A project report on

PROGNOSTIC MODEL FOR ALZHEIMER'S DISEASE USING DEEP LEARNING BASED ON MRI IMAGES

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

MOHIT KAUSHIK (19BCE1522)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

A project report on

PROGNOSTIC MODEL FOR ALZHEIMER'S DISEASE USING DEEP LEARNING BASED ON MRI IMAGES

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

MOHIT KAUSHIK (19BCE1522)

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

DECLARATION

I hereby declare that the thesis entitled "**PROGNOSTIC MODEL FOR ALZHEIMER'S DISEASE USING DEEP LEARNING BASED ON MRI IMAGES**" submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai, is a record of bonafide work carried out by me under the supervision of Dr. Ilavarasi A K.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai **MOHIT KAUSHIK**

Date: 15 – 04 – 2023 Signature of the Candidate

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **“Prognostic model for Alzheimer's Disease using deep learning based on MRI images”** is prepared and submitted by **Mohit Kaushik** (**19BCE1522**) to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** programme is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:

Name: Dr. ILAVARASI A K

Date: 24-04-2023

Signature of the Examiner 1	Signature of the Examiner 2
Name:	Name:
Date: 26-04-2023	Date: 26-04-2023

Approved by the Head of Department
B. Tech. CSE

Name: Dr. Nithyanandam P

Date: 24 – 04 – 2023

(Seal of SCOPE)

ABSTRACT

Elderly people with Alzheimer's disease (AD) suffer from a neurodegenerative brain condition. A critical component of preventing AD is identifying patients with moderate cognitive impairment (MCI) who are more likely to develop AD. Although the efficacy of combination of structural MRI (sMRI) and resting-state functional MRI (rs-fMRI) has not yet been thoroughly studied., both techniques have shown encouraging results in the to the identification of Alzheimer's disease (AD).We integrated them to investigate how they might be used to distinguish between MCI and AD as well as how patients might proceed from MCI to AD. With the help of rs-fMRI and sMRI characteristics, we developed and evaluated a variety of machine learning and deep learning algorithms to distinguish MCI to AD progression. Our method of determining MCI and AD progression made use of a limited amount of ideal features, and our deep learning model's give an accuracy for the last epoch's combination of sMRI and rs-fMRI dataset was 0.9986.The merging of sMRI and rs-fMRI for the prognostic prediction of AD is clarified by our findings.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. Ilavarasi A K, Assistant Professor (Senior), SCOPE, Vellore Institute of Technology, Chennai, for her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my endeavor. My association with her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Artificial Intelligence and Healthcare Analytics.

It is with gratitude that I would like to extend thanks to our honorable Chancellor, Dr. G. Viswanathan, Vice Presidents, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan and Mr. G V Selvam, Assistant Vice-President, Ms. Kadhambari S. Viswanathan, Vice-Chancellor, Dr. Rambabu Kodali, Pro-Vice Chancellor, Dr. V. S. Kanchana Bhaaskaran and Additional Registrar, Dr. P.K.Manoharan for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dean, Dr. Ganesan R, Associate Dean Academics, Dr. Parvathi R and Associate Dean Research, Dr. Geetha S, SCOPE, Vellore Institute of Technology, Chennai, for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Nithyanandam P, Head of the Department, Project Coordinators, Dr. Abdul Quadir Md, Dr. Priyadarshini R and Dr. Padmavathy T V, B. Tech. Computer Science and Engineering, SCOPE, Vellore Institute of Technology, Chennai, for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staff at Vellore Institute of Technology, Chennai, who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai **MOHIT KAUSHIK**

Date: 15 – 04 - 2023 Name of the student

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

MCI	Mild Cognitive Impairment
AD	Alzheimer’s disease
MRI	Magnetic Resonance Imaging
Fig	Figure
CNN	Convolutional Neural Network
ML	Machine Learning
CV	Computer Vision

Chapter 1

Introduction

Alzheimer’s disease may be an encephalopathy that slowly destroys memory and thinking skills and, eventually, the power to hold out the simplest tasks. Patients would eventually die as a result of the condition. Until date, the cause of Alzheimer's disease has remained unknown, and no effective medications or treatments have been reported to slow or prevent the progression of the disease. It steadily worsens over time. Initially, the changes are minor and may be difficult to detect. They do, however, become more prominent and conspicuous over time. Your capacity to function independently will deteriorate as the condition develops. The disease affects individuals to different degrees, and none of the instances of Alzheimer's are comparable. Early on, you may notice slight memory loss or difficulty with thinking and judgment. You may have difficulty remembering names, paying bills, preparing meals, or caring for yourself. You could observe that, as the illness worsens, your loved one is having difficulty performing more challenging jobs, including driving a car or doing housework. It is not mandatory that individual with MCI will surely progress to Alzheimer’s disease. The chance of acquiring Alzheimer's disease is higher for people with amnestic MCI, according to a study. Alzheimer's disease has no known cure, and available therapies merely treat its symptoms. The advancement of Alzheimer's disease can be hampered, though, with early MCI detection and therapeutic intervention. For the purpose of identifying the root of their symptoms, people who are experiencing cognitive impairment should see a doctor. Alzheimer's disease (AD) is the cause of more than 60% of dementia cases, with patients typically experiencing increasing memory loss, language problems, and disorientation. Alzheimer disease is the most prevalent form of the irreversible dementias, which affect an estimated 3 to 4 million patients, accounting for 60% or more of cases. To assess whether a patient has MCI or Alzheimer's disease, doctors may employ cognitive tests, brain imaging, and other diagnostic techniques. Treatment may include symptom management drugs, lifestyle changes, and cognitive training. Mild cognitive impairment (MCI) is a condition when a person has deterioration in cognitive performance that is higher than what would be expected for their age and educational level, but not to the extent that it affects every day functioning. MCI is frequently regarded as a step between normal ageing and dementia.

Memory loss and increasing cognitive decline are symptoms of Alzheimer's disease, a form of dementia. While not everyone who has MCI will go on to have Alzheimer's disease, research has revealed that those who do have MCI have a higher risk of getting the condition than those who don't. It is crucial to highlight that not all cases of MCI proceed to Alzheimer's disease. As a matter of fact, some MCI sufferers may

experience steady cognitive function or even improvements in cognitive function over time.
MCI is challenging to identify, particularly in the early phases when there aren't many symptoms. The Mini-Mental State Examination (MMSE), Montreal Cognitive Assessment (MoCA), and Mattis Dementia Rating Scale (MDRS) are three cognitive tests that might assist identify MCI.
MCI can be caused by a variety of factors, including: Stroke, which happens when a blood vessel supplying the brain with oxygen and nutrients becomes blocked and impairs cognitive activity, can result in MCI. MCI comes in three varieties: **Progressive**, which denotes that the symptoms will progressively get worse. Dementia or other health issues could result from this. **Transient**, indicating that the condition will pass and have no lasting effects on the individual. **Atypical**, which means the individual may not have any of the typical symptoms or may have symptoms that are not typical of MCI.
As there is still much about the illness that is unknown and as each person's experience with MCI and Alzheimer's disease can differ, it can be difficult to predict how MCI will progress to AD. Some elements, however, might make development more predictable. These include: -A history of diabetes or high blood pressure, which is linked to an increased risk of getting Alzheimer's disease. Furthermore, various risk factors such as advancing age, hereditary variables, head injuries, tube disorders, infections, and environmental factors all have a role in the disease. [2]. In order to improve the prognosis of Alzheimer's disease in MCI patients, research is also being done to create novel biomarkers and technologies.
The objective of this study is to create a model that can be used as a biomarker to identify MCI patients who are more prone to develop Alzheimer's disease. Deep learning has shown significant potential in medical image analysis because of its ability to extract high-level features from large datasets; several papers on this subject have been released using various techniques and datasets. We present a deep-learning model for predicting the Alzheimer's disease from medical images in this article. Our model is built on top of a Convolutional Neural Network (CNN), which has numerous layers and is trained using supervised learning methods like back propagation through time (BPTT) and stochastic gradient descent. (SGD).

LITERATURE REVIEW

There are several research that are relevant to the diagnosis of Alzheimer's illnesses. With an accuracy of 0.83, Kavitha C, et al. use a variety of machine learning algorithms to predict Alzheimer's disease in its early stages using the Open Access Series of Imaging Studies (OASIS) dataset. Lingling developed a prediction model for AD detection in the elderly population using regression analysis based on the Least Absolute Shrinkage Selection Operator (LASSO). According to the research, the training set's area under the ROC curve was 0.822, while the validation set's was 0.801, indicating that the model created with these 6 predictors had a modest level of predictive ability.

In order to identify dementia in patients, Morshedul Bari Antor et al. conducted an analysis of comparison of machine learning algorithms. In accordance to test data, the author claimed a 92% accuracy rate. Selim Buyrukoglu and colleagues build a prediction model for early Alzheimer's disease diagnosis by employing homogeneous and heterogeneous ensemble approaches in the feature selection process. The study asserts that the Random Forest algorithm, when used with a feature subset derived from a heterogeneous ensemble technique, provides the highest accuracy of 0.91.

The objective of Charlotte James, et.al., is to evaluate machine learning algorithms' ability to forecast dementia occurrence within two years to existing models and to identify the best analytical strategy and an appropriate number of variables to employ. The predictive study collected information from 15307 people to perform secondary analysis on variables that potentially influence the likelihood of developing dementia. According to the research, only 6 variables are required to obtain 90% accuracy in predicting the dementia incidence.

In this research, Sarah Parisot, et.al., provides a comprehensive assessment of a general paradigm for brain analysis in large populations that makes use of both imaging and non-imaging data. The effectiveness of the framework was examined on two sizable datasets with a variety of underlying data, ABIDE and ADNI, for the diagnosis of Alzheimer's disease and the prediction of Autism Spectrum Disorder, respectively. The research we conducted demonstrates that, with an 80.0% classification accuracy for ADNI and a 70.4% classification accuracy for ABIDE, our innovative approach can outperform state-of-the-art findings on both databases. Jack Albright's suggested system makes use of neural network technology. In order to forecast the future course of AD using clinical data, he looked into the use of machine learning techniques. The "All-Pairs" technique, a novel methodology developed for this study that involves comparing all feasible pairings of temporal data points for each patient, was used to process data from 1737 patients. In both patients with mild cognitive impairment and patients whose initial cognitive function was normal, the research asserts that a neural network model was successful (mAUC = 0.866) at predicting the course of AD.

In their study, Lee Kuok Leong et al. suggested comparing and evaluating several machine learning methods with pre-processing and feature selection using the Boruta algorithm for the prediction of AD. This study uses a variety of methods, including data pre-processing, feature selection using the Boruta algorithm, and data acquisition. The experimental findings demonstrate the Boruta Algorithm's successful performance, which had an accuracy of 94.39%.

Based on hippocampus MRI, the approach known as deep learning is employed for early Alzheimer's disease dementia prediction. In addition to discussing dementia caused by AD in a time-to-event analysis context, this research also classified individual participants into subgroups based on the anticipated progression risk.The author's c-index result of 0.762 indicates that they correctly predicted the dementia case. The research by Cristina L. Saratxaga et al. largely utilises a novel approach based on deep learning and image processing techniques for MRI-based Alzheimer's diagnosis is proposed and contrasted with earlier literature efforts.

Deevyankar Agarwal, et.al. conducted a thorough analysis of the state of early Alzheimer's disease diagnosis using deep learning models with transfer learning and neuroimaging biomarkers. When using a combination of 3D convolutional networks and local transfer learning, the classification of AD reached the greatest classification accuracy of 98.20%, and the accuracy achieved for the prognosis model is 87.78%. A deep learning approach employing residual neural networks and rs-fmri was proposed by Farhen Ramzan et al. for the automated diagnosis and multiclass category of Alzheimer's disease phases. For CN, SMC, EMCI, LMCI, MCI, and AD, respectively, the accuracy obtained using the help find tune models is 100%, 96.85%, 97.38%, 97.43%, 97.40%, and 98.01%.

MRI data based prognostic prediction is done in Yong Fan proposed system. He predicted whether the individual will progress from AD Dementia from MCI. The deep learning time to-event model effectively predicted each subject's progression to AD dementia on 439 ADNI-assessing MCI individuals, with a concordance value of 0.762 and follow-up spanning from 6 to 78 months. For the prediction of MCI and AD, Ramfrez et al. developed an ensemble method using random forest one vs rest classifier. The suggested method achieves a classification accuracy of 56.25% on the test subset of 160 real patients. A strategy for diagnosing AD and tracking the course of the disease was suggested in the study written by George Manis et al. The model's monitoring accuracy was 99% and detection accuracy of 94% respectively.

S.No	Title	Year	Model	Metrics
1	A six stage approach for the diagnosis of the Alzheimer's disease based on fMRI data	2009	Random Forest Support Vector Machine	Accuracy - 0.94

2	Ensemble of random forests One vs. Rest classifiers for MCI and AD prediction using ANOVA cortical and subcortical feature selection and partial least squares	2017	ANOVA feature selection Partial least squares Ensemble of One vs. Rest random forest classifiers	Accuracy - 0.56
3	Disease prediction using graph convolutional networks: Application to Autism Spectrum Disorder and Alzheimer's disease	2018	Graph Convolutional Network	Accuracy - 0.80
4	Forecasting the progression of Alzheimer's disease using neural networks and a novel preprocessing algorithm	2019	Neural networks and a Novel Preprocessing algorithm	AUC - 0.866
5	Prediction of Alzheimer's disease (AD) Using Machine Learning Techniques with Boruta Algorithm as Feature Selection Method	2019	Boruta Algorithm Deep Neural Network (DNN) Random Forest (RF) Gradient Boosting Machines (GBM)	Accuracy - 0.943

			Support Vector Machine (SVM) Logistic Regression (LR)	
6	A deep learning model for early prediction of Alzheimer’s disease dementia based on hippocampal MRI	2019	Deep Learning Model	C-index - 0.762
7	A deep learning model for early prediction of Alzheimer's disease dementia based on hippocampal magnetic resonance imaging data	2019	Deep-learning time-to-event model	C-index - 0.864
8	A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer’s Disease Stages Using Resting-State fMRI and Residual Neural Networks	2020	Deep learning algorit hm	Accuracy - 0.98
9	Alzheimer Disease Forecasting using Machine Learning Algorithm		K-Nearest Neighbor Adaboost	

		2020	Classifier Support Vector Machine Logistic Regression Decision Tree Random Forest classifier	Accuracy - 0.868
10	Construction of a risk prediction model for Alzheimer's disease in the elderly population	2021	Least Absolute Shrinkage Selection Operator (LASSO) regression analysis	AUC - 0.801
11	A Comparative Analysis of Machine Learning Algorithms to Predict Alzheimer's disease	2021	Logistic Regression Decision Tree Random Forest Support Vector Machine	Accuracy - 0.92
12	EARLY DETECTION OF ALZHEIMER'S DISEASE USING DATA MINING: COMPARISON OF ENSEMBLE FEATURE SELECTION APPROACHES	2021	Random Forest Artificial Neural Network Logistic Regression Support Vector Machine Naive Bayes data mining	Accuracy - 0.91

			algorithms	
13	Performance of Machine Learning Algorithms for Predicting Progression to Dementia in Memory Clinic Patients	2021	Gradient-boosted trees algorithm Logistic Regression Support Vector Machine Random Forest	Accuracy - 0.90
14	Transfer Learning for Alzheimer's Disease through Neuroimaging Biomarkers: A Systematic Review	2021	3D convolutional network	Accuracy - 0.982
15	Early-Stage Alzheimer's Disease Prediction Using Machine Learning Models	2022	Decision Tree Random Forest Gradient Boosting Voting classifier Support Vector Machine	Accuracy - 0.83

Table 1

Chapter 2

Materials and methods

2.1. DATASETS

The Alzheimer's Disease Neuroimaging Initiative (ADNI) database provided the information needed for this study. The dataset used for this study are structural MRI (sMRI) and resting-state functional MRI (rs-fMRI). sMRI is a non-invasive technique for studying brain architecture and disease whereas rs-fMRI studies about the activity in the brain of the individual in a resting state. The present study includes 22 patients among which 5 belong to Mild Cognitive Impairment (MCI) research group and rest 17 belongs to Alzheimer’s disease (AD) research group. Each patient data is further divided into slices. The rs-fMRI dataset have 48 slices for each patient while sMRI have 256 slices per patient. Each slice displays the MRI image from a fresh angle. The patients are selected from different age group starting from 55 years to 90 years old. Participants used a 3T Philips scanner to scan for sMRI and rs-fMRI in accordance with the ADNI collection protocol. rs-fMRI data were acquired with a Gradient Echo (GRE) sequences [Repetition time (TR) = 3000ms; Echo Time (TE) = 30.0ms; matrix=64.0 x 64.0 pixel; flip angle=80.0 degree, voxel size = 3.3mm^3 x 3.3mm^3 x 3.3mm^3). With T1- weighted Magnetization Prepared Rapid Acquisition Gradient Echo (MPRAGE) sequence , sMRI data were collected [TR = 6.8ms; TE = 3.2ms; matrix= 256 x 256 x 170 pixels; flip angle = 8.0 degree; voxel size = 1 mm^3 x 1 mm^3 x 1 mm^3].

