

CATEGORIZATION AND COMPARISON OF ALZHEIMER'S DISEASE USING ML AND CNN

*A project report submitted in partial fulfilment of the requirement
for the award of degree of*

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

B. Satyanvesh
(18341A0523)

Yogita Adari
(18341A0503)

Likitha Adari
(18341A0502)

K Kushal Mehar
(18341A0553)

B. Harikrishna Naik
(18341A0517)

K. Murali
(18341A0554)

Under the esteemed guidance of

Ms. P. RAMYA

Assistant Professor, Dept. of CSE

GMR Institute of Technology

An Autonomous Institute Affiliated to JNTUK, Kakinada

(Accredited by NBA, NAAC with 'A' Grade & ISO 9001:2008 Certified Institution)

**GMR Nagar, Rajam – 532127,
Andhra Pradesh, India
April 2020**

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the thesis entitled **CATEGORIZATION AND COMPARISON OF ALZHEIMER'S DISEASE USING ML AND CNN** submitted by **B.SATYANVESH (18341A0523), YOGITA ADARI (18341A0503), LIKITHA ADARI (18345A0502), K. KUSHAL MEHAR (18341A0553), B.HARIKRISHNA NAIK (18341A0517), K.MURALI (18341A0554)** has been carried out in partial fulfilment of the requirement for the award of degree of **Bachelor of Technology in Computer Science and Engineering of GMRIT, Rajam** affiliated to **JNTUK, KAKINADA** is a record of bonafide work carried out by them under my guidance & supervision. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree.

Signature of Supervisor

Ms. P. Ramya
Assistant Professor
Department of CSE
GMRIT, Rajam.

Signature of HOD

Dr. A. V. Ramana
Professor & Head
Department of CSE
GMRIT, Rajam.

The report is submitted for the viva-voce examination held on

Signature of Internal Examiner

Signature of External Examiner

ACKNOWLEDGEMENT

It gives us an immense pleasure to express deep sense of gratitude to my guide **Ms. P. RAMYA**, Assistant Professor, Department of Computer Science and Engineering for her whole hearted and invaluable guidance throughout the project work. Without her sustained and sincere effort, this project work would not have taken this shape. She encouraged and helped us to overcome various difficulties that we have faced at various stages of our project work.

We would like to sincerely thank our Head of the department **Dr. A. V. Ramana**, for providing all the necessary facilities that led to the successful completion of our project work.

We would like to take this opportunity to thank our beloved Principal **Dr.C.L.V.R.S.V.Prasad**, for providing all the necessary facilities and a great support to us in completing the project work.

We would like to thank all the faculty members and the non-teaching staff of the Department of Electronics and Communication Engineering for their direct or indirect support for helping us in completion of this project work.

Finally, we would like to thank all of our friends and family members for their continuous help and encouragement.

B. SATYANVESH (18341A0523)

YOGITA ADARI (18341A0503)

LIKITHA ADARI (18341A0502)

K KUSHAL MEHAR (18341A0553)

B. HARIKRISHNA NAIK (18341A0517)

K. MURALI (18341A0556)

ABSTRACT

Alzheimer's disease is one of the classifications of dementia and is sometimes referred to as senile dementia. Most prevalent symptoms include memory loss and cognitive resistance to do simple tasks. Although the disease is rife, a lack of discernible biomarkers often leads to delayed treatment. Number of patients who undergo accurate diagnosis and receive an appropriate treatment is very low. Usually, a very large number of people receive diagnosis when it is too late. Given the fact that this illness doesn't have a cure, only an early diagnosis and prevention methods can delay the onset. Algorithms from both machine learning and Deep neural networks are efficient when used to analyze the biomarkers of the disease. Biomarkers such as Plaques and tangles in the Gray-matter combined with past trends observed by the algorithm determine the percentage possibility of underlying AD. The analysis can be done using numerical and neuro-imaging data. Machine learning algorithms like SVM, Random Forest, and Logistic Regression use numerical data that marks the percentage change of the hippocampus and entorhinal cortex; Deep learning algorithm like CNN (convolutional neural network) works with TW1 images from MRI scans and classifies into mild, moderate and severe AD.

Keywords: Alzheimer's Disease, ADNI, Machine Learning Algorithms, Convolutional Neural Network

TABLE OF CONTENTS

CONTEXT NAME	PAGE NO
ACKNOWLEDGEMENT	iii
ABSTRACT	iv
LIST OF TABLES	vii
LIST OF FIGURES	viii
LIST OF SYMBOLS & ABBREVIATIONS	ix
1. INTRODUCTION	
1.1 Introduction	1
1.2 Major challenges in the current literature in line with your proposed work	2
1.3 Solutions to those challenges	2
2. LITERATURE SURVEY	3
2.1 Comparison table	13
3. METHODOLOGY	14
3.1 Architecture	14
3.2 Dataset	15
3.3 Data Description	15
3.4 Convolutional Neural Network	16
3.4.1 Convolution layer	16
3.4.2 Pooling layer	17
3.4.3 Fully Connected Layer	18
3.4.4 Activation Functions	18
3.5 Machine Learning Classifiers	20
3.5.1 Logistic Regression	20
3.5.2 Random Forest	21
3.5.3 Support Vector Machine	22

