompatible shape (None, 224, 224, 1).
WARNING:tensorflow:Model was constructed with shape (None, None, None, 3) for input KerasTensor(type_spec=TensorSpec(shape=(Non
e, None, None, 3), dtype=tf.float32, name='input_4'), name='input_4', description="created by layer 'input_4'"), but it was cal
led on an input with incompatible shape (None, 224, 224, 1).
40/40 [==============================] - 46s 1s/step - loss: 0.9915 - accuracy: 0.7367
Out[27]: [0.9915016293525696, 0.7367187738418579]
```

You are signed in as 218006
Home Page - Select or create a n
B_tech_project - Jupyter Notebo
Alzheimer Detection and Classifi
localhost:8888/notebooks/B_tech_project.ipynb
jupyter B_tech_project Last Checkpoint: 10 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
loss
accuracy
val_loss
val_accuracy
25
20
15
10
5
0

Overall, from the performance, this data really seems to fit well on SVM, which runs way faster than CNN. LDA also had a high accuracy as well on Alzheimer detection. But considering the nature of LDA, having to see that the data is linearly separable from SVM, it shouldn't be that different with LDA as it separates data linearly as well. Most surprising result was the that CNN did not work that well on my end with the high learning rate, I don't know what is the difference in fundamental, however, the result of my CNN models including VGG16 and EfficientNetB0 sometimes were trained to have lower accuracy even compared to random guessing. This part of the project was interesting to learn why 'right sized learning rate' is important overall as it increased the accuracy significantly by more than 30% compared to those with bigger learning rates.

Overall Summary

Having 80% accuracy and such performance on testing data is already 'good', however, in this project, there were many models that went over 90%. Here is the overall results of the model and its' performance. I highlighted the results with over 90% performance.

6.2 Result & Analysis

Project	**Models**	**Performance on test data**
Alzheimer Detection	PCA	174 axes needed to reach 80% accuracy
Alzheimer Detection	SVM (Kernel: Linear)	98% accuracy and f1 score
Alzheimer Detection	SVM (Kernel: 2-degree polynomial)	86% accuracy and f1 score
Alzheimer Detection	SVM (Kernel: RBF)	83% accuracy and f1 score
Alzheimer Detection	LDA	90.1% accuracy

Alzheimer Detection	CNN(VGG16)	50% accuracy
Alzheimer Classification	SVM (Kernel: Linear)	98.7% accuracy
Alzheimer Classification	SVM (Kernel:2-degree polynomial)	83.1% accuracy
Alzheimer Classification	SVM (Kernel: RBF)	77% accuracy
Alzheimer Classification	CNN(EffnetB0) - LR:0.1	58% accuracy
Alzheimer Classification	CNN(EffnetB0) - LR:0.01	85% accuracy
Alzheimer Classification	CNN(EfficientnetB0) - Resize	83% accuracy

Conclusion:

6.3 Conclusion and Future Scope

In this project, we used two machine learning algorithms i.e. Logistic Regression and Support Vector Machine on the Alzheimer’s disease dataset to build a model and further compare the accuracy obtained by each of the model. The project aimed at analysing the data through graphs to predict which of the features are more likely to cause dementia in a person. Through this project, given the features, we can easily predict and classified the stages of dementia as well as a person will have dementia or not.

Furthermore, a front end is including radio buttons and labels is developed using tkinter to perform functions like dropping the columns, converting gender and group to integer values and upload using the database and queries. All this can be done by a single click of button hence improving the usability of project.

6.3.1 Proposals for Future Works

Before deploying the model, it can be tested in systems and further used in a real time scenario. Also, this project can be used for generation of synthetic data for the early detection of Alzheimer’s disease by using the train and test model.

References:

[1] Karas GB, Scheltens P, Rombouts SA et al.: Global and local gray matter loss in mildcognitive impairment and Alzheimer’s disease, Neuroimage. 23(2), 708-716. 2004.

[2] Jack CR, Jr, Petersen RC, Xu Y et al.: Rates of hippocampal atrophy correlate withchange in clinical status in aging and Alzheimer’s disease, Neurology. 55, 484?489, 2000

[3] Iwatsubo T, Iwata A, Suzuki K et al.; Japanese and North American Alzheimer’s DiseaseNeuroimaging Initiative studies: harmonizationfor international trials, Alzheimers Dement 2018 doi: 10.1016/j.jalz.2018.03.009.

[4] Penny WD.: Statistical parametric mapping: the analysis of functional brain images,Academic Press, 2006.

[5] Otsu N; A Threshold Selection Method from Gray - Scale Histograms, IEEE Trans. SMCSMC 9(1), 62-66, 1979.

[6] Zahn, CT, Roskies, R.Z : Fourier descriptors for plane closed curves,IEEE Trans.Computers, C-21, 269-281, 1972.

[7] Granlund, GH : Fourier preprocessing for hand print character recognition, IEEE Trans.Computers, C-21, 195-201, 1972.

[8] Uesaka Y: A new Fourier descriptor applicable to open curves (in Japanese), IEICE Trans.Fund. Electr., 67-A(3), 166-173, 1984.

[9] Burger W, Burge MJ, Digital Image Processing, Springer-Verlag London,2017

[10] Carpenter, A. (2020,). Neural architecture transfer.
Medium. https://towardsdatascience.com/neural-architecture-transfer-54226b2306e3

[11] Marius, H. (2021). State-of-the-art image classification algorithm: Fixefficientnet-l2.
Medium. https://towardsdatascience.com/state-of-the-art-image-classification-algorithm-fixefficientnet-l2-98b93deeb04c

[12] Team, K. (n.d.). Keras documentation: Image classification via fine-tuning with EfficientNet. Retrieved
from https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/

AMJAD ALI

ORIGINALITY REPORT

8%	2%	1%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Visvesvaraya Technological University, Belagavi Student Paper	4%
2	Submitted to Christ University Student Paper	2%
3	louisdl.louislibraries.org Internet Source	1%
4	Submitted to Manipal University Student Paper	<1%
5	www.nature.com Internet Source	<1%
6	docksci.com Internet Source	<1%
7	ebin.pub Internet Source	<1%
8	"Progress in Artificial Intelligence", Springer Science and Business Media LLC, 2021 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		