2.2. PROPOSED METHODOLOGY

The figure below depicts a block diagram of the proposed system for MCI prediction on MRI data. The proposed system is to use Magnetic Resonance Imaging (MRI) images to predict the conversion of patient’s from Mild Cognitive Impairment to Alzheimer’s diseases with the help of their current situation. To build more realistic model, we will combine structural MRI data with resting state functional MRI data.

The proposed system is divided into three basic modules: Data Pre – Processing, Feature Extraction and Selection and methods. The data pre-processing modules purely focused on cleaning the data. Both the dataset will be pre-processed before training the deep learning model. Pre-processing techniques like intensity normalization, tissue segmentation, correction of the non-uniformity of image intensity, etc. are performed under this module. In feature extraction and feature selection modules, we use VGG16 pre-trained model for feature extraction. **VGG16** refers to a VGG model with 16 weight layers then we used LASSO feature selection algorithm to select the necessary features from the data. LASSO is an ensemble method which combinable use both filter and wrapper technique for feature selection.

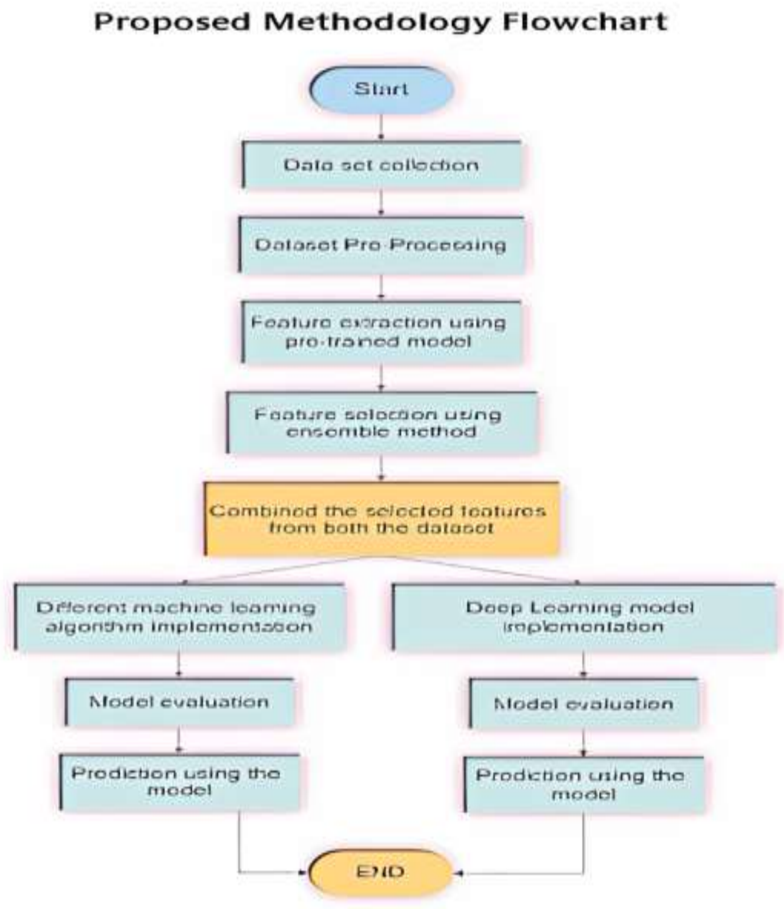

Proposed Methodology Flowchart
Start
Data set collection
Dataset Pre Processing
Feature extraction using pre-trained model
Feature selection using ensemble method
Combined the selected features from both the dataset
Different machine learning algorithm implementation
Deep Learning model implementation
Model evaluation
Model evaluation
Prediction using the model
Prediction using the model
END

Fig. 1

After selecting the optimal features from the extracted features, we split our data into testing and training data. Different machine learning and ensemble based algorithms were used on different parameters to make classification. We found out that in certain cases, the model faces over-fitting, so to reduce it, I decided to go for data augmentation. Such a strategy can assist our neural network classifier in reducing overfitting, or in other words, it can improve the model's ability to generalize data samples. In an attempt to improve classification, we used Convolution Neural Network (CNN) deep learning technique. The model is evaluated on the basis of accuracy it achieved on training and testing data.

2.3 FEATURE STANDARDIZATION

When the mean in the numerator is subtracted from each feature's value, it results in zero-mean and unit-variance. Numerous machine learning methods, such as SVMs, logistic regression, and ANNs, utilize this technique extensively for normalization. The preprocessing phase is of utmost significance when thinking about a classification task in machine learning. This stage can have a significant impact on the

classification outcomes, particularly when using support vector machines (SVMs). The range of the features in machine learning is standardized using a technique called feature scaling.
Calculations are generally made by finding the distribution mean and standard deviation for each characteristic. The mean is then subtracted from each feature. The values of each feature are then divided by their respective standard deviations (mean has already been eliminated). Standardising the numerical features offered for this challenge to zero-mean and unit variance

$$\hat{x} = \frac{x - \bar{x}}{\sigma}$$

Fig. 2

.
where the original feature, its mean, and its standard deviation are, in turn, denoted by the letters x, ‾x, and σ.
Outliers in the data won't be impacted by standardisation because it doesn't have a bounding range. Standardisation is useful in clustering analysis to assess similarities across features based on certain distance metrics.

2.4 FEATURE EXTRACTION USING PRE-TRAINED MODEL

The technique of converting raw data into a mathematical operation while keeping the data from the original database is known as feature extraction. One method for producing effective results from raw data is feature extraction using pre-trained models. In general, we need to gather a lot of training data and put the model through a lot of difficult steps like resampling, binning, and normalization. As a result, feature extraction can be time-consuming while also increasing model efficiency. Instead of creating new models from scratch, we may quickly extract characteristics from raw data by using predefined models. In this study, we extract features from MRI scans using the VGG16 model.

VGG16 is a well-known convolutional neural network (CNN) architecture that was introduced by the Visual Geometry Group (VGG) at the University of Oxford in 2014. It was proposed by Karen Simonyan and Andrew Zisserman in their paper titled "Very Deep Convolutional Networks for Large-Scale Image Recognition" and has since gained widespread attention and adoption in the field of computer vision. The VGG16 architecture comprises of 16 layers, including 13 convolutional layers and 3 fully connected layers. These convolutional layers are responsible for extracting features from input images, while the fully connected layers are used for making predictions. VGG16 is known for using small 3x3 convolutional filters in each convolutional layer, allowing it to capture local patterns and features in images and learn complex hierarchical features.

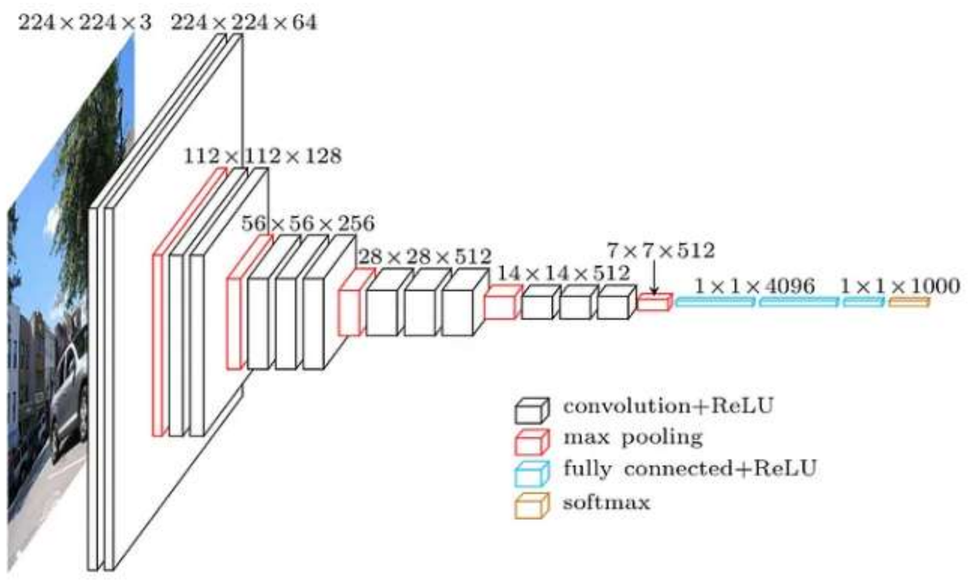


Fig. 3 VGG16 Architecture

One of the key features of VGG16 is the use of max pooling layers after each set of convolutional layers. Max pooling is a down sampling operation that reduces the spatial dimensions of the feature maps, while retaining the most important features. This helps in reducing the computational requirements and makes the network more efficient. The fully connected layers in VGG16 are used for making predictions based on the features extracted from the convolutional layers. These layers are responsible for mapping the learned features to the output classes in the task of image classification. The last fully connected layer is usually followed by a softmax activation function, which produces the class probabilities for the input image.

VGG16 has a fixed input size of 224x224 pixels, which means that input images are resized to this size before being fed into the network. This fixed input size is a trade-off between capturing fine-grained details and keeping the computational requirements manageable. It also requires images to be preprocessed by subtracting the mean RGB values of the training dataset from each pixel, which helps in reducing the impact of lighting conditions and improves the model's performance.

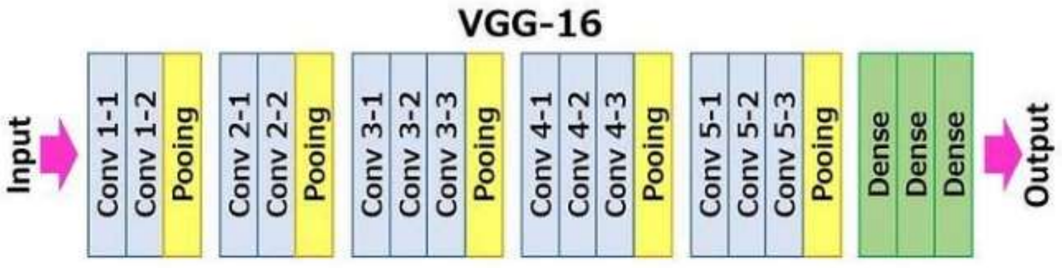


Fig. 4 VGG16 Architecture Map

VGG16 achieved top-performing results in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014, a benchmark competition for image classification. It surpassed the previous best performance with a top-5 error rate of 7.3% on the validation set, showcasing its excellent performance in large-scale image recognition tasks. However, VGG16 has some limitations, including its relatively deeper architecture with more parameters compared to some other CNN architectures. This may result in higher computational requirements during training and inference, making it less suitable for resource-constrained environments. Nonetheless, VGG16 has become a popular choice for transfer learning, where pre-trained VGG16 models are used as a starting point for training on other image recognition tasks with smaller datasets.

In summary, VGG16 is a widely used CNN architecture known for its simplicity, effectiveness, and state-of-the-art performance in image recognition tasks. Its use of small convolutional filters, fixed input size, and pre-processing steps make it a powerful tool for large-scale image recognition applications. Despite its limitations, VGG16 has had a significant impact on the field of computer vision and continues to be a widely used architecture in various image recognition tasks.

2.5 FEATURE SELECTION BASED ON LASSO METHOD

By limiting the number of dimensions, feature selection can enhance model performance and avoid overfitting. This study successfully distinguished between MCI and AD using feature selection. We used the least absolute shrinkage, selection operator (Lasso), and a linear model to choose a subset of features that could differentiate between MCI and AD. It is a statistical formula used mostly for the regularization and feature selection of data models.

It capitalises on the shrinkage approach, also referred to as the penalised regression method, as part of its regularisation strategy for feature selection. The finalisation factor g, known as alpha, was employed in this method to regularize/adjust the coefficient estimate to zero using the L1 penalty. Following the shrinkage, comes the feature selection phase, during which all non-zero values are chosen to be incorporated in the model. The reduction of prediction mistakes, which are frequent in statistical models, can be significantly reduced with this technique.

Models with good prediction accuracy are available from LASSO. Since the method involves coefficient shrinkage, which lowers variance and minimises bias, accuracy increases. It operates most effectively when there are few observations and many features. In order to limit shrinkage, parameter, which is significantly reliant upon λ. The more λ coefficients that must be zero, the greater grow.

The model changes to the Ordinary Least Squares regression when λ is equal to zero. As a result, as λ rises, the variance declines noticeably and the bias in the result rises as well.

We carried out the lasso approach using sklearn Library from python. It identifies significant features crucial for conversion from MCI to AD with Alpha value = 0.1 and random state value =10. Using the slope value, this method chooses the features. We will eliminate the features with poor slope value in favor of only choosing the best features.

Chapter 3

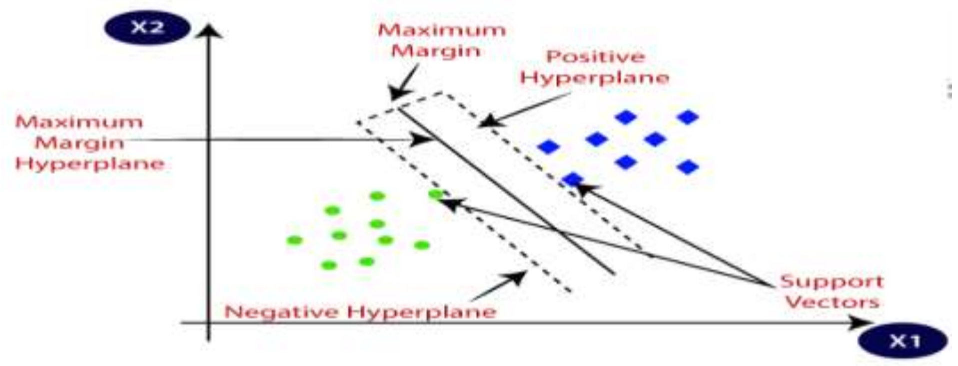
MODELS

3.1. SUPPORT VECTOR MACHINE

When you train a model developed using machine learning with labeled data, this is known as supervised learning. It indicates that the appropriate classification has already been applied to the data you have. SVM is a supervised ML model used for classification as well as regression but mainly performed classification..