4. IMPLEMENTATION	23
4.1 Importing Libraries	23
4.2 Dataset Loading	23
4.3 Data Preprocessing	
4.3.1 Removing rows with missing values	23
4.3.2 Imputation	24
4.3.3 Splitting Train/Validation/Test sets	24
4.3.4 Cross Validation	24
4.3.5 Image Augmentation	24
4.4 Performance Measures	25
4.4.1 Confusion Matrix	25
4.4.2 Accuracy	26
4.4.3 Precision	26
4.4.4 Recall	26
4.4.5 Specificity	26
4.4.6 F1 Score	26
4.5 Graphical User Interface	27
4.5.1 Front-end Development	27
4.5.2 Database Connectivity	27
5. RESULTS	28
5.1 Evaluation Results	28
5.2 GUI	33
6. CONCLUSIONS AND FUTURE SCOPE	34
APPENDIX	35
REFERENCES	57

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
1	Comparison Table	12
2	Values of performance matrix of all algorithm	31

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
1	Architecture of AD prediction using ML and CNN	14
2	Convolution layer	17
3	Pooling layer	17
4	Sigmoid function curves	18
5	Tan h function curve	19
6	ReLu function curve	19
7	SoftMax function curve	20
8	Logistic functions	20
9	Random Forest	21
10	Linear vectors	22
11	Multi linear vectors	22
12	Performance Metrics for Logistic Regression	28
13	Performance Metrics for Support Vector Machine	28
14	Performance metrics for Random Forest	28
15	Acc graph for CNN Model	29
16	AUC graph for CNN Model	29
17	Model Loss graph for CNN	29
18	Testing Accuracy for CNN Model	30
19	Confusion matrix and Classification Report for CNN Model	30
20	Confusion Matrix	31
21	Bar Graph representing all metrics of ML and CNN	32
22	GUI for ML Classifiers	33
23	GUI for CNN Model	33

LIST OF SYMBOLS & ABBREVIATIONS

AD	:	Alzheimer's Disease
AUC	:	Area Under Curve
ADNI	:	Alzheimer's Disease Neuroimaging Initiative
ANN	:	Artificial Neural Network
ASF	:	Atlas Scaling Factor
BFS	:	Breadth First search
BN	:	Batch Normalization
CABD	:	Computer-Aided-Brain-Diagnosis
CBIR	:	Content-based image retrieval
CERF	:	Cluster Evolutionary Random Forest
CNN	:	Convolutional Neural Network
DFS	:	Depth First Search
DL	:	Deep Learning
EDUC	:	Years of Education
EHR	:	Electronic Health Record
ETIV	:	Estimated Total Intracranial Volume
FCNN	:	Fully Connected Neural Network
FTD	:	Frontotemporal Dementia
GAP	:	Global Average Pooling
KNN	:	K – Nearest Neighbor
LDA	:	Linear Discriminant Analysis
LDFL	:	Landmark-Based Deep Feature Learning
MCI	:	Mild Cognitive Impairment
ML	:	Machine Learning
MLP	:	Multilayer Perceptron

MIRIAD	:	Minimal Interval Resonance Imaging in Alzheimer's Disease
MMSE	:	Mini Mental State Examination
NWBV	:	Normalize Whole Brain Volume
NR	:	Neural Regression
PCA	:	Principal Component Analysis
PHR	:	Patient Health Record
PET	:	Positron Emission Tomography
SES	:	Socioeconomic Status
SPECT	:	Single-Photon Emission Computed Tomography
SVM	:	Support Vector Machine

1. INTRODUCTION

A woman in her late 70s complains about a loss of memory segments, and how she was unable to differentiate her house from the rest of the houses in the colony. When asked why she postponed so long to refer a neurologist, she claimed that she cast-off the early symptoms as nothing but stress. Over 55 million people have registered the problem with retention and doing chores which are rather easy. Even though it sounds meek, over the time it takes a toll on the patient and ultimately reduces the capability to interpret on their own. Although there is no cure, current treatments include reversing the clock to before the treatment, i.e., up to 6-9 months before starting the treatment. These treatments help the patient to do their tasks self-reliantly. This sickness is affecting so many nevertheless, the treatment doesn't reach all. It's just too late for some patients. The primary diagnosis includes talking to the family about the first onset of the disease, evaluating history about the ailment, cognitive testing (neuropsychological evaluation) that determines the problem-solving ability, routine screening for memory-loss, regular MRI (not to mention the risk of frequent exposure to the radio waves), image-scanning like (PET and SPECT) scans that are used to conclude about the atypical dementia. Finally, a lumbar puncture or amyloid can confirm the diagnosis of Alzheimer's Disease. All in all, it is a tedious task. Whereas using a ML and DL model reduces the time to diagnose the disease and improves the efficiency with which it can be detected. Using the neuroimaging studies in the dataset, data should be pre-processed and cleaned to retrieve the features. These features when fitted into a respective model produce a result that categorizes the level of AD (or monitors a conversion from MCI to AD). The approach includes using different types of methodologies from both machine learning and deep learning. A comparison between both the models produces an effective board to pick the best model to classify the disease. Machine Learning

models included are Support vector machine, Logistic Regression, Decision Tree and Random Forest. Each with an advantage of its own, work with numeric data. On the other hand, CNN uses neuro-image dataset. Convolution neural network uses input layer, ConvNets (convo + ReLu), pooling layers, fully connected layer, SoftMax layer and an output layer. CNN is widely known for images classification. It has the strength in categorizing the images from one another due to its robust channels. Once the models are fitted, their respective efficiency is tested. Final result is hence produced which determines whether the comparison has been successful.