It can be used to separate two different classes, for instance, if we have two shapes square and oval and we want to separate them into different classes and wants our model to correctly classify the shape into their desired category. SVMs were first presented in the 1960s, however they were later improved in 1990. SVMs are implemented in a different way than other machine learning algorithms. They have recently gained a lot of popularity because to their capacity to manage numerous continuous and categorical variables. This algorithm basically works on the construction of decision boundary, which cut off the space into different sections and each section contains different class.
This decision boundary is called as hyper plane. But the question arises is how do this model decides the boundary? So the answer to this question is support vectors. Support vectors are basically the uttermost points of each class from the boundary. There are two types of hyper plane that are present in the model: positive hyper plane and negative hyper plane. The positive hyper plane coincide with the positive class extreme point while negative hyper plane coincide with the negative class extreme points

Although, we have linear SVM classifier which separates the class well but we won't recommend it because the distance between the positive and negative hyper plane is minimum and it is not good for correct classification, so we prefer hyper plane that are aligned at some angle because the distance between positive and negative hyper plane is maximum in this case.



Important SVM principles include the following:–

- Support Vectors – the data points which lie nearest to the hyper plane. With the aid of these data points, a boundary line will be determined.
- Hyperplane − It is basically a decision boundary which separates different classes across its sides.
- Margin − It can be thought of as space present between the positive and negative hyper plane. Maximum margin is thought to be a good margin, whereas a minimum margin is thought to be a bad margin.

Different kernel types, including linear, polynomial, rbf, and sigmoid, are used in SVM algorithms. When there are several features, as there are in text classification tasks, the linear kernel performs incredibly well. In comparison to the majority of the others, linear kernel functions are faster and require less parameter optimisations. The equation of linear kernel is defined as follow:

$$y = weight ^T * X + b$$

where, weight stands for the weight vector, X for the input data we want trying to classify, and b stands for predicted linear coefficient. This equation represents the decision boundary of SVM. Due to its lower processing efficiency and less precise predictions compared to other kernels, the polynomial kernel is not frequently utilized in practice. A polynomial kernel's equation is given below:

$$y = (a + X1^T * X2) ^ b$$

The polynomial kernel equation we are using is one of the most straightforward ones. The polynomial decision boundary used to divide your data is represented by the expression y. Your data are represented by the two Xs.

The Gaussian Radial Basis Function (RBF) is widely used kernels in SVMs. The kernel is mostly used for dealing with the non-linearty data and helps in developing non-linear decision boundary. An RBF kernel's equation is as follows:

$$f(X1, X2) = \exp(-\text{gamma} * \|X1 - X2\|^2)$$

Gamma in this equation describes the influence that a single training point has on the surrounding data points. The dot product between your features is $\|X1 - X2\|$.

3.2. RANDOM FOREST ALGORITHM

Random forest is a supervised ML model capable of doing both regression and classification. This algorithm is used to tackle difficult problems such as classifying the dry fruits into their particular category, stock market prediction, etc., and all this is possible with the help of ensemble learning. It is called as a forest because it contains n numbers of decision trees which combinely work to make the decision. With the help of bagging technique, random forest train's its network. Decision tree outputs are used by random forest to make the final output. In this method, all decision trees take a subset of data points from the training data and with the help of this data they make their decision. After getting n outputs, the random forest use voting method to make the final output, if there is a classification task then the final output is decided by selecting the output with the maximum number of votes and if it is a regression problem then to get the final output we take the average value of all the outputs.

With the increase in the number of estimators, we can do the following tasks- elevates the accuracy of the model, records high precision and prevents over-fitting. As compared to decision tree, in random forest we are free to randomly select the root node and the group of nodes.

Bagging is the practice of using multiple samples of data (training data) rather than a single sample. A training dataset contains observations and qualities that will be used to produce predictions. Depending on the training data the decision trees and the random forest method produce varied results. After these outputs are ranked, the highest-ranked output will be picked as the final output.

Root node, leaf node and decision node are the crucial component of every decision tree. The leaf node generally depicts the result made by each decision tree and with the help of maximum voting method; we can easily finalize our output by selecting the result with the highest frequency. A straightforward random forest diagram is shown below:

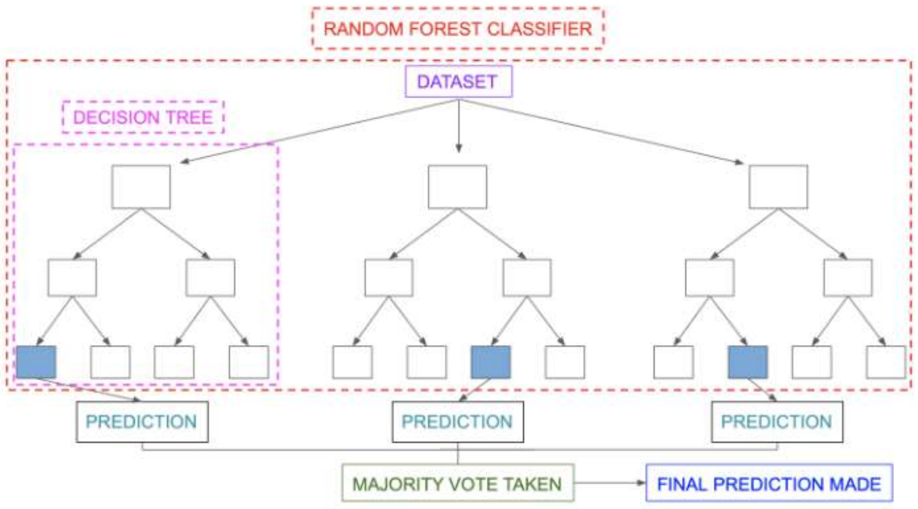


Fig. 6

We start the classification process with the optimum training data and selected features, as it will enhance the speed of computation and also reduce the time complexity of algorithm. Random forest helps in identifying which feature contribute in making the correct decision and following training, it calculates this score automatically for each feature and adjusts the findings to make the total importance equal to 1. Accordingly we will highlight the features with the high score and separate them from rest of the features.

n_estimators hyper parameter depicts the number of decision tree, random forest is using for making the decision.

max_feature hyper parameter represents the maximum number of features, the model is using to distinguish the nodes from one another.

n_jobs hyper parameter is used to tells the engine about the number of processor to be used during computation

The model's output is reproducible thanks to the **random_state** hyperparameter. When the model has a fixed value for random_state, the same hyper parameters, and the same training data, it will consistently generate the same results.

One of random forest's key advantages is its versatility. It makes it simple to observe the relative weights it assigns to the input characteristics.
The random forest approach is also highly useful because it typically correct predictions when utilizing its default hyper parameters. There aren't many hyper parameters and knowing them is easy. Over-fitting is one of the most serious problems with machine learning, but the random forest classifier will usually prevent it.

3.3. LOGISTIC REGRESSION

In classification problems, when we try to find out the possibility of an input belongs to particular class we use logistic regression. It is a supervised ML method and is further divided into 3 types based on their categories output - binomial, multinomial and ordinal category. In binomial class, there could be two possible outcomes like accept or reject, while in case of multinomial more than two unordered outcomes are possible, for instance - car, jeep, truck, etc. While if we are looking for some ordered outcomes like ranking of students then we should go for ordinal category.

In Logistic regression, instead of getting exact value we generally get a probability value that fall between 0 and 1. Instead of using a regression line in logistic regression, we use a S - shaped curve that foretells two maximum value either 0 or 1. This curve represents the probability of certain outcome, such as whether he go to hospital or not, eat the food or not, etc. The classification of observations using various type of data is possible with the help of logistic regression, which also makes it simple to identify the best variable to utilize.

It is used for classification algorithms its name is logistic regression. It is referred to as regression because it takes the output of the linear regression function as input and uses a sigmoid function to estimate the probability for the given class. Sigmoid function plays a vital role in the logistic regression working because sigmoid function is responsible for converting the predicted values into probabilities. To get the probability value from the predicted value, we have to perform certain operation. First of all we have to find the log of odds - i.e., the log function of probability divided by its complement and then we will put this log of odds value into sigmoid function to get the desired probability value.

The equation of sigmoid function is as follow:

$$q = 1 / (1 + \exp(-p))$$

at this place, q describes the dependent variable and p represents the independent variable.

Sigmoid function is also responsible for the activation of neuron. It mainly disturbs the linear pattern of the regression line and creates non-linearity to train the model to learn difficult task and improve the model. It basically work on the threshold value which is generally 0.5, if any values lies above 0.5 then in case of binary classification the model predict it as 1 else 0.

Logistic regression differs from linear regression in that it predicts the likelihood that an instance will belong to a specific class or not, whereas the output of the former is a continuous value that can be anything. The softmax function is utilized in this instance instead of the sigmoid function.

3.4. ENSEMBLE METHOD

Even though machine learning algorithms performs very well but there are some cases where one ML model failed drastically and other performs well, but we don't know under which scenario which algorithms will pass and which one will fail, so to overcome this, experts come up with a technique called ensemble learning. In this technique, numerous machine learning algorithms are combined together to predict the output. It will be helpful in increasing the accuracy of the result significantly. These models are known as base learners.

On the basis of generation of base learners, ensemble learning is divided into two categories: Sequential and Parallel ensemble learning. In sequential ensemble learning, the base learners are generated in sequential flow. In this category, all the base learners are dependent on each other to complete the task. If the performance of sequential ensemble learning is weak then we need to assign higher weights to earlier misrepresented learners.

In parallel ensemble learning, all the base learners are generated parallel to each other. This mechanism promotes independence among base learners. The independence among the base learners increases the accuracy of the system and reduces the computation time of system.

There are two type of ensemble - homogenous and heterogeneous. In homogenous ensemble, all the base learners use the same algorithm and same training data for computing the result while in case of heterogeneous, all the base learners will not use the same algorithm and a subset of training data is supplied to each other for computation which spreads heterogeneous in the environment, but almost every time homogenous ensemble is used by the developers.

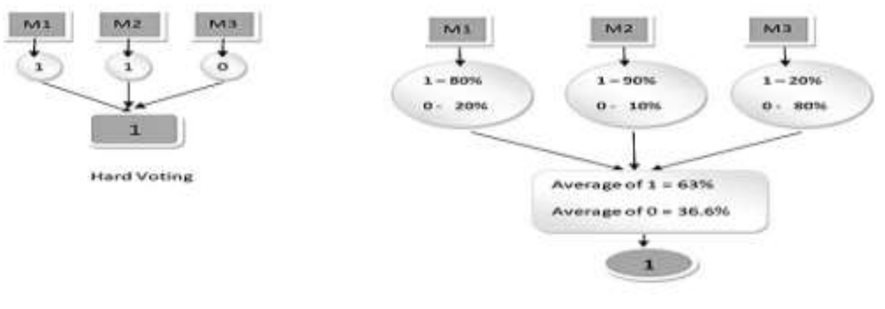
3.4.1 Voting Classifiers

Voting classifier is a machine learning model which is generally used to enhance the accuracy of the model with the help of several model. It is an ensemble learning models which will take the output from different models and accordingly give the output. It can be used for classification as well as regression. If the case is of regression, then this models will predict the output after taking average from all the models output. In classification case, it chooses the output on the basis of maximum voting or occurrence of the result.

It is classified in two different categories:

Hard Voting: In hard voting, it will predict the output on the basis of maximum votes received by the result.

Soft Voting: In soft voting, it will predict the value on the basis of average probability value. Assume that, given certain input to three models, the prediction probabilities for classes M and N are 0.36, 0.43, and 0.52, respectively. Because class M's average is 0.4366 and class N's is 0.14, class M is clearly the winner.



(Image by Author), Left: Hard Voting, Right: Soft Voting

Fig. 7

It is not guaranteed that the voting ensemble will do better than any one of the ensemble's individual models. A model should likely be used instead of the voting ensemble if it performs better than any other model included in the ensemble. The opposite is not always true. In comparison to individual models, a voting ensemble can provide forecasts with smaller variance. A decreased variance in prediction error for regression tasks demonstrates this. The accuracy variance for classification tasks is also less, which is another indication of this. Given the increased stability or confidence of the model, a lower mean performance of the ensemble could be desired. However, this reduced variance may cause it.

3.4.2 Adaboost Classification

The AdaBoost algorithm, also known as adaptive boosting, is a boosting approach that is used in machine learning as an ensemble method. As the weights are reallocated to each instance, greater weights are assigned to instances that were incorrectly classified, thus the term "adaptive boosting." Boosting is used in supervised learning to reduce bias and variation. It is based on the idea that students progress in stages. Except for the first, each student following the first is formed from earlier learners. Simply put, weak students become strong. The AdaBoost method works on the same basic principle as boosting, with a minor difference.

The goal is to combine numerous weak classifiers into one powerful classifier. A single classifier may not be able to predict an object's class with sufficient accuracy, but we can develop one such strong model by aggregating numerous weak classifiers and gradually learning from the mistakenly classified items of the others. Any of your basic classifiers, such as Decision Trees (which are typically the default), Logistic Regression, and so on, might be the one listed above. A good classifier assigns classes to objects more reliably than random guessing but still performs poorly overall. A weak classifier, for example, would presume that people under 45 can ingest sugar but those over 45 cannot. Even if your accuracy rate improved to more than 55%, you would still misclassify a huge percentage of data points.

3.4.3 Bagging Classifier

To lessen variance within a noisy dataset, ensemble learning techniques like bagging, often referred to as bootstrap aggregation, are frequently used. In this process, different base classifiers are trained parallel to each other, so as to increase the speed of the model and also to reduce the dependency between the base models. We trained the base classifiers with a sample of training data. Each of the fundamental classifiers has its own unique training set. The final training set might include some of the original data again while omitting others. With the help of voting, bagging retards overfitting. This increases bias, but the decrease in variance more than makes up for it.

To enhance the system's overall performance, bagging combines the results of various models. The Bagging Classifier is an ensemble technique that creates numerous distinct subsets of the training data via bootstrap resampling, and then trains a new model on each subset. All of the models' forecasts are combined to produce the final results.
By averaging the results of various models, one of Bagging's key advantages is its ability to lower the variance of the underlying model. As the models are trained on various subsets of the data, it can also help to reduce overfitting by lowering the correlation between the models.

Decision trees, neural networks, and linear models are just a few examples of the many diverse base models that can be employed with the Bagging classifier, a general-purpose ensemble technique. It is a simple, efficient way for enhancing the performance of a single model.

Any base classifier with large variance, such decision tree classifiers, can be enhanced with the Bagging classifier. With the exception of the number of estimators and the bootstrap parameter, the Bagging classifier can be used in the same way as the Base classifier.

3.5 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a class of deep learning algorithms that excel at processing and recognizing images. It is constructed of a number of layers, including convolutional, pooling, and fully linked layers.

The fundamental elements in a CNN are its layers which help the model to classify the object accurately. It consists of three different layers which perform different task and all the layers are connected to each other. The first layer is convolution layer, which is responsible for all the arithmetic with the image. It basically contains a kernel/filter which runs over the entire image to extract out the relevant features from the image. It constructs a feature map matrix with the help of dot product between the pictures pixels and the kernel pixels. With each successive layer, the complexity and responsibilities of kernels increases to correctly fetch the important features from the image. In this layer basically the image is converted into numerical form. The next layer is a pooling layer which is mainly responsible for dimensionality reduction without affecting the important features. It helps the model to enhance its accuracy by reducing the number of features. It basically does so by two techniques - Max pooling technique and Average pooling technique. In these techniques, a filter run over the images and select the maximum value in the max pooling technique and Average value in case of Average pooling technique. So this layer, reduce the computation of the model. The last layer is Fully Connected layer in which the final classification is done. It produces the output with the help of dense network among which all the layers are connected to its next layer. This layer produces a score for each class and the class with the maximum score is selected as an output class for the input image.

A sizable collection of labeled images is used to train CNNs, which then use this data to identify patterns and features that are related to particular objects or classes. Once trained, the CNN can easily categorize the image into desired category based on their features.

Different image validation strategy, such as object categorization, object detection, and image segmentation, have been successfully completed with CNNs at the cutting edge of technology. They are widely utilized in the disciplines of CV, image processing, etc., and is quite helpful in the field of surveillance, driverless motors and medical imaging.

Whether they be lines, gradients, circles, eyes, or even faces and eyes, CNN does a pretty good job of recognizing design in the input image. Convolutional neural networks for computer vision are extremely reliable because of this feature. Preprocessing is not required when working with CNN because it can run immediately on an underdone image. A feed forward neural network with as many as 20 nodes is known as a convolutional neural network. Each convolutional layer in CNN's system is capable of recognizing more complex shapes and is stacked on top of the others. Handwritten digit recognition can be accomplished with three or four convolutional layers, and facial recognition can be accomplished with 25 layers. The target of this field is to train the machine to make its decision making capability as comparative to human beings. As human beings is capable of taking decisions in any circumstances we want our machine to be that much efficient.

In essence, CNN's architecture is a series of layers that convert a 3-dimensional picture volume—that is, one with width, height, and depth—into a 3-dimensional output volume. The fact that each neuron in the current layer is connected to a small piece of the output from the previous layer, which is analogous to applying an N*N filter to the input image, is a key thing to keep in mind in this situation.

It employs M filters, which are essentially feature extractors that pull out features like edges, corners, and so on. Convolutional neural networks (CNNs) are built using the following layers: [INPUT-CONV-RELU-POOL-FC] –

- **INPUT**− It is a layer that stores the unprocessed pixel values or accepts the input image from the user.

- **CONV**− is responsible for all the arithmetic with the image. It basically extract out the relevant features from the image

- **RELU**− this layer adds an activation function to the output of the prior layer.

- **POOL**− the primary function of this layer's is down sampling which reduces the dimensionality without losing any important features.

- **FC**− in this layer, the final classification is done.

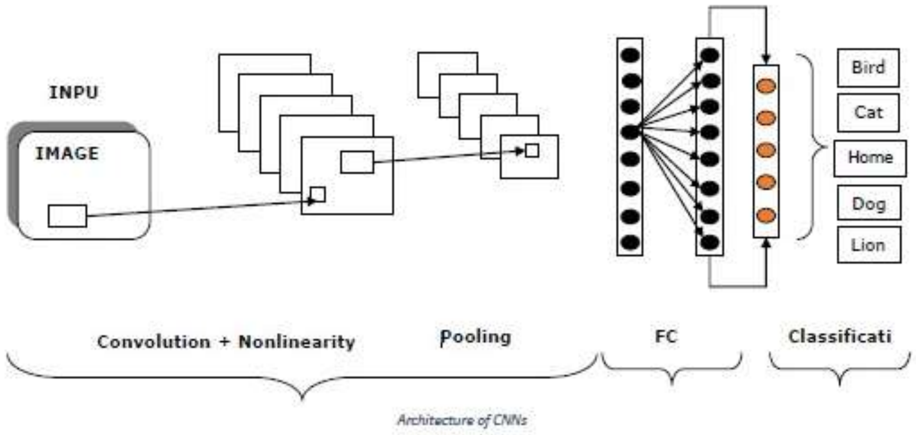

INPU
IMAGE
Bird
Cat
Home
Dog
Lion
Convolution + Nonlinearity
Pooling
FC
Classificati
Architecture of CNNs

Fig. 8

Chapter 4

Result

In this project, I have integrated two different datasets namely structural MRI and resting state functional MRI. As an input to the model for our study, we used MRI pictures. The dataset sample is shown below.:

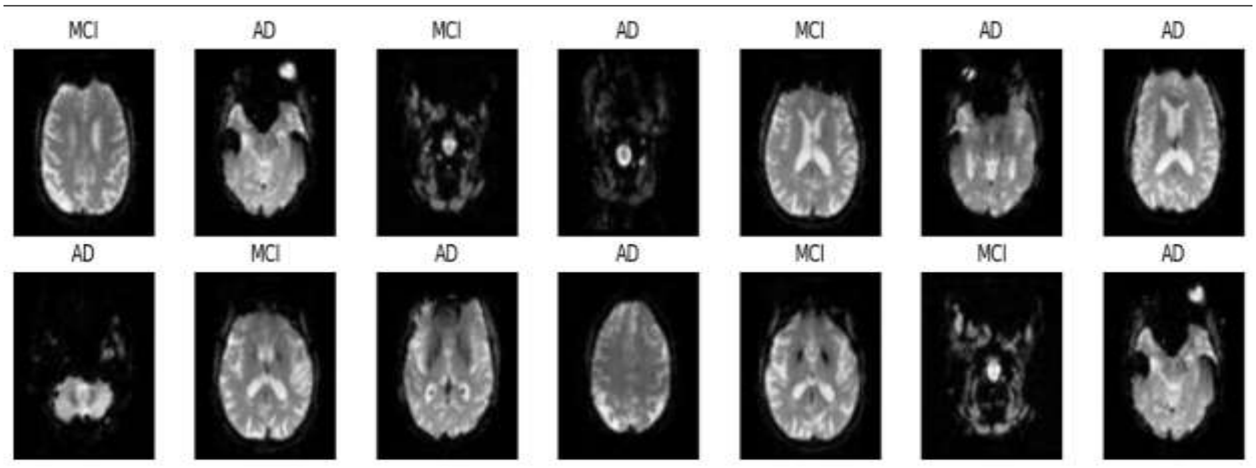
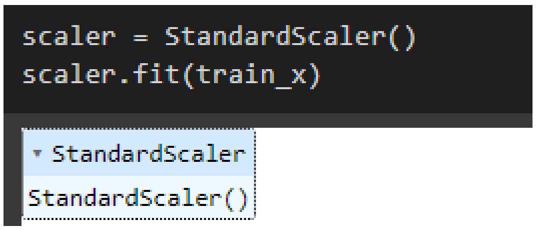


Fig . 9

First of all, we have extracted features from the MRI images using pre-trained model. We have used VGG16 model for feature extraction with an input shape of 224x224x3. This model trained total 134,260,544 parameters. The dataset is split into training and testing data with respective weights of 80% and 20%.. The model converts the image into numeric array values and reshapes the image into 224x224. Since not all numerical values fall into a specific range, we employed the Standard Scaler method to standardise the feature value.

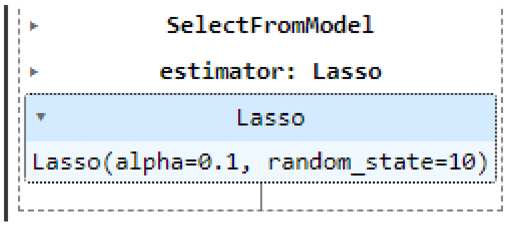


```
scaler = StandardScaler()
scaler.fit(train_x)
```

```
▾ StandardScaler
StandardScaler()
```

Fig .10

We are aware that these numerous features don't work with machine learning techniques, so to reduce the features or select the appropriate features, we have used LASSO feature selection algorithm with alpha value of 0.1 and random state of 10. It makes use of shrinking. Shrinkage is the process by which data values are shrunk towards a central point as the mean.



```
SelectFromModel
estimator: Lasso
Lasso
Lasso(alpha=0.1, random_state=10)
```

Fig .11

To produce the prognostic prediction from MCI to AD, many machine learning techniques have been applied. Apart from machine learning, we have also used a deep learning approach to do the prediction. Support Vector Machine (SVM), Random Forest algorithm, Logistic Regression, ensemble methods including Voting classification, Adaboost classification, bagging, and Convolution Neural Network (CNN) are the algorithms utilised in this procedure.

A confusion matrix is a table that lists how many values are predicted correctly and how many are predicted wrongly. It is employed to evaluate a classification model's effectiveness. The accuracy of the SVM algorithm using the linear kernel is 85.60% on the training data and 85.93% on the testing data. The linear kernel SVM's confusion matrix is

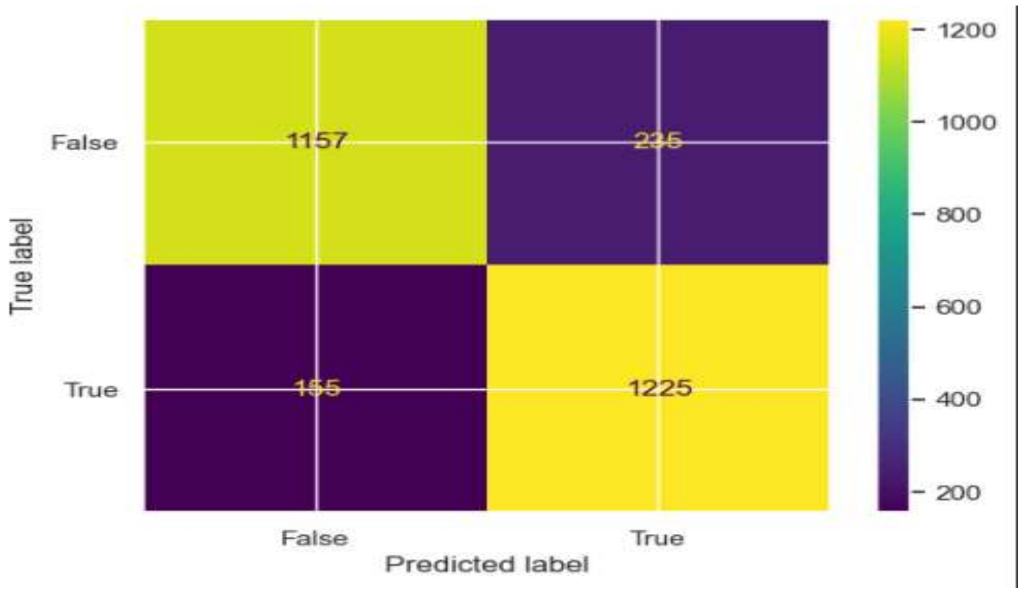


Fig .12

It depicts that the TP value predicted by model is 1225, FN value is 155, FP value is 235 and FN value is 1157. As TP and TN values are high as compared to FN and FP, so we consider it as a good model.

The degree 2 polynomial kernel used in the SVM technique gives an accuracy of 89.90% on training data and 90.94% on testing data. The polynomial kernel SVM's confusion matrix is

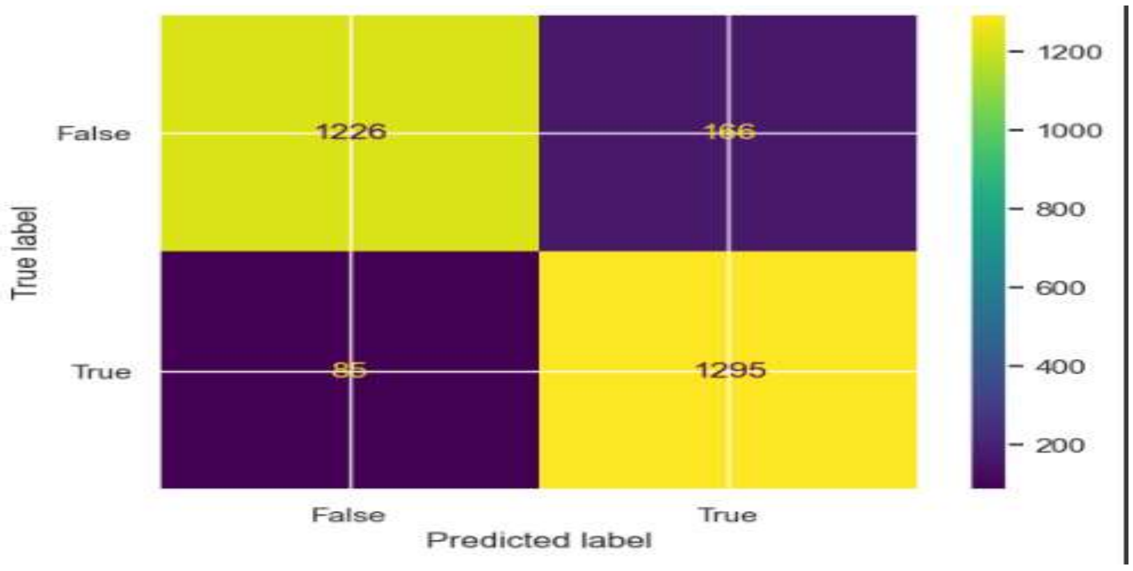


Fig .13

It depicts that the TP value predicted by model is 1295, FN value is 85, FP value is 166 and FN value is 1226. As TP and TN values are high as compared to FN and FP, so we consider it as a good model.

The rbf kernel used in the SVM technique an accuracy of 93.92% on training data and 94.40% on testing data. SVM's confusion matrix with rbf kernel is

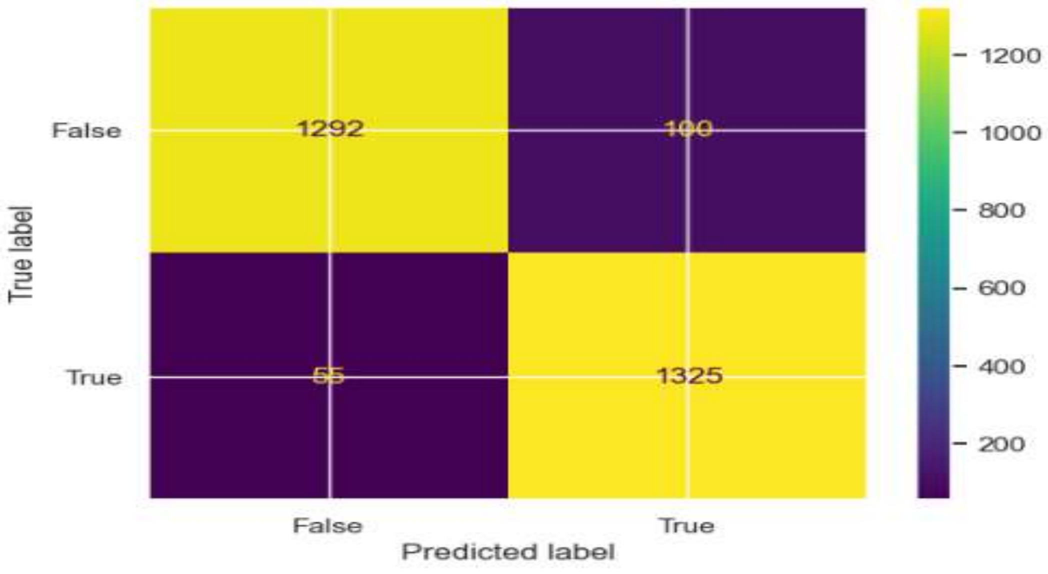


Fig .14

It depicts that the TP value predicted by model is 1325, FN value is 55, FP value is 100 and FN value is 1292. As TP and TN values are high as compared to FN and FP, so we consider it as a good model.