Major challenges in the current literature in line with the proposed work:

Although the existing methods have been efficient enough in detecting the onset of the disease, the dominance of CNN over machine learning algorithm can be quite significant. Support vector machine is the only algorithm that can measure up with CNN in terms of performance. Major challenge lies in correctly diagnosing the illness at the right moment, as any delay in recognizing the symptoms can worsen the occurrence of the disease. Making it impossible for treating the patient. Hence, any mis-diagnosis possess greater threat than the disease itself. It is therefore of stark importance to build a model that is robust and accurate.

Solutions to those challenges :

To build such a model, the only possible way is to train the model with copious amounts of data so that it doesn't falter anymore. Comparing the performances of the ML models and CNN models can determine which model suits the data better. For example, ML models work the best with numeric data; whereas CNN models excel when neuro-image data is considered.

2. LITERATURE SURVEY

[1] Tanveer, M., Richhariya, B., Khan, R. U., Rashid, A. H., Khanna, P., Prasad, M., & Lin, C. T. (2020). Machine learning techniques for the diagnosis of Alzheimer's disease. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 16(1s), 1–35.

They conducted a study of 165 publications published between 2005 and 2019, utilizing various feature extraction and machine learning approaches. The authors examined three important machine learning algorithms for Alzheimer's disease diagnosis: SVM, ANN, and DL. In addition, they have done other learning strategies such as transfer, ensemble, and multi-kernel learning are explored as well. SVM-based models have been frequently employed for Alzheimer's disease, demonstrating its resilience. This is since approaches like ANN suffer from the limitations of local minima, whereas SVM does not. The widespread use of SVM is because it is simpler to explain than deep neural networks, which are black-box models. In the future, this issue should be addressed by concentrating on the clinical interpretability of deep learning models. It's also observed that the feature extraction step has received more attention from researchers than the classification phase.

[2] Bron, E. E., Klein, S., Jiskoot, L. C., Linders, J., van Swieten, J. C., van der Flier, W. M., & van der Lugt, A. (2021). Cross-cohort generalizability of deep and conventional machine learning for MRI-based diagnosis and prediction of Alzheimer's disease. *NeuroImage: Clinical*, 31, 102712.

In this paper patients are divided into two groups. The first group consists of 1715 participants, which was included from ADNI. The second group was included from the health-RI Parelsnoer Neurodegenerative Disease Biobank. Pre-processing the images is done by rectifying the images using N4 algorithm and transforming it to MNI-space with brain masks showing similar transformations. SVM and CNN were the two approaches used to build the model. In SVM, the C parameter is used in 5-fold on the training set. Scikit-learn enabled the classifier implementation. CNN, however used FCN architecture that utilizes two layers of pooling layers similar to many CNN. The whole network consists of 7 models including the filters. The series of network includes, 3D convolution layer, dropout, batch normalization (BN) and a ReLu activation function, followed

by another 3D layer and so on. Saliency maps are another such mechanism that depicted a change in the prediction score. The result hence produced showed that the AUC using modulated GM maps showed higher latency over the AUC using T1w images. For CNN, a similar outcome was observed with GM maps producing higher accuracy than the T1w images.

[3] Kim, J. P., Kim, J., Park, Y. H., Park, S. N., Seo, S. W., & Seong, J.-K. (2019). Machine learning based hierarchical classification of frontotemporal dementia and Alzheimer's disease. *NeuroImage: Clinical*, 23, 101811.

In this paper the author used hierarchical classification for Frontotemporal dementia (FTD) is one of the most common types of early-onset dementia. An individual subject categorization model would be much more useful than a group analysis. The author's goal in this work was to use a machine learning-based classification algorithm using surface-based cortical thickness data to categorize each individual patient into one of diagnostic categories in a hierarchical way. This hierarchical approach was continued throughout the tree until the individual was eventually assigned one of the final clinical labels. The classification findings utilizing the whole hierarchical tree revealed that each participant was correctly categorized into one of five clinical categories with 75.8 percent accuracy. We created an automatic classifier machine learning - based for the differential classification of FTD clinical symptoms. Cortical thickness data alone might diagnose FTD clinical subgroups and AD with good to exceptional accuracy using a fully automated classifier.

[4] Liu, M., Zhang, J., Nie, D., & Shen, D. (2018). Anatomical landmark based deep feature representation for MR images in brain disease diagnosis. *IEEE Journal of Biomedical and Health Informatics*, 22(5), 1476–1485.

The author used data to identify discriminative anatomical landmarks in MR images and then proposed a convolutional neural network (CNN) for patch-based deep feature learning. They suggested an anatomical landmark-based deep feature learning (LDFL) framework for Disease prediction in this study. The suggested LDFL approach has been verified in both brain illness categorization and MR information extraction tasks. The Alzheimer's Disease Neuroimaging Initiative ADNI and the Minimal Interval Resonance Imaging in Alzheimer's Disease (MIRIAD) datasets were used in our study. The general structure of our proposed technique consists of four

major components: 1) landmark discovery, 2) landmark-based patch extraction, 3) patch-based feature learning, and 4) illness classification and picture retrieval applications. In this study, they suggested a landmark-based deep feature learning system for automatically extracting patch-based representations from MR images for the detection of Alzheimer's disease-related brain illness. This method paves the path for discriminative biomarkers in computer-aided diagnosis of Alzheimer's disease and morphological analysis of MR images.