SVM		Accuracy	
		Lasso Algorithm	
S.No	**Kernel**	**Training data**	**Testing Data**
1	**linear**	0.8560	0.8593
2	**polynomial (degree 2)**	0.8995	0.9094
3	**rbf**	0.9392	0.9440

Table 2

The random forest with 100 number of estimators gives an accuracy of 100% on training data and 94.40% on testing data. The confusion matrix of Random Forest with 100 number of estimator is

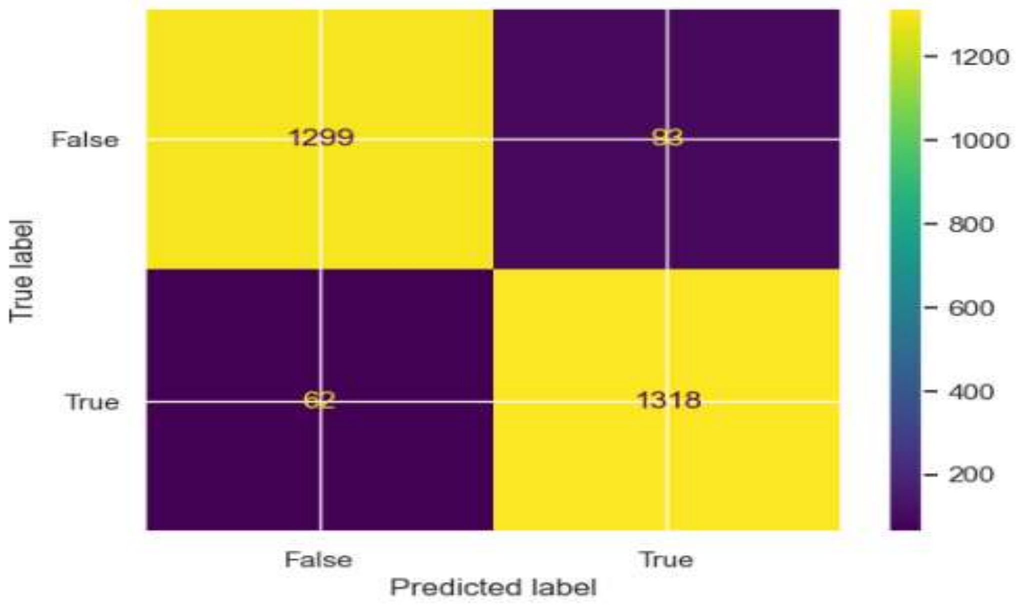


Fig .15

It depicts that the TP value predicted by model is 1318, FN value is 62, FP value is 83 and FN value is 1299. As TP and TN values are high as compared to FN and FP, so we consider it as a good model but this model faces over-fitting.

The random forest with 1000 number of estimators gives an accuracy of 100% on training data and 94.58% on testing data. The confusion matrix of Random Forest with 1000 number of estimator is

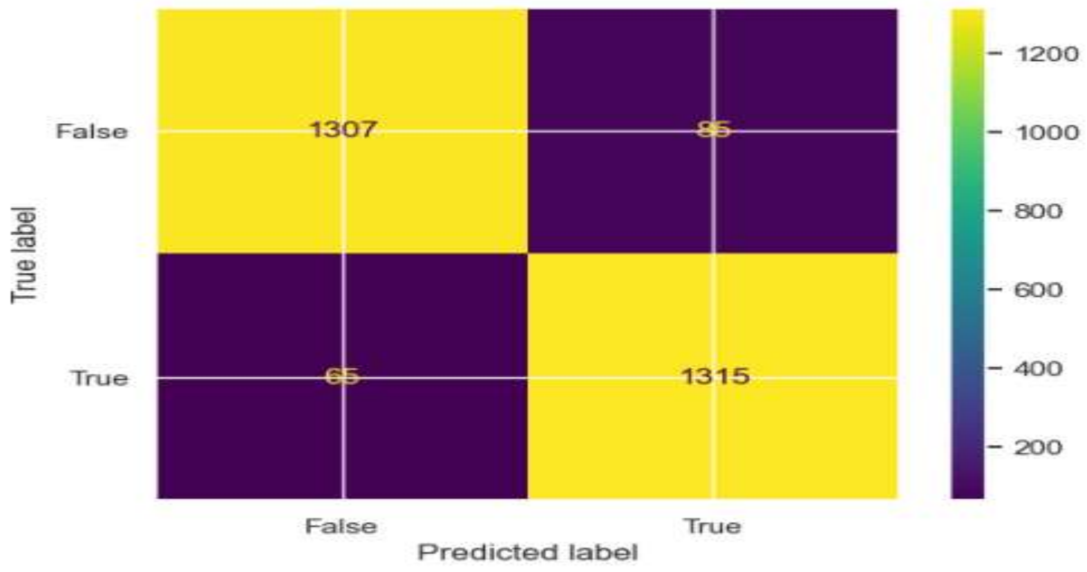


Fig .16

It depicts that the TP value predicted by model is 1315, FN value is 65, FP value is 85 and FN value is 1307. As TP and TN values are high as compared to FN and FP, so we consider it as a good model but this model faces over-fitting.

The random forest with 5000 number of estimators gives an accuracy of 100% on training data and 94.48% on testing data. The confusion matrix of Random Forest with 5000 number of estimator is

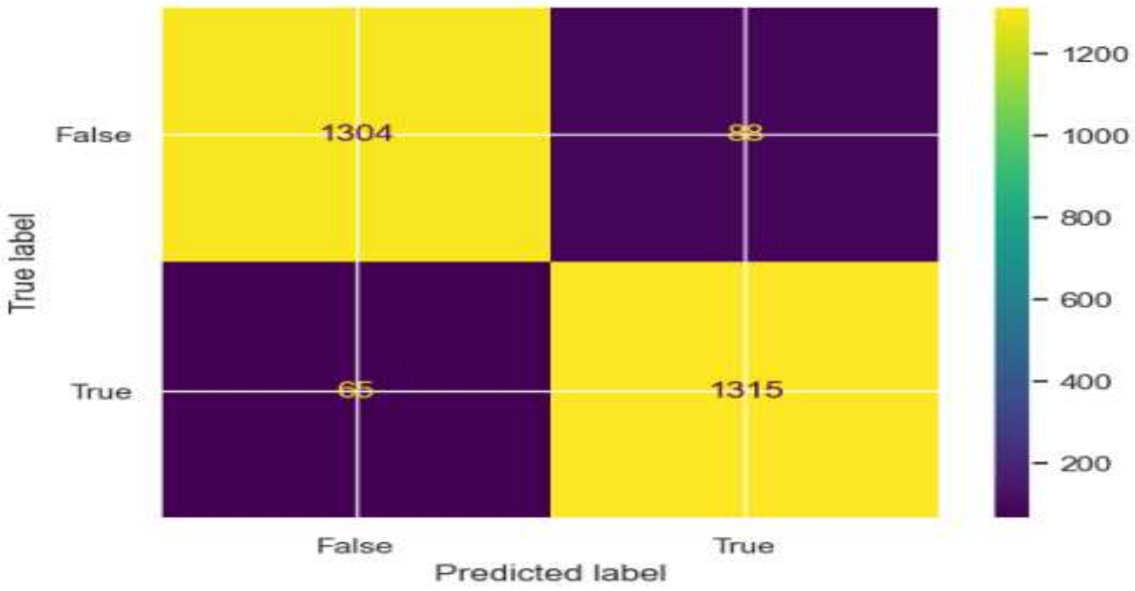


Fig .17

It depicts that the TP value predicted by model is 1315, FN value is 65, FP value is 88 and FN value is 1304. As TP and TN values are high as compared to FN and FP, so we consider it as a good model but this model faces over-fitting.

RANDOM FOREST		Accuracy	
		Lasso Algorithm	
S.No	No. of estimator	Training data	Testing Data
1	**100**	1.0	0.9440
2	**1000**	1.0	0.9459
3	**5000**	1.0	0.9449

Table 3

The following model, logistic regression, performs the best and provides 100% accuracy on both training and test data. The Logistic Regression's confusion matrix is

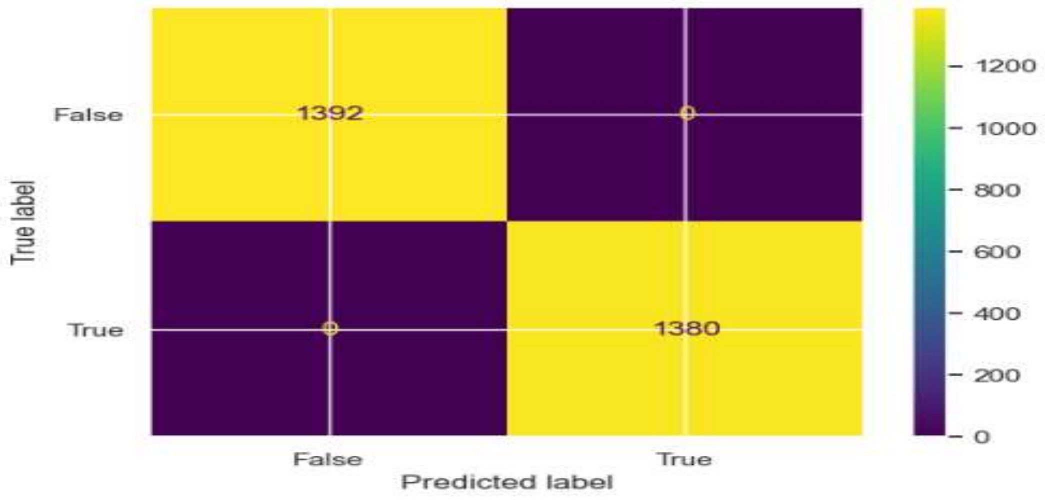


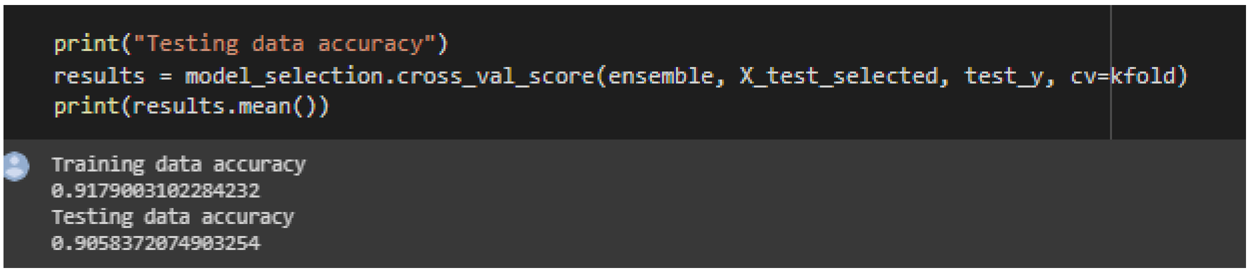
Fig .18

It depicts that the TP value predicted by model is 1380, FN value is 0, FP value is 0 and FN value is 1392. As TP and TN values are high as compared to FN and FP, so we consider it as a good model.

LOGISTIC REGRESSION			
		Accuracy	
S.No	Algorithms	Training Data	Testing Data
1	LASSO	1.0	1.0

Table 4

Now, we will discuss about the performance of ensemble method. We have totally used three different ensemble methods. Voting ensemble classifier is the first one. A voting classifier consists of a variety of models that are combined and trained to produce high accuracy, which predicts the output class with the highest probability of being chosen. Here this model predicts among the two classes : MCI or AD. This algorithm gives an accuracy of 91.79% accuracy on the training data and 90.58% on testing data. This ensemble method shows over-fitting as the model perform best over the training data as compared to testing data.

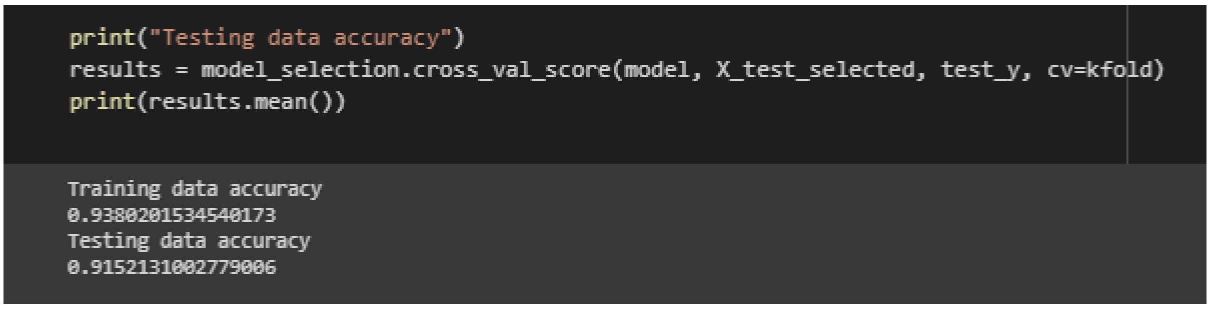


```
print("Testing data accuracy")
results = model_selection.cross_val_score(ensemble, X_test_selected, test_y, cv=kfold)
print(results.mean())
```

```
Training data accuracy
0.9179003102284232
Testing data accuracy
0.9058372074903254
```

Fig .19

The second ensemble method is bagging.Bagging is the process of selecting with replacement a random sample of data from a training batch. These weak models are then trained individually based on the type of task—regression or classification—after producing several data samples. In our case, these weak learners are trained to perform classification task. Bagging reached a level of accuracy of 93.80% on training data and 91.52% on testing data. This model also gives us an over-fitting result.

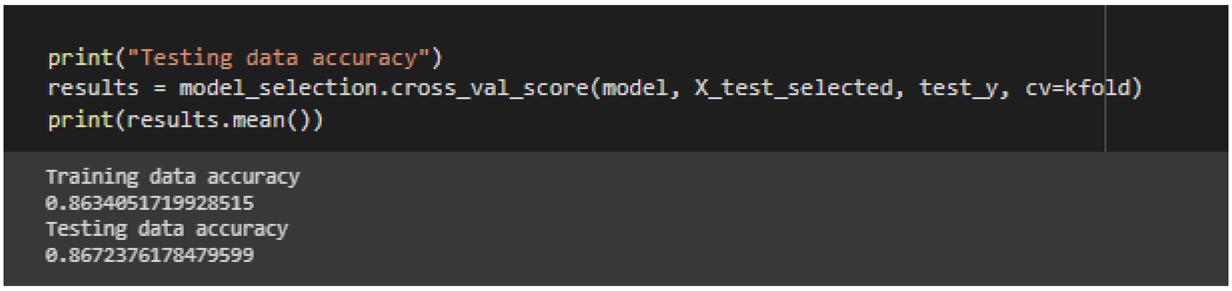


```
print("Testing data accuracy")
results = model_selection.cross_val_score(model, X_test_selected, test_y, cv=kfold)
print(results.mean())
```

```
Training data accuracy
0.9380201534540173
Testing data accuracy
0.9152131002779006
```

Fig .20

The last ensemble model is Adaboost, which gives an accuracy of 86.34% on training data and 86.73% on testing data with 70 numbers of estimators with a random state of 10. Among the given machine learning algorithms, we found that the logistic regression performs best with complete accuracy on both testing and training data with LASSO feature selection algorithm.



```
print("Testing data accuracy")
results = model_selection.cross_val_score(model, X_test_selected, test_y, cv=kfold)
print(results.mean())
```

```
Training data accuracy
0.8634051719928515
Testing data accuracy
0.8672376178479599
```

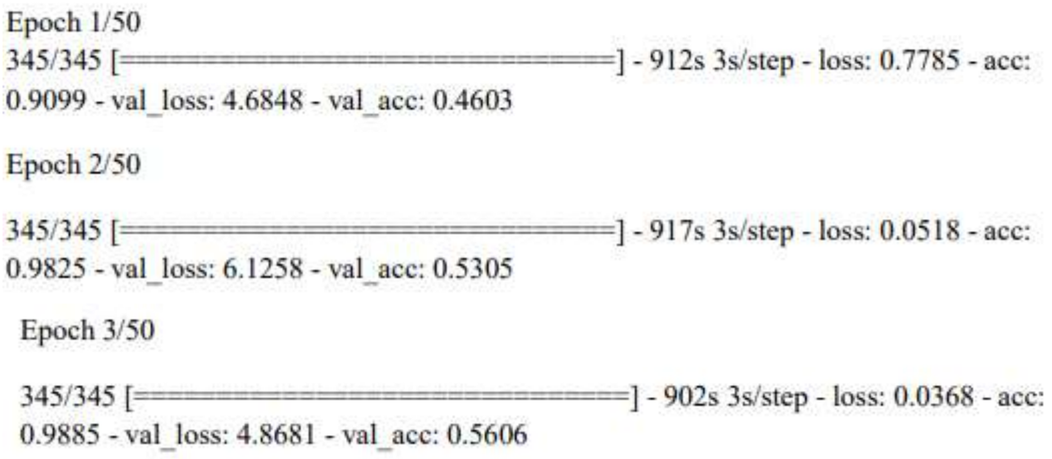
Fig .21

Utilizing the Adam optimizer and a categorical cross-entropy loss function, the neural network is built. As a result, this loss function is employed since it is the most often used one in the multiclass classification work. In the meanwhile, I select Adam as the optimizer because it simply minimizes loss value in the majority of neural network tasks.