[5] Naik, B., Mehta, A., & Shah, M. (2020). Denouements of Machine Learning and Multimodal Diagnostic Classification of Alzheimer's disease. *Visual Computing for Industry, Biomedicine, and Art*, 3(1).

Alzheimer's disease (AD) is a common type of neurological condition characterized by chronic memory loss and cognitive impairment. Its primary goal is to give the ideal hyperplane that distinguishes data points of one class from data points of another class that were not generated, therefore many tests were carried out to assess the efficiency using a multiclassifier. The primary goal of this research was to examine the accuracies of several ML classification methods when applied to MRI data for AD classification. In the case of advanced stages of Alzheimer's disease, the treatment method must be decided immediately by a doctor. This element poses a significant hurdle to the use of SVM in neuroimaging since some inherent SVM tasks, such as image processing, might take several days to complete. Delaying a clinical decision would thus be futile and perhaps risky to patients. To improve the classification performance rates of the SVM classifier based on a single modality, MRI, adding the PET/SPECT/CSF modality to MRI produced greater accuracy than MRI alone. When a single modality is merged with several modalities, the classifier's accuracy improves.

[6] Mofrad, S. A., & Lundervold, A. S. (2021). A predictive framework based on brain volume trajectories enabling early detection of Alzheimer's disease. *Computerized Medical Imaging and Graphics*, 90, 101910.

The goal is to detect the conversion of CN to MCI to AD (Alzheimer's Detection). Dataset used in this experiment is: ADNI The procedure includes extraction of subcortical segments and cortical parcellation to from TW1 images using Free Surfer v.6.0, and after selecting the 3D regions which show a tendency for AD are then fitted into a model for prediction. They have used a mixed approach which is then again divided into (i) feature selection, model development and validation,

and (ii) model-evaluation. A combination of ADNI and ADNI with AIBL datasets are used for predictive performance analysis. The accuracy obtained is around 73% and 78% for CN and MCI. The main aim of this paper is to improve the accuracy and the performance using the same techniques as its base reference model which had previously achieved only 64% efficiency among 224 candidates. It has also suggested instabilities in the algorithm that leads to low quality of the models. Use of FreeSurfer v.6.0 can be held accountable for reducing some of variation but instability still persists.

[7] Martinez-Murcia, F. J., Ortiz, A., Gorriz, & Castillo-Barnes, D. (2020). Studying the manifold structure of Alzheimer's disease: A deep learning approach using convolutional autoencoders. *IEEE Journal of Biomedical and Health Informatics*, 24(1), 17–26.

CNN replaces MLP (multilayer perception) by introducing faster and accurate models that have changed the course of future models as well. A system of encoder-decoder has been introduced that reduces the reconstruction error and extracts features. This paper also uses an approach of Z-layers features, which further includes MLP, NR (Neural Regression) and SVM. The dense layers in encoder and decoder are interfaced with Z-Layer. The output of the encoder was replaced by GAP (Global Average Pooling), this helps in overcoming the problem of over-fitting and improves convergence while drastically reduced parameters. ReLu are widely used when it comes to CNN, but the Z-layer uses linear activation for Z-manifold. Batch normalization was used to accelerate convergence and works as a regularizer. Regression analysis is used to correlate the Z-features with other data types. The best-scoring tissue map was noted as GM with a correlation coefficient of 0.63. It has a self-supervised environment which guides the model without any prior assumptions. The non-linear data structure can be used for complex datasets. The difference between GM and WM is the case of regression. It is pointed out that AD affects GM first and progresses to WM. Finally, an accuracy of 84% has been achieved.

[8] Al-Shoukry, S., Rassem, T. H., & Makbol, N. M. (2020). Alzheimer's diseases detection by using Deep Learning Algorithms: A mini-review. *IEEE Access*, 8, 77131–77141.

In this paper the author analyze some of the key material on Alzheimer's disease in this work and look at how Deep learning can assist researchers detect the condition at an early stage, as most of the selected patients are already known to have it. In this analysis, they looked at relevant publications that looked at Alzheimer's disease and used Magnetic Resonance Image (MRI) data,

focusing on two primary areas of research: biomarkers and neuroimaging. They also gave a brief history of Alzheimer's disease, including the illness's discovery and brain imaging methods. In the Alzheimer's disease area, it explains the transition from machine learning to Deep learning. A review of Alzheimer's disease modules, datasets, diagnosis approaches, and detection was also given. This study reviewed the some of the important related AD datasets and diagnose techniques and detection. This approach is feasible for early-stage neuroimaging research.

[9] Ghoraani, B., Boettcher, L. N., Hssayeni, M. D., Rosenfeld, A., Tolea, M. I., & Galvin, J. E. (2021). Detection of mild cognitive impairment and alzheimer's disease using dual-task gait assessments and machine learning. *Biomedical Signal Processing and Control*, 64, 102249.

In this paper, the author purposed an approach to establish an accurate and consistent approach for identifying MCI, and AD. This is the first study to identify key gait variables for machine learning-based categorization and to establish an automated and objective approach for detecting a cognitive decline in MCI and AD patients using just gait analysis. The use of gait as a screen for cognitive impairment may urge doctors to perform further tests in order to diagnose MCI and AD. They gathered gait data from 78 older people as they walked in a number of single and dual task situations. Each subject had 108 gait features extracted, and the uncorrelated significant features were identified. The clinical diagnosis was then determined using a machine learning technique based on the specified gait variables. The method achieves 25 uncorrelated significant gait variables for distinguishing healthy vs. MCI, healthy vs. AD, and MCI vs. AD, as well as 13 for MCI vs. AD. Using the specified gait characteristics, the five-fold classification accuracy was 78 percent, which was somewhat lower than 83 percent.

[10] Bi, X.-an, Hu, X., Wu, H., & Wang, Y. (2020). Multimodal Data Analysis of Alzheimer's disease based on Clustering Evolutionary Random Forest. *IEEE Journal of Biomedical and Health Informatics*, 24(10), 2973–2983.

In this work, they conducted multimodal data fusion research on Alzheimer's illness. Our findings have a wide range of implications for Alzheimer's disease diagnostics and the advancement of computational medicine. To begin, they used correlation analysis to look for links between brain areas and genes. their technique effectively fuses data from many modalities and makes follow-up

analysis easier. Second, the CERF is suggested to evaluate "brain region-gene pairings" and extract the discriminative fusion characteristics that distinguish AD from HC. Finally, the CERF is included in an AD diagnostic framework that includes fusion feature synthesis, feature selection, and sample categorization. They discovered aberrant brain areas and pathogenic genes of AD using this framework, including the as thalamic, lingual gyrus, angular gyrus, precuneus, insula DAB1 gene, and LRP1B gene. However, validation of results with bigger data sets, as well as the necessity to validate these conclusions and "brain region-gene combinations" in future studies, must be highlighted.

[11] Hazarika, R. A., Abraham, A., Sur, S. N., Maji, A. K., & Kandar, D. (2021). Different techniques for Alzheimer's disease classification using brain images: A study. *International Journal of Multimedia Information Retrieval*, 10(4), 199–218.

According to the author, standard diagnosis is expensive and not always precise, thus researchers are attempting to build a reliable and low-cost approach for AD classification using brain imaging. Several AD classification techniques are covered in this article, including performance comparisons, benefits and demerits, and detailed observations. The ANN-based classification strategy produces the most convincing findings, as evidenced by the result comparison (approximately 93.19 percent). One of the primary obstacles that researchers may confront is obtaining enough data points from various data sources (both online and offline). Because the anatomy of the brain is so complicated, it is difficult to distinguish the changes. As a result, effective pre-processing activities such as skull stripping, and segmentation of various components are critical as well as difficult.

[12] Kruthika, K. R., Rajeswari, M., & Maheshappa, H. D. (2019). Multistage classifier-based approach for Alzheimer's disease prediction and retrieval. *Informatics in Medicine Unlocked*, 14, 34–42.

Alzheimer's disease (AD) leads to dementia as definite cause and cure of the disease are still unavailable so, earlier stage detection leads to proper treatment. Apart from physical clinical diagnosis and neurological examination paper suggested for better and efficient diagnostic tools for AD. Content-based image retrieval (CBIR) systems applied as it is combining automated image classification systems and radiologist's knowledge leads accuracy of prediction. In this paper author used machine learning models Naïve bayes classifier, Support Vector Machine (SVM) and

K-nearest neighbor (KNN) to classify stages of the disease. In this paper, the swarm intelligence technique - PSO for feature selection was applied to represent the brain structural change that is linked to the Alzheimer's disease is progressing clinically. As compared to earlier individual machine learning techniques, such as SVM and KNN, the multistage classifier utilized in this thesis performed well for AD detection. In addition, the image retrieval strategy that followed the proposed method for AD classification yielded positive results.

[13] 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(2).

As Alzheimer's disease is neurological brain disorder and its incurable. So early-stage detection of Alzheimer's disease can prevent brain tissue damage and proper diagnosis with machine learning by analyzing magnetic resonance imaging (MRI) data. So, author proposed deep convolutional neural network as an approach for detecting AD disease for binary classification the model can find out different stages of Alzheimer's disease. The performance comparison of the proposed classification results the baseline deep CNN models, the ensembled model observed that proposed ensembled model achieves encouraging performance and outperforms the other models and as author made an efficient approach to AD diagnosis using MRI data as he proposed network can be very beneficial for early-stage AD diagnosis. Even though the suggested model has only been evaluated on the AD dataset, we believe it can be applied to other medical classification problems with success. Furthermore, the proposed method has a lot of potential for applying CNN to other fields with a small dataset.

[14] Khan, N. M., Abraham, N., & Hon, M. (2019). Transfer learning with intelligent training data selection for prediction of Alzheimer's disease. *IEEE Access*, 7, 72726–72735.

In this paper author discussed, the detection of Alzheimer's Disease (AD) using MRI's Results which contain neuroimaging data using machine learning. They adopt transfer learning to solve these problems, in which the state-of-the-art VGG architecture is pre-trained with weights from large benchmark datasets of natural images. In this paper, we propose a transfer learning-based method for Alzheimer's diagnosis from MRI images. They hypothesize that adopting a robust and proven architecture for natural images and employing transfer learning with intelligent training data selection can not only improve the accuracy of a model, but also reduce reliance on a large

training set. Finally, author provide Class Activation Maps (CAM) that demonstrate how the proposed model focuses on discriminative image regions that are neuropathologically relevant, and can help the healthcare practitioner in interpreting the model's decision-making process.