ENSEMBLE METHOD		Accuracy	
		Lasso Algorithm	
S.No	**Ensemble Method**	**Training data**	**Testing Data**
1	**Bagging**	0.9380	0.9152
2	**Adaboost**	0.8634	0.8672
3	**Voting**	0.9179	0.9058

Table 5

We trained Convolution neural network with 50 different epochs and each epochs contains 345 iterations. Additionally, the generator creates unique images for each epoch, which is very beneficial for our neural network classifier because it allows it to more effectively generalize samples in the test set. And therefore, the training procedure is described below.



```
Epoch 1/50
345/345 [==============================] - 912s 3s/step - loss: 0.7785 - acc:
0.9099 - val_loss: 4.6848 - val_acc: 0.4603

Epoch 2/50

345/345 [==============================] - 917s 3s/step - loss: 0.0518 - acc:
0.9825 - val_loss: 6.1258 - val_acc: 0.5305

Epoch 3/50

345/345 [==============================] - 902s 3s/step - loss: 0.0368 - acc:
0.9885 - val_loss: 4.8681 - val_acc: 0.5606
```

```
Epoch 4/50

345/345 [==============================] - 892s 3s/step - loss: 0.0178 - acc:
0.9943 - val_loss: 6.8855 - val_acc: 0.5556

Epoch 5/50

345/345 [==============================] - 889s 3s/step - loss: 0.0225 - acc:
0.9915 - val_loss: 5.6141 - val_acc: 0.5578

Epoch 6/50

345/345 [==============================] - 895s 3s/step - loss: 0.0134 - acc:
0.9950 - val_loss: 5.2381 - val_acc: 0.5773

Epoch 7/50

345/345 [==============================] - 882s 3s/step - loss: 0.0292 - acc:
0.9908 - val_loss: 4.5382 - val_acc: 0.5673

Epoch 8/50

345/345 [==============================] - 886s 3s/step - loss: 0.0167 - acc:
0.9946 - val_loss: 4.6579 - val_acc: 0.5138

Epoch 9/50

345/345 [==============================] - 887s 3s/step - loss: 0.0140 - acc:
0.9956 - val_loss: 5.9067 - val_acc: 0.5241

Epoch 10/50

345/345 [==============================] - 887s 3s/step - loss: 0.0054 - acc:
0.9979 - val_loss: 5.9841 - val_acc: 0.6063

Epoch 11/50

345/345 [==============================] - 886s 3s/step - loss: 0.0209 - acc:
0.9944 - val_loss: 5.1235 - val_acc: 0.5234

Epoch 12/50

345/345 [==============================] - 891s 3s/step - loss: 0.0109 - acc:
0.9961 - val_loss: 3.5280 - val_acc: 0.6063

Epoch 13/50

345/345 [==============================] - 885s 3s/step - loss: 0.0104 - acc:
0.9969 - val_loss: 4.7842 - val_acc: 0.6421
```

```
Epoch 14/50

345/345 [==============================] - 889s 3s/step - loss: 0.0104 - acc:
0.9973 - val_loss: 7.0409 - val_acc: 0.5748

Epoch 15/50

345/345 [==============================] - 887s 3s/step - loss: 0.0352 - acc:
0.9898 - val_loss: 4.7940 - val_acc: 0.4653

Epoch 16/50

345/345 [==============================] - 881s 3s/step - loss: 0.0352 - acc:
0.9897 - val_loss: 5.0822 - val_acc: 0.5652

Epoch 17/50

345/345 [==============================] - 877s 3s/step - loss: 0.0076 - acc:
0.9977 - val_loss: 4.5629 - val_acc: 0.5974

Epoch 18/50

345/345 [==============================] - 878s 3s/step - loss: 0.0078 - acc:
0.9976 - val_loss: 5.9212 - val_acc: 0.5602

Epoch 19/50

345/345 [==============================] - 879s 3s/step - loss: 0.0043 - acc:
0.9985 - val_loss: 5.9163 - val_acc: 0.5397

Epoch 20/50

345/345 [==============================] - 885s 3s/step - loss: 0.0033 - acc:
0.9988 - val_loss: 8.2835 - val_acc: 0.5581

Epoch 21/50

345/345 [==============================] - 882s 3s/step - loss: 0.0129 - acc:
0.9963 - val_loss: 6.9717 - val_acc: 0.5815

Epoch 22/50

345/345 [==============================] - 882s 3s/step - loss: 0.0077 - acc:
0.9982 - val_loss: 4.5308 - val_acc: 0.6134

Epoch 23/50

345/345 [==============================] - 889s 3s/step - loss: 0.0095 - acc:
0.9973 - val_loss: 4.0008 - val_acc: 0.5018

Epoch 24/50

345/345 [==============================] - 884s 3s/step - loss: 0.0056 - acc:
0.9982 - val_loss: 3.2792 - val_acc: 0.4908
```

```
Epoch 25/50

345/345 [==============================] - 891s 3s/step - loss: 0.0089 - acc:
0.9975 - val_loss: 5.5279 - val_acc: 0.5889

Epoch 26/50

345/345 [==============================] - 889s 3s/step - loss: 0.0036 - acc:
0.9989 - val_loss: 6.0065 - val_acc: 0.5485

Epoch 27/50

345/345 [==============================] - 889s 3s/step - loss: 0.0038 - acc:
0.9985 - val_loss: 5.4206 - val_acc: 0.4901

Epoch 28/50

345/345 [==============================] - 898s 3s/step - loss: 0.0068 - acc:
0.9980 - val_loss: 6.7133 - val_acc: 0.5659

Epoch 29/50

345/345 [==============================] - 890s 3s/step - loss: 0.0113 - acc:
0.9963 - val_loss: 5.7399 - val_acc: 0.5971

Epoch 30/50

345/345 [==============================] - 888s 3s/step - loss: 0.0024 - acc:
0.9993 - val_loss: 7.6576 - val_acc: 0.5149


Epoch 31/50

345/345 [==============================] - 888s 3s/step - loss: 0.0075 - acc:
0.9978 - val_loss: 5.6439 - val_acc: 0.5868

Epoch 32/50

345/345 [==============================] - 886s 3s/step - loss: 0.0030 - acc:
0.9991 - val_loss: 6.1503 - val_acc: 0.5216

Epoch 33/50

345/345 [==============================] - 887s 3s/step - loss: 0.0040 - acc:
0.9985 - val_loss: 5.6095 - val_acc: 0.5400

Epoch 34/50

345/345 [==============================] - 887s 3s/step - loss: 0.0021 - acc:
0.9995 - val_loss: 6.1325 - val_acc: 0.6775
```

```
Epoch 35/50

345/345 [==============================] - 888s 3s/step - loss: 0.0034 - acc:
0.9989 - val_loss: 7.3082 - val_acc: 0.5018

Epoch 36/50

345/345 [==============================] - 887s 3s/step - loss: 0.0033 - acc:
0.9993 - val_loss: 5.1301 - val_acc: 0.5773

Epoch 37/50

345/345 [==============================] - 889s 3s/step - loss: 0.0066 - acc:
0.9980 - val_loss: 9.3388 - val_acc: 0.6637

Epoch 38/50

345/345 [==============================] - 889s 3s/step - loss: 0.0121 - acc:
0.9965 - val_loss: 7.4489 - val_acc: 0.4901

Epoch 39/50

345/345 [==============================] - 887s 3s/step - loss: 0.0037 - acc:
0.9990 - val_loss: 8.5773 - val_acc: 0.5578

Epoch 40/50

345/345 [==============================] - 886s 3s/step - loss: 0.0059 - acc:
0.9985 - val_loss: 5.3746 - val_acc: 0.5358

Epoch 41/50

345/345 [==============================] - 884s 3s/step - loss: 0.0057 - acc:
0.9987 - val_loss: 6.3553 - val_acc: 0.6368

Epoch 42/50

345/345 [==============================] - 896s 3s/step - loss: 0.0026 - acc:
0.9995 - val_loss: 9.3820 - val_acc: 0.6651

Epoch 43/50

345/345 [==============================] - 900s 3s/step - loss: 9.5994e-04 - acc:
0.9997 - val_loss: 8.4869 - val_acc: 0.5985

Epoch 44/50

345/345 [==============================] - 890s 3s/step - loss: 1.2167e-04 - acc:
1.0000 - val_loss: 8.4374 - val_acc: 0.5974
```

```
Epoch 45/50

345/345 [==============================] - 891s 3s/step - loss: 0.0063 - acc:
0.9981 - val_loss: 6.5842 - val_acc: 0.6006

Epoch 46/50

345/345 [==============================] - 889s 3s/step - loss: 0.0012 - acc:
0.9995 - val_loss: 8.3901 - val_acc: 0.6580

Epoch 47/50

345/345 [==============================] - 890s 3s/step - loss: 0.0034 - acc:
0.9993 - val_loss: 7.4001 - val_acc: 0.6655

Epoch 48/50

345/345 [==============================] - 889s 3s/step - loss: 2.5880e-04 - acc:
1.0000 - val_loss: 7.6939 - val_acc: 0.5861

Epoch 49/50

345/345 [==============================] - 883s 3s/step - loss: 0.0035 - acc:
0.9989 - val_loss: 10.9411 - val_acc: 0.6339

Epoch 50/50

345/345 [==============================] - 891s 3s/step - loss: 0.0059 - acc:
0.9986 - val_loss: 7.4194 - val_acc: 0.6095
```

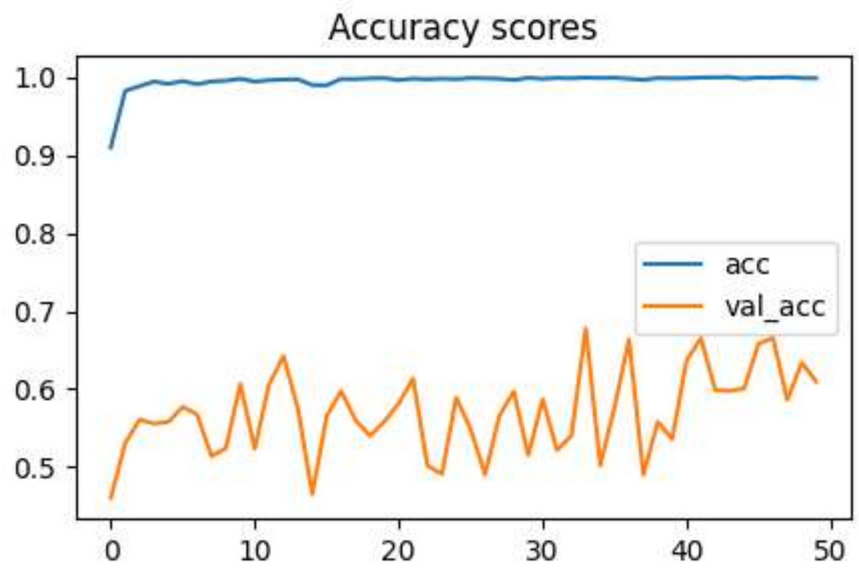


Fig .22

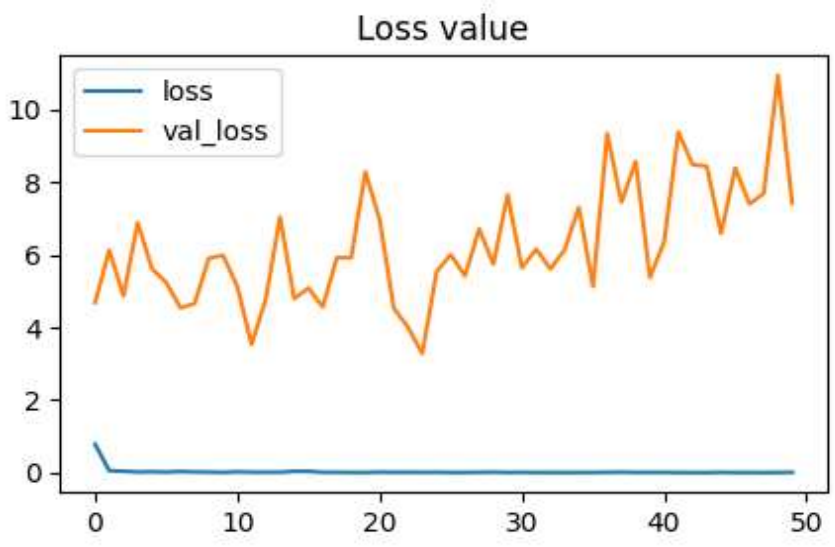
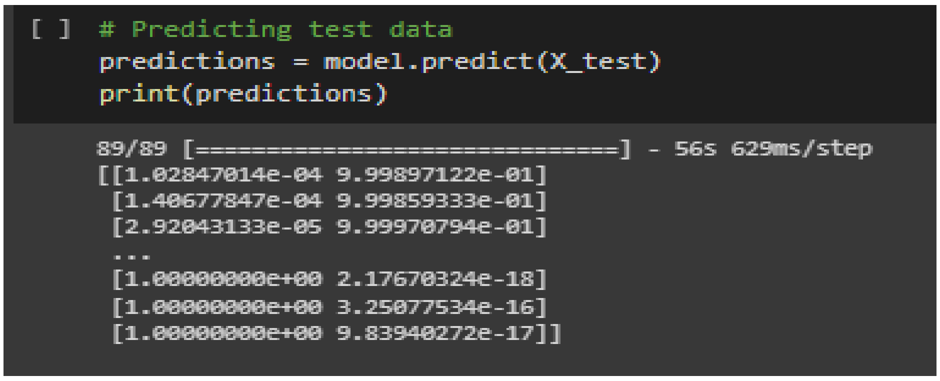


Fig .23

Both the testing accuracy and loss value appear to be shifting within these 50 epochs, but based on the two figures above, we may conclude that the model's performance is continuing to improve.

Here, are the prediction result of the test data which predicts the result into three classes – MCI, Progression to AD and AD,



```
[ ]  # Predicting test data
     predictions = model.predict(X_test)
     print(predictions)

     89/89 [==============================] - 56s 629ms/step
     [[1.02847014e-04 9.99897122e-01]
      [1.40677847e-04 9.99859333e-01]
      [2.92043133e-05 9.99970794e-01]
      ...
      [1.00000000e+00 2.17670324e-18]
      [1.00000000e+00 3.25077534e-16]
      [1.00000000e+00 9.83940272e-17]]
```

Fig .24

We can observe that the predictions result array contains 2 columns where the first columns represents the probability of person belonging to AD class and the second column represents the probability of person belonging to MCI class. If the value lies between the 0.5 to 0.7 ranges then we can classify it as the person belonging to Progression to AD class.

Chapter 5

Conclusion and Future Work

The center of attention of this study was to develop and evaluate a machine learning and deep learning technique for comparing and integrating rs-fMRI and sMRI data for MCI and AD categorization as well as to determine whether a patient is transitioning from MCI to AD. We developed seven different classifiers for doing the task by considering the sMRI features and rs-fMRI features. We use VGG16 pre-trained model for feature extraction from both the datasets. The patient was then accurately classified into the required group using the Least Absolute Shrinkage and Selection Operator (LASSO) technique, which I used to choose the best small subset of features beneficial for each classifier. The classifiers underwent training, cross-validation, and performance evaluation on the characteristics chosen by the method. We discovered that Logistic Regression performs best in the machine learning category with an accuracy of 1.0, while CNN performs best in deep learning with an accuracy of 0.9986 in the 50th epoch. When compared to a single modality strategy, the experiment results showed that integrating sMRI and rs-fMRI data is a reliable method for prognostic prediction of AD.