[15] Fan, Z., Xu, F., Qi, X., Li, C., & Yao, L. (2019). Classification of Alzheimer's disease based on Brain MRI and machine learning. *Neural Computing and Applications*, 32(7), 1927–1936.

In this paper the author discussed that Alzheimer's disease (AD) is the most common and a neurodegenerative disease with hidden and progressive development accounting for 50–60% of all patients. Alzheimer's disease can lead to changes in memory, cognitive function and behavior. This paper tells us that the method of classifying diseases through the in information of images may be more helpful for clinical diagnosis. The principal component analysis (PCA) is one of the most used to extraction methods at present. The principal component analysis to realize face recognition system. this paper also used principal component analysis combined with linear discriminant analysis (LDA) and support vector machine (SVM) to classify and recognize AD, magnetic resonance imaging (MRI) imaging data. This paper introduces the basic concepts, classification and learning forms of machine learning. At the same time, the basic idea and principle of support vector machine are introduced. It shows that SVM method has a strong ability to fit the disease process of Alzheimer's and can be used to evaluate unknown samples.

[16] Kishore, P., Usha Kumari, C., Kumar, M. N. V. S. S., & Pavani, T. (2021). Detection and analysis of alzheimer's disease using various machine learning algorithms. *Materials Today: Proceedings*, 45, 1502–1508.

According to the author From the perspective of data mining, the proposed arrangement demonstrates a large processing model. As an input, the data is sent to the system. External sources, clinical systems, and the unorganized Electronic Health Record (EHR) / Patient Health Record (PHR). This paper describes the work that was done by preparing an Alzheimer's rate and quality framework using several machine learning methods. It's easier to figure out what stage they're at with the use of classification algorithm, so therapy can begin. As a result, this study presents many algorithms for classifying data in order to increase the efficiency in diagnosing the disease in question, and it is discovered that the Support Vector Machine with linear kernel model provides superior accuracy than other models.

[17] Albright, J. (2019). Forecasting the progression of alzheimer's disease using neural networks and a novel preprocessing algorithm. *Alzheimer's & Dementia: Translational Research & Clinical Interventions*, 5(1), 483–491.

In this research they examined into using machine learning techniques to forecast the progression of Alzheimer's disease in the future using clinical data. Many of the machine-learning algorithms used in this study were successful in predicting the course of Alzheimer's disease in both cognitively normal and MCI individuals. These methods could be used to detect persons at high risk for Alzheimer's disease before they can be identified with MCI or dementia. Because one of the major reasons for the recurrent failure of AD clinical testing is the inability to recognize patients at an early stage, these strategies may help raise the likelihood of finding a therapy for AD. This study looked into using machine learning techniques to forecast the progression of Alzheimer's disease in the future using clinical data.

[18] Noor, M. B., Zenia, N. Z., Kaiser, M. S., Mamun, S. A., & Mahmud, M. (2020). Application of deep learning in detecting neurological disorders from Magnetic Resonance Images: A survey on the detection of alzheimer's disease, parkinson's disease and schizophrenia. *Brain Informatics*, 7(1).

The performance of current deep learning (DL)-based algorithms for detecting neurological illnesses is rigorously examined and compared in this paper. The Convolutional Neural Network surpasses other approaches in diagnosing neurological illnesses, according to a comparison of performance of multiple DL designs across various disorders and imaging modalities. The most prevalent DL approaches have been investigated in this research for diagnosing these three significant neurological illnesses using MRI scan data. These DL approaches' advantages, disadvantages, and performance for MRI images have been summarized. The maximum use of CNN in the identification of Alzheimer's disease and Parkinson's disease was one of the study's main findings.

[19] AbdulAzeem, Y., Bahgat, W. M., & Badawy, M. (2021). A CNN based framework for classification of alzheimer's disease. *Neural Computing and Applications*, 33(16), 10415–10428.

In this study the author presents an end-to-end system for AD classification based on CNN. The framework has five layers: the first process is important for MRI, the second layer for adaptive

thresholding and data augmentation, and the third layer for cross validation. The Glorot Uniform weight initializer prevents neuron activation functions from starting in saturated or dead regions, leading in significantly faster convergence and increased accuracy. Experiments are used to investigate the impact of varying sample sizes, activation functions, and learning rates. The results of the experiments revealed that the suggested framework beats state-of-the-art methodologies in both binary and multi-classification classification accuracy. On the ADNI dataset, the suggested framework had a classification accuracy of 97.5 percent.

[20] Acharya, U. R., Fernandes, S. L., WeiKoh, J. E., Ciaccio, E. J., Fabell, M. K., Tanik, U. J., Rajinikanth, V., & Yeong, C. H. (2019). Automated detection of alzheimer's disease using brain MRI images– a study with various feature extraction techniques. *Journal of Medical Systems*, 43(9).

In this paper, the author's goal of this research is to create a Computer-Aided-Brain-Diagnosis (CABD) system which can tell if a brain scan indicates Alzheimer's disease. Filtering, feature extraction, Student's t-test based feature selection, and KNN based classification are among the quantitative techniques used in the paradigm. The existing feature-based processes are also evaluated in depth, and feature extraction is carried out using the student's t-test. Furthermore, when compared to other approaches, the ST + KNN requires a smaller number of characteristics. The proposed paradigm outperformed the benchmark AD dataset and the brain MRI collected from the medical clinic. Other classification algorithms, such as SVM, neural-networks, random forest, and AdaBoost, will be used instead of KNN in future studies.

CATEGORIZATION AND COMPARISON OF ALZHEIMER'S DISEASE USING ML AND CNN

Comparison Table:

SNo	Author Name	Algorithms Used	Drawbacks	Accuracy
1.	.Deepak Gupta et al.,(2019)	Naïve bayes, KNN, SVM, Random Forest, Decision Tree.	High computation time	94%
2.	Hoda El Merabet et al.,(2019)	Random forest, SVM, ANN	No drawbacks	96%
3.	Hiran V Nath et al.,(2017)	Decision Tree	Decrypting and decode the encrypted object blocks is a time consuming process.	
4.	Samuel Hess et al.,(2019)	CART, Random Forest, Ada boost, Gentle Boost, Robust Boost	Overfitting of data	99.4%
5.	Hrushikesh Shukla et al.,(2019)	Naïve bayes, Random Forest, decision tree, logistic regression	Model doesn't provide any information about the family from which malware is derived	97.2%
6.	S. Abijah Roseline et al.,(2019)	Random forest, Extra trees cla XG Boost, Logistic Regression	No drawbacks discussed	98.91%
7.	Wei Li et al.,(2021)	Naïve bayes, XG Boost, Logistic regression	No drawbacks discussed	97.22%
8.	Muhammad Ijaz et al.,(2019)	Logistic regression, Decision tree, Random forest, AdaBoost classifier, Tree classifier, Gradient classifier	Dynamic analysis had some limitations due to controlled behaviour and it cannot be analysed due to limited access of network.	99.36%
9.	Rushab Vyas et al.,(2017)	KNN, Random Forest, Decision tree, SVM classifier	Took long time on analysing some of the benign files because of the large size.	98.7%
10	Raphael fettaya et al.,(2020)	CNN, Clustering algorithm (HBSCAN)	No drawbacks discussed	99.83%

Table 1

METHODOLOGY

ARCHITECTURE

With copious amounts of data, it becomes strenuous to work the data to produce the desired results, hence the necessity of implementing a model reinforced with an algorithm arises. Although the paper establishes a comparison between the models, it is widely important to understand the working of individual algorithm to pick out the best performing algorithm.