According to the literature study, numerous attempts have been built to identify Alzheimer's disease using different machine learning algorithms and microsimulation techniques; nonetheless, determining the precise level of disease remains a difficult challenge. For subsequent research, we need to expand our understanding using a more advanced EDA procedure with a larger sample size.

Appendices

1. Sample Code

```
# Load the model
mod = VGG16(weights='imagenet')

# Restructure the model to obtain features from second-to-last layer
mod = Model(inputs=mod.inputs, outputs=mod.layers[-2].output)

# Print model summary
print(mod.summary())
```

```
Count= 0
b=0
Directory = './MCI_2/'
for image in os.listdir(Directory):
    image_path = Directory+image

    # load the image
    Imag1 = load_img(image_path, target_size=(224, 224))
    # convert pixel to numpy array
    Imag1 = img_to_array (Imag1)
    # reshape the image for the model
    Imag1 = Imag1.reshape((1, Imag1.shape[0], Imag1.shape[1], Imag1.shape[2]))
    # preprocess the image
    Imag1 = preprocess_input(Imag1)
    # extract features
    Feature = model.predict(img, verbose=0)

    Features.append(Feature)
    Features_labels.append(b)
    Count+=1
    print(Count)

train_x, test_x, train_y, test_y = train_test_split(Features, Features_labels,
test_size = 0.2)
```

```
from sklearn.feature_selection import SelectFromModel
from sklearn. linear_model import Lasso

select = SelectFromModel(Lasso(alpha=0.1, random_state=10))
select.fit(scaler.transform(train_x), train_y)

X_train_selected = select.transform(scaler.transform(train_x))
X_test_selected = select.transform(scaler.transform(test_x))
```

```
from sklearn import metrics
from sklearn import svm

#train SVM model
lin = svm.SVC(kernel = 'linear')
lin.fit(X_train_selected, train_y)
#store predictions and ground truth
pred_y = lin.predict(X_train_selected)
true_y = train_y

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

#Now, use the SVM model to predict Test data
pred_y = lin.predict(X_test_selected)
true_y = test_y

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



confusion_matrix = metrics.confusion_matrix(true_y, pred_y)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```

```
#LASSO FEATURE SELECTION
# Voting Ensemble for Classification
```

```
K_fold = model_selection.KFold(n_splits=10)
# create the sub models
estimators = []
mod1 = LogisticRegression()
estimators.append(('logistic', mod1))
modl2 = DecisionTreeClassifier()
estimators.append(('cart', mod2))
mod3 = SVC()
estimators.append(('svm', mod3))
# create the ensemble model
ensembler = VotingClassifier(estimators)

print("Training data accuracy")
results = model_selection.cross_val_score(ensembler, X_train_selected, train_y, cv=
K_fold)
print(results.mean())

print("Testing data accuracy")
results = model_selection.cross_val_score(ensembler, X_test_selected, test_y, cv=
K_fold)
print(results.mean())
```

```
  #logistic regression
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings('ignore')
logistic_model =  LogisticRegression()
logistic_model.fit(train_x,train_y)
```

```
# CNN (Convolutional Neural Network)

input_layer = Input(shape=(X_train.shape[1], X_train.shape[2], 1))

convol1 = Conv2D(16, (3, 3), activation='relu', strides=(1, 1),
   padding='same')(input_layer)
convol2 = Conv2D(32, (3, 3), activation='relu', strides=(1, 1),
   padding='same')( convol2)
convol3= MaxPool2D((2, 2))( convol3)
```

```
convol4 = Conv2D(16, (2, 2), activation='relu', strides=(1, 1),
    padding='same')( convol4)
convol5 = Conv2D(32, (2, 2), activation='relu', strides=(1, 1),
    padding='same')( convol5)
convol6 = MaxPool2D((2, 2))( convol6)

convol7 = Flatten()(convol7)
convol8 = Dense(100, activation='relu')( convol8)
cnn = Dense(50, activation='relu')(cnn)
output_layer = Dense(2, activation='softmax')(cnn)

model = Model(inputs=input_layer, outputs=output_layer)
```

2. Sample Output

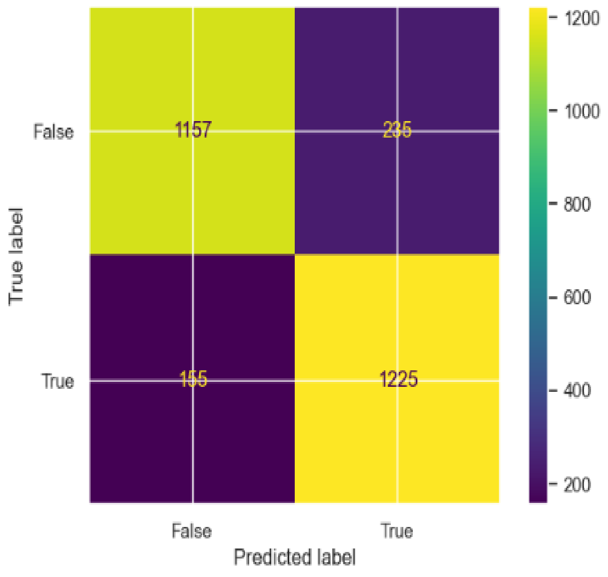
```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0         
                                                                 
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792      
                                                                 
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928     
                                                                 
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0         
                                                                 
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856     
                                                                 
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584    
                                                                 
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0         
                                                                 
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168    
                                                                 
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080    
                                                                 
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080    
                                                                 
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0         
                                                                 
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160   
                                                                 
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808   
```

```
(X_train_selected)
```

```
array([[ 2.10282295, -0.3075689 , -0.65747033, ...,  0.75004046,
        -1.18769758, -1.24870496],
       [-0.0572567 , -0.14269593, -0.03951568, ..., -0.51537198,
         0.10452011, -0.54243208],
       [-0.55829269, -0.3075689 , -0.53639832, ..., -0.51537198,
         0.85634754, -0.54214918],
       ...,
       [ 1.54692882, -0.3075689 ,  3.21750919, ...,  4.8741858 ,
        -1.3253065 ,  2.38811694],
       [-0.91742366, -0.3075689 , -0.65747033, ..., -0.51537198,
         0.42102103,  0.26927163],
       [-0.42761862, -0.3075689 ,  2.19129835, ..., -0.51537198,
        -1.3253065 , -1.35135589]])
```

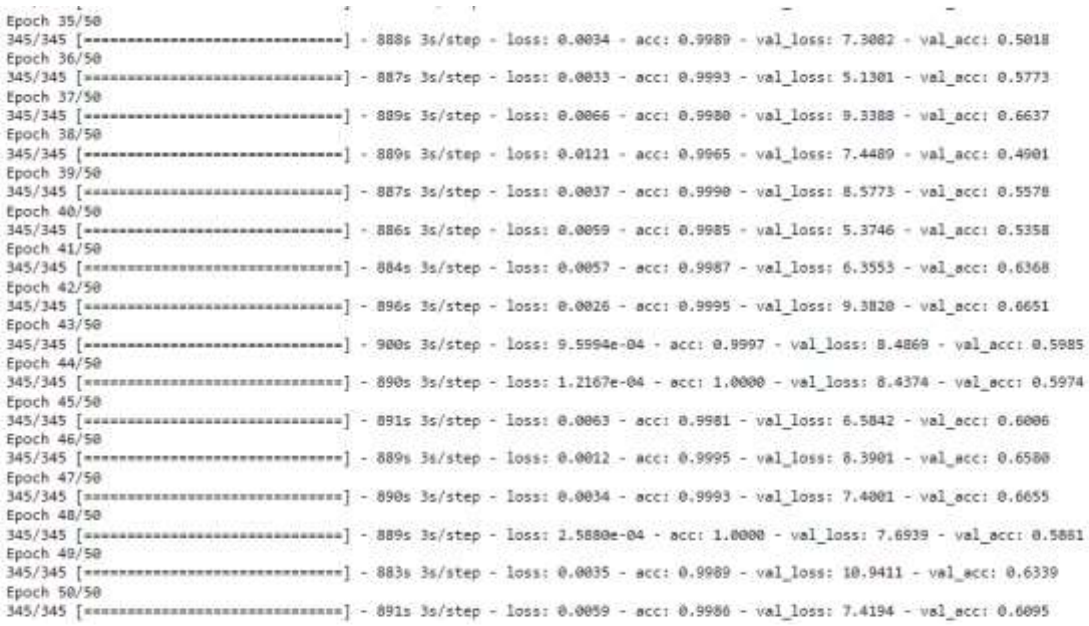
```
confusion_matrix = metrics.confusion_matrix(y_true, y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```

```
Accuracy :  0.8560988812702995
Accuracy :  0.8593073593073594
```



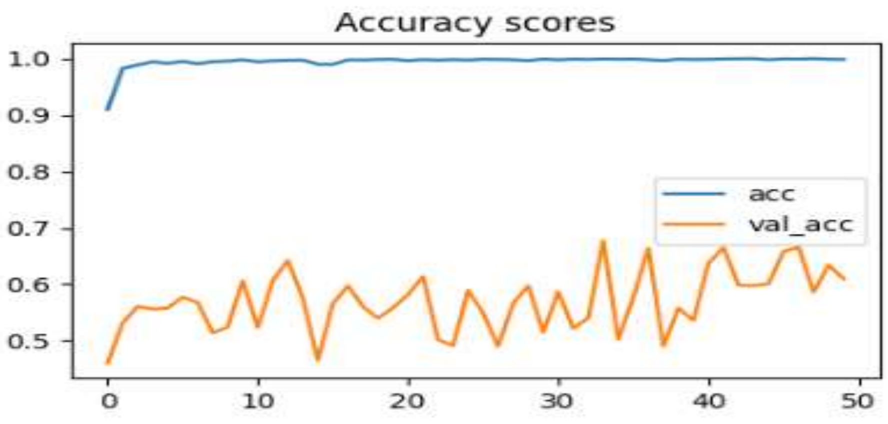
```
#fit the modelg

history = model.fit(train_gen, epochs=50,
          validation_data=(X_test, y_test_one_hot))
```

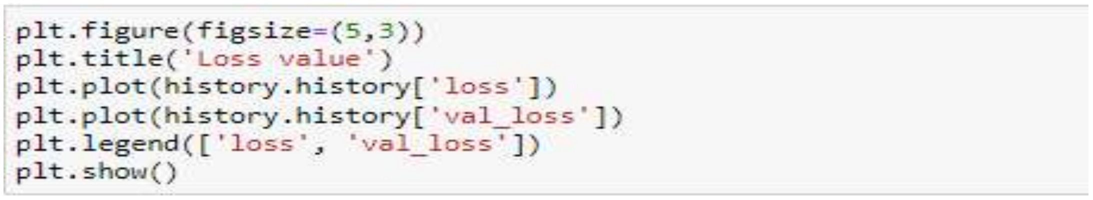
```
Epoch 1/50
345/345 [==============================] - 912s 3s/step - loss: 0.7785 - acc: 0.9099 - val_loss: 4.6848 - val_acc: 0.4603
Epoch 2/50
345/345 [==============================] - 917s 3s/step - loss: 0.0518 - acc: 0.9825 - val_loss: 6.1258 - val_acc: 0.5305
Epoch 3/50
345/345 [==============================] - 902s 3s/step - loss: 0.0368 - acc: 0.9885 - val_loss: 4.8681 - val_acc: 0.5606
Epoch 4/50
345/345 [==============================] - 892s 3s/step - loss: 0.0178 - acc: 0.9943 - val_loss: 6.8855 - val_acc: 0.5556
Epoch 5/50
345/345 [==============================] - 889s 3s/step - loss: 0.0225 - acc: 0.9915 - val_loss: 5.6141 - val_acc: 0.5578
Epoch 6/50
345/345 [==============================] - 895s 3s/step - loss: 0.0134 - acc: 0.9950 - val_loss: 5.2381 - val_acc: 0.5773
Epoch 7/50
345/345 [==============================] - 882s 3s/step - loss: 0.0292 - acc: 0.9908 - val_loss: 4.5382 - val_acc: 0.5673
Epoch 8/50
345/345 [==============================] - 886s 3s/step - loss: 0.0167 - acc: 0.9946 - val_loss: 4.6579 - val_acc: 0.5138
Epoch 9/50
345/345 [==============================] - 887s 3s/step - loss: 0.0140 - acc: 0.9956 - val_loss: 5.9067 - val_acc: 0.5241
Epoch 10/50
345/345 [==============================] - 887s 3s/step - loss: 0.0054 - acc: 0.9979 - val_loss: 5.9841 - val_acc: 0.6063
Epoch 11/50
345/345 [==============================] - 886s 3s/step - loss: 0.0209 - acc: 0.9944 - val_loss: 5.1235 - val_acc: 0.5234
Epoch 12/50
345/345 [==============================] - 891s 3s/step - loss: 0.0109 - acc: 0.9961 - val_loss: 3.5280 - val_acc: 0.6063
Epoch 13/50
345/345 [==============================] - 885s 3s/step - loss: 0.0104 - acc: 0.9969 - val_loss: 4.7842 - val_acc: 0.6421
Epoch 14/50
345/345 [==============================] - 889s 3s/step - loss: 0.0104 - acc: 0.9973 - val_loss: 7.0409 - val_acc: 0.5748
Epoch 15/50
345/345 [==============================] - 887s 3s/step - loss: 0.0352 - acc: 0.9898 - val_loss: 4.7940 - val_acc: 0.4653
Epoch 16/50
345/345 [==============================] - 881s 3s/step - loss: 0.0352 - acc: 0.9897 - val_loss: 5.0822 - val_acc: 0.5652
Epoch 17/50
345/345 [==============================] - 877s 3s/step - loss: 0.0076 - acc: 0.9977 - val_loss: 4.5629 - val_acc: 0.5974
```



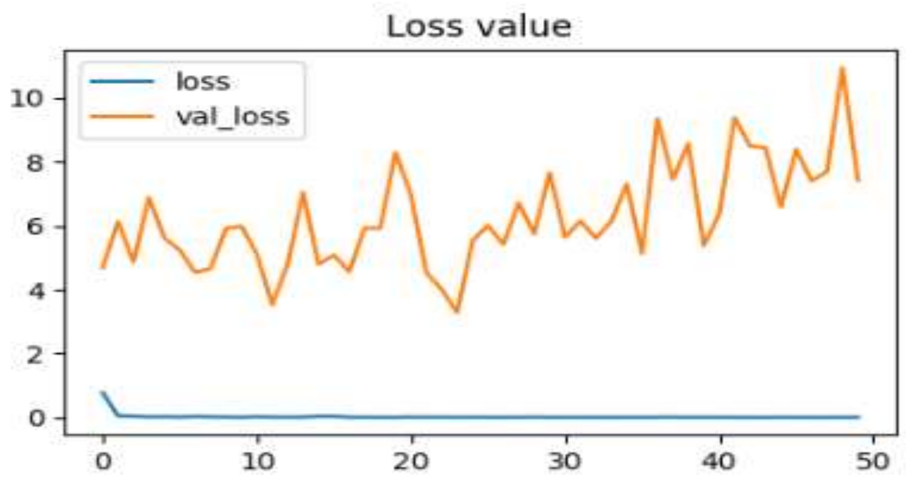
```
Epoch 35/50
345/345 [==============================] - 888s 3s/step - loss: 0.0034 - acc: 0.9989 - val_loss: 7.3082 - val_acc: 0.5018
Epoch 36/50
345/345 [==============================] - 887s 3s/step - loss: 0.0033 - acc: 0.9993 - val_loss: 5.1301 - val_acc: 0.5773
Epoch 37/50
345/345 [==============================] - 889s 3s/step - loss: 0.0066 - acc: 0.9980 - val_loss: 9.3388 - val_acc: 0.6637
Epoch 38/50
345/345 [==============================] - 889s 3s/step - loss: 0.0121 - acc: 0.9965 - val_loss: 7.4489 - val_acc: 0.4901
Epoch 39/50
345/345 [==============================] - 887s 3s/step - loss: 0.0037 - acc: 0.9990 - val_loss: 8.5773 - val_acc: 0.5578
Epoch 40/50
345/345 [==============================] - 886s 3s/step - loss: 0.0059 - acc: 0.9985 - val_loss: 5.3746 - val_acc: 0.5358
Epoch 41/50
345/345 [==============================] - 884s 3s/step - loss: 0.0057 - acc: 0.9987 - val_loss: 6.3553 - val_acc: 0.6368
Epoch 42/50
345/345 [==============================] - 896s 3s/step - loss: 0.0026 - acc: 0.9995 - val_loss: 9.3820 - val_acc: 0.6651
Epoch 43/50
345/345 [==============================] - 900s 3s/step - loss: 9.5994e-04 - acc: 0.9997 - val_loss: 8.4869 - val_acc: 0.5985
Epoch 44/50
345/345 [==============================] - 890s 3s/step - loss: 1.2167e-04 - acc: 1.0000 - val_loss: 8.4374 - val_acc: 0.5974
Epoch 45/50
345/345 [==============================] - 891s 3s/step - loss: 0.0063 - acc: 0.9981 - val_loss: 6.5842 - val_acc: 0.6006
Epoch 46/50
345/345 [==============================] - 889s 3s/step - loss: 0.0012 - acc: 0.9995 - val_loss: 8.3901 - val_acc: 0.6580
Epoch 47/50
345/345 [==============================] - 890s 3s/step - loss: 0.0034 - acc: 0.9993 - val_loss: 7.4001 - val_acc: 0.6655
Epoch 48/50
345/345 [==============================] - 889s 3s/step - loss: 2.5880e-04 - acc: 1.0000 - val_loss: 7.6939 - val_acc: 0.5861
Epoch 49/50
345/345 [==============================] - 883s 3s/step - loss: 0.0035 - acc: 0.9989 - val_loss: 10.9411 - val_acc: 0.6339
Epoch 50/50
345/345 [==============================] - 891s 3s/step - loss: 0.0059 - acc: 0.9986 - val_loss: 7.4194 - val_acc: 0.6095
```

```
plt.figure(figsize=(5,3))
plt.title('Accuracy scores')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['acc', 'val_acc'])
plt.show()
```





```
plt.figure(figsize=(5,3))
plt.title('Loss value')
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss', 'val_loss'])
plt.show()
```



REFERENCES

[1]. Lin, W., Tong, T., Gao, Q., Guo, D., Du, X., Yang, Y., ... & Alzheimer's Disease Neuroimaging Initiative. (2018). Convolutional neural networks-based MRI image analysis for the Alzheimer's disease prediction from mild cognitive impairment. *Frontiers in neuroscience*, *12*, 777.

[2]. Breijyeh, Z., & Karaman, R. (2020). Comprehensive review on Alzheimer's disease: causes and treatment. *Molecules*, *25*(24), 5789.

[3]. Guttman, R., Altman, R. D., & Nielsen, N. H. (1999). Alzheimer disease: report of the Council on Scientific Affairs. *Archives of family medicine*, *8*(4), 347.

[4]. Loddo, A., Buttau, S., & Di Ruberto, C. (2022). Deep learning based pipelines for

Alzheimer's disease diagnosis: a comparative study and a novel deep-ensemble method. *Computers in biology and medicine*, *141*, 105032.

[5]. Buyrukoğlu, S. (2021, September). Improvement of machine learning models' performances based on ensemble learning for the detection of Alzheimer disease. In *2021 6th International Conference on Computer Science and Engineering (UBMK)* (pp. 102-106). IEEE.

[6]. Suk, H. I., & Shen, D. (2016). Deep ensemble sparse regression network for Alzheimer's disease diagnosis. In *Machine Learning in Medical Imaging: 7th International Workshop, MLMI 2016, Held in Conjunction with MICCAI 2016, Athens, Greece, October 17, 2016, Proceedings 7* (pp. 113-121). Springer International Publishing.

[7]. Grassi, M., Rouleaux, N., Caldirola, D., Loewenstein, D., Schruers, K., Perna, G., ... & Alzheimer's Disease Neuroimaging Initiative. (2019). A novel ensemble-based machine learning algorithm to predict the conversion from mild cognitive impairment to Alzheimer's disease using socio-demographic characteristics, clinical information, and neuropsychological measures. *Frontiers in neurology*, *10*, 756.

[8]. Tanveer, M., Rashid, A. H., Ganaie, M. A., Reza, M., Razzak, I., & Hua, K. L. (2021). Classification of Alzheimer's disease using ensemble of deep neural networks trained through transfer learning. *IEEE Journal of Biomedical and Health Informatics*, *26*(4), 1453-1463.

[9]. Islam, J., & Zhang, Y. (2018). Brain MRI analysis for Alzheimer's disease diagnosis using an ensemble system of deep convolutional neural networks. *Brain informatics*, *5*, 1-14.

[10]. Khoei, T. T., Labuhn, M. C., Caleb, T. D., Hu, W. C., & Kaabouch, N. (2021, May). A Stacking-based Ensemble Learning Model with Genetic Algorithm For detecting Early Stages of Alzheimer's Disease. In *2021 IEEE International Conference on Electro Information Technology (EIT)* (pp. 215-222). IEEE.

[11]. Kim, M., Kim, J., Qu, J., Huang, H., Long, Q., Sohn, K. A., ... & Shen, L. (2021, December). Interpretable temporal graph neural network for prognostic prediction of Alzheimer's disease using longitudinal neuroimaging data. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (pp. 1381-1384). IEEE.

[12]. Agarwal, D., Marques, G., de la Torre-Díez, I., Franco Martin, M. A., García Zapiraín, B., & Martín Rodríguez, F. (2021). Transfer learning for Alzheimer's

disease through neuroimaging biomarkers: a systematic review. *Sensors*, *21*(21), 7259.

[13]. Ramzan, F., Khan, M. U. G., Rehmat, A., Iqbal, S., Saba, T., Rehman, A., & Mehmood, Z. (2020). A deep learning approach for automated diagnosis and multi-class classification of Alzheimer’s disease stages using resting-state fMRI and residual neural networks. *Journal of medical systems*, *44*, 1-16.

[14]. Li, H., Habes, M., Wolk, D. A., Fan, Y., & Alzheimer's Disease Neuroimaging Initiative. (2019). A deep learning model for early prediction of Alzheimer's disease dementia based on hippocampal magnetic resonance imaging data. *Alzheimer's & Dementia*, *15*(8), 1059-1070.

[15]. Kang, W., Lin, L., Zhang, B., Shen, X., Wu, S., & Alzheimer's Disease Neuroimaging Initiative. (2021). Multi-model and multi-slice ensemble learning architecture based on 2D convolutional neural networks for Alzheimer's disease diagnosis. *Computers in Biology and Medicine*, *136*, 104678.

[16]. Tripoliti, E. E., Fotiadis, D. I., Argyropoulou, M., & Manis, G. (2010). A six stage approach for the diagnosis of the Alzheimer’s disease based on fMRI data. *Journal of biomedical informatics*, *43*(2), 307-320.

[17]. Aël Chetelat, G., & Baron, J. C. (2003). Early diagnosis of Alzheimer’s disease: contribution of structural neuroimaging. *Neuroimage*, *18*(2), 525-541.

[18]. Petersen, R. C., Doody, R., Kurz, A., Mohs, R. C., Morris, J. C., Rabins, P. V., ... & Winblad, B. (2001). Current concepts in mild cognitive impairment. *Archives of neurology*, *58*(12), 1985-1992.

[19]. Bokde, A. L. W., Lopez-Bayo, P., Meindl, T., Pechler, S., Born, C., Faltraco, F., ... & Hampel, H. (2006). Functional connectivity of the fusiform gyrus during a face-matching task in subjects with mild cognitive impairment. *Brain*, *129*(5), 1113-1124.

[20]. Gray, K. R., Aljabar, P., Heckemann, R. A., Hammers, A., Rueckert, D., & Alzheimer's Disease Neuroimaging Initiative. (2013). Random forest-based similarity measures for multi-modal classification of Alzheimer's disease. *NeuroImage*, *65*, 167-175.

[21]. Whitwell, J. L., Shiung, M. M., Przybelski, S. A., Weigand, S. D., Knopman, D. S., Boeve, B. F., ... & Jack, C. R. (2008). MRI patterns of atrophy associated with progression to AD in amnestic mild cognitive impairment. *Neurology*, *70*(7), 512-520.

[22]. Zhang, D., Wang, Y., Zhou, L., Yuan, H., Shen, D., & Alzheimer's Disease Neuroimaging Initiative. (2011). Multimodal classification of Alzheimer's disease and

mild cognitive impairment. *Neuroimage*, *55*(3), 856-867.

[23]. Eskildsen, S. F., Coupé, P., García-Lorenzo, D., Fonov, V., Pruessner, J. C., Collins, D. L., & Alzheimer's Disease Neuroimaging Initiative. (2013). Prediction of Alzheimer's disease in subjects with mild cognitive impairment from the ADNI cohort using patterns of cortical thinning. *Neuroimage*, *65*, 511-521.

[24]. Misra, C., Fan, Y., & Davatzikos, C. (2009). Baseline and longitudinal patterns of brain atrophy in MCI patients, and their use in prediction of short-term conversion to AD: results from ADNI. *Neuroimage*, *44*(4), 1415-1422.

[25]. Wolz, R., Julkunen, V., Koikkalainen, J., Niskanen, E., Zhang, D. P., Rueckert, D., ... & Alzheimer's Disease Neuroimaging Initiative. (2011). Multi-method analysis of MRI images in early diagnostics of Alzheimer's disease. *PloS one*, *6*(10), e25446.

[26]. de Vos, F., Schouten, T. M., Hafkemeijer, A., Dopper, E. G., van Swieten, J. C., de Rooij, M., ... & Rombouts, S. A. (2016). Combining multiple anatomical MRI measures improves Alzheimer's disease classification. *Human brain mapping*, *37*(5), 1920-1929.

[27]. Mesulam, M. M. (1998). Some cholinergic themes related to Alzheimer's disease: synaptology of the nucleus basalis, location of m2 receptors, interactions with amyloid metabolism, and perturbations of cortical plasticity. *Journal of Physiology-Paris*, *92*(3-4), 293-298.

[28]. Fleisher, A. S., Sherzai, A., Taylor, C., Langbaum, J. B., Chen, K., & Buxton, R. B. (2009). Resting-state BOLD networks versus task-associated functional MRI for distinguishing Alzheimer's disease risk groups. *Neuroimage*, *47*(4), 1678-1690.

[29]. He, Y., Chen, Z. J., & Evans, A. C. (2007). Small-world anatomical networks in the human brain revealed by cortical thickness from MRI. *Cerebral cortex*, *17*(10), 2407-2419.

[30]. Rombouts, S. A., Barkhof, F., Goekoop, R., Stam, C. J., & Scheltens, P. (2005). Altered resting state networks in mild cognitive impairment and mild Alzheimer's disease: an fMRI study. *Human brain mapping*, *26*(4), 231-239.

[31]. Sorg, C., Riedl, V., Muhlau, M., Calhoun, V. D., Eichele, T., Läer, L., ... & Wohlschläger, A. M. (2007). Selective changes of resting-state networks in individuals at risk for Alzheimer's disease. *Proceedings of the National Academy of Sciences*, *104*(47), 18760-18765.

[32]. Supekar, K., Menon, V., Rubin, D., Musen, M., & Greicius, M. D. (2008). Network analysis of intrinsic functional brain connectivity in Alzheimer's

disease. *PLoS computational biology*, *4*(6), e1000100.

[33]. Bozzali, M., Padovani, A., Caltagirone, C., & Borroni, B. (2011). Regional grey matter loss and brain disconnection across Alzheimer disease evolution. *Current medicinal chemistry*, *18*(16), 2452-2458.

[34]. Khazaee, A., Ebrahimzadeh, A., & Babajani-Feremi, A. (2015). Application of pattern recognition and graph theoretical approaches to analysis of brain network in Alzheimer's disease. *Journal of Medical Imaging and Health Informatics, 5*(6), 1145-1155.

[35]. Khazaee, A., Ebrahimzadeh, A., & Babajani-Feremi, A. (2015). Identifying patients with Alzheimer's disease using resting-state fMRI and graph theory. *Clinical Neurophysiology*, *126*(11), 2132-2141.

[36]. Khazaee, A., Ebrahimzadeh, A., & Babajani-Feremi, A. (2016). Application of advanced machine learning methods on resting-state fMRI network for identification of mild cognitive impairment and Alzheimer's disease. *Brain imaging and behavior*, *10*, 799-817.

[37]. Khazaee, A., Ebrahimzadeh, A., Babajani-Feremi, A., & Alzheimer's Disease Neuroimaging Initiative. (2017). Classification of patients with MCI and AD from healthy controls using directed graph measures of resting-state fMRI. *Behavioural brain research*, *322*, 339-350.

[38]. Kavitha, C., Mani, V., Srividhya, S. R., Khalaf, O. I., & Tavera Romero, C. A. (2022). Early-stage Alzheimer's disease prediction using machine learning models. *Frontiers in public health*, *10*, 240.

[39]. Parisot, S., Ktena, S. I., Ferrante, E., Lee, M., Guerrero, R., Glocker, B., & Rueckert, D. (2018). Disease prediction using graph convolutional networks: application to autism spectrum disorder and Alzheimer's disease. *Medical image analysis*, *48*, 117-130.

[40]. Wang, L., Li, P., Hou, M., Zhang, X., Cao, X., & Li, H. (2021). Construction of a risk prediction model for Alzheimer's disease in the elderly population. *BMC neurology*, *21*, 1-10.

[41]. Albright, J., & Alzheimer's Disease Neuroimaging Initiative. (2019). Forecasting the progression of Alzheimer's disease using neural networks and a novel preprocessing algorithm. *Alzheimer's & Dementia: Translational Research & Clinical Interventions*, *5*, 483-491.

[42]. Bari Antor, M., Jamil, A. H. M., Mamtaz, M., Monirujjaman Khan, M., Aljahdali, S., Kaur, M., ... & Masud, M. (2021). A comparative analysis of machine learning algorithms to predict alzheimer's disease. *Journal of Healthcare Engineering*, *2021*.

[43]. Malavika, G., Rajathi, N., Vanitha, V., & Parameswari, P. (2020). Alzheimer Disease Forecasting using Machine Learning Algorithm. *Biosc. Biotech. Res. Comm. Special Issue*, *13*(11).

[44]. Leong, L. K., & Abdullah, A. A. (2019, November). Prediction of alzheimer's disease (AD) using machine learning techniques with Boruta algorithm as feature selection method. In *Journal of Physics: Conference Series* (Vol. 1372, No. 1, p. 012065). IOP Publishing.

[45]. Li, H., Habes, M., Wolk, D. A., Fan, Y., & Alzheimer's Disease Neuroimaging Initiative. (2019). A deep learning model for early prediction of Alzheimer's disease dementia based on hippocampal magnetic resonance imaging data. *Alzheimer's & Dementia*, *15*(8), 1059-1070.

[46]. Buyrukoğlu, S. (2021). Early detection of Alzheimer's disease using data mining: comparison of ensemble feature selection approaches. *Konya Mühendislik Bilimleri Dergisi*, *9*(1), 50-61.

[47]. James, C., Ranson, J. M., Everson, R., & Llewellyn, D. J. (2021). Performance of machine learning algorithms for predicting progression to dementia in memory clinic patients. *JAMA network open*, *4*(12), e2136553-e2136553.