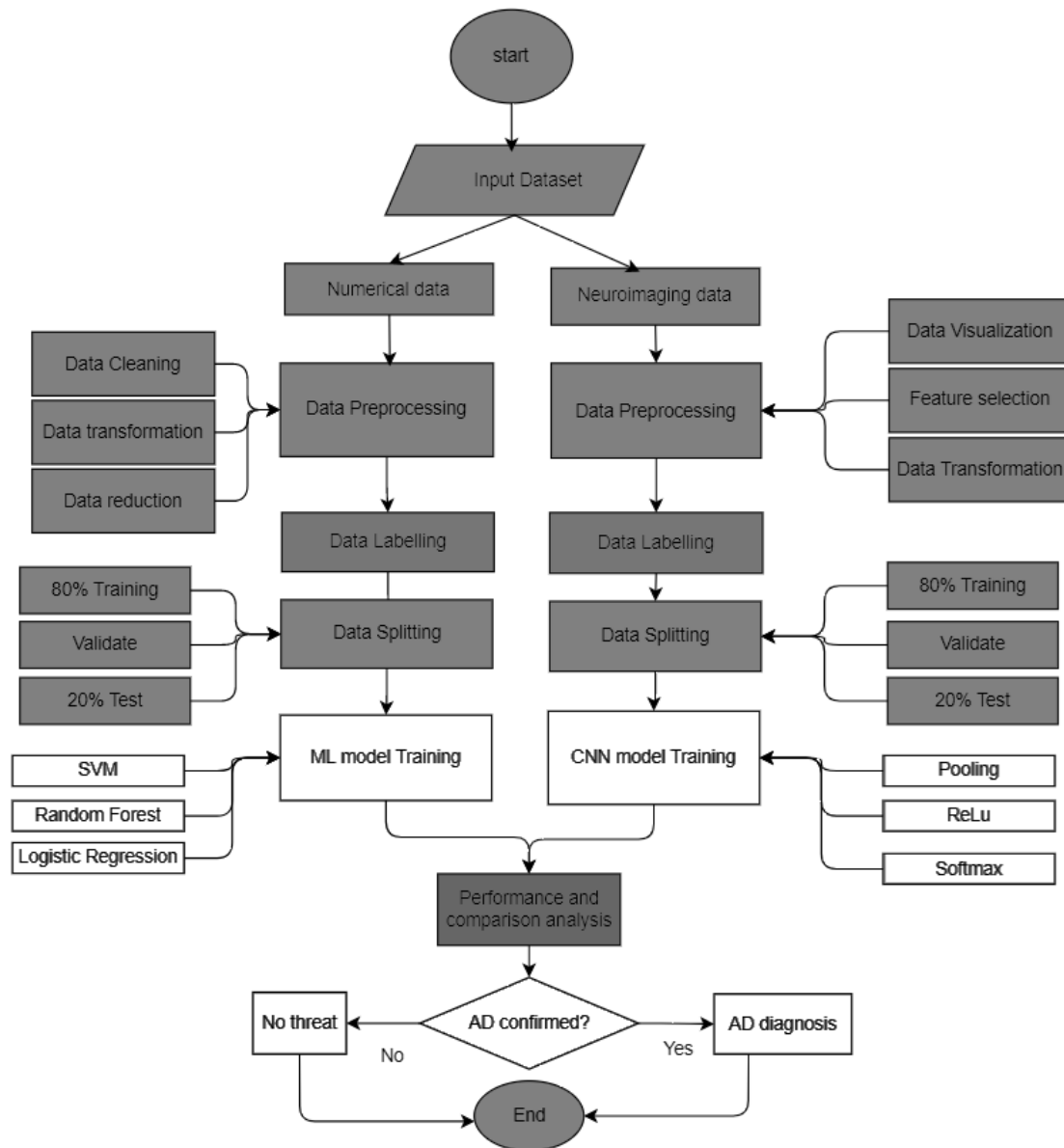


Fig1 Architecture of AD prediction using ML and CNN

DATA SET

We're using the ADNI dataset for this classification. The ADNI dataset contains clinical data on each participant, such as recruiting, demographics, physical tests, and cognitive testing. As comma separated values (CSV) files, the entire collection of clinical data can be downloaded in bulk. It holds enormous potential for groundbreaking findings in Alzheimer's research. Around 6500 images of MRI scans are taken as input for training the model.

DATA DESCRIPTION

For machine learning algorithms, we have used numeric dataset which consist of features like

EDUC	Years of education
SES	Socioeconomic Status
MMSE	Mini Mental State Examination
ETIV	Estimated Total Intracranial Volume
NWBV	Normalize Whole Brain Volume
ASF	Atlas Scaling Factor

- Education: (1 < high school (HS), 2 = HS Graduate, 3 = Some College, 4 = College Graduate, 5 = Beyond College Graduate)
- Socioeconomic Status (SES): (1 = lower, 2 = medium, 3 = intermediate, 4 = higher middle, 5 = upper)
- Mental state examination (MMSE): (0,30). The MMSE is a 30-item survey that has been proven to be accurate and reliable in detecting dementia.
- Total intracranial volume (ETIV) estimate: (1132–1992) mm³. The ETIV variable measures the volume of the intracranial brain.
- Normalized whole brain volume (NWBV): (0.64–0.90) mg. This variable represents the total volume of the brain.
- Atlas scaling factor (ASF): (0.88–1.56) The ASF is a one-parameter scaling factor that allows comparisons of projected total intracranial volume based on human anatomy variances.

CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks (CNN) are artificial neural networks that are inspired by biological systems. In contrast to conventional machine learning models, CNN is provided raw picture pixel values instead of feature vectors. When we observe an image, our brain analyses a tremendous quantity of data. Each neuron has its own field and is coupled to other neurons in such a way that the full visual range is covered. In the neurobiological system, each neuron reacts to signals only in a limited part of the visual field is known as receptive field, and each neuron in a CNN analyses data only within its receptive field.

CNN is more effective than image detection and classification because of functions such as minimal interconnection among successive layers, parameter sharing of weights between nearby pixels, and similar representation.

Layers in Convolutional neural network is:

1. Convolution layer
2. Pooling Layer
3. Fully connected layer

Convolution Layer

The main building block of CNN is the convolutional layer. Convolution is a mathematical operation to merge two sets of information. In our case the convolution is applied on the input data using a convolution filter to produce a feature map.

We perform the convolution operation by sliding this filter over the input. At every location, we do element-wise matrix multiplication and sum the result. This sum goes into the feature map. The green area where the convolution operation takes place is called the *receptive field*. Due to the size of the filter the receptive field is also 3x3.

Let's say we have a 32x32x3 image and we use a filter of size 5x5x3. When the filter is at a particular location it covers a small volume of the input, and we perform the convolution operation described above. We slide the filter over the input like above and perform the convolution at every location aggregating the result in a feature map. This feature map is of size 32x32x1, shown as the red slice on the right.

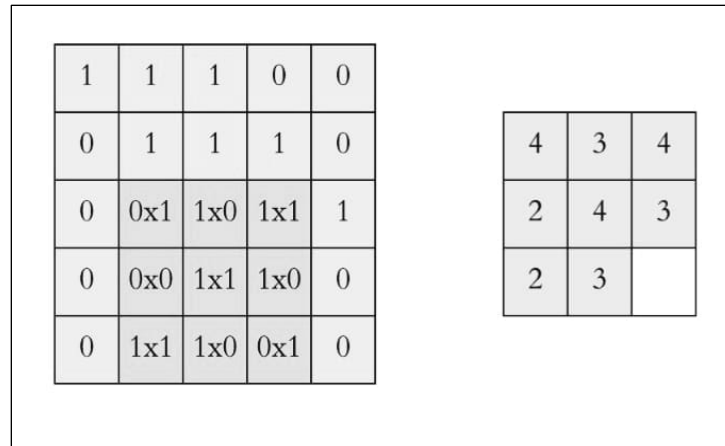


Fig 2 Convolution layer

Pooling Layer

The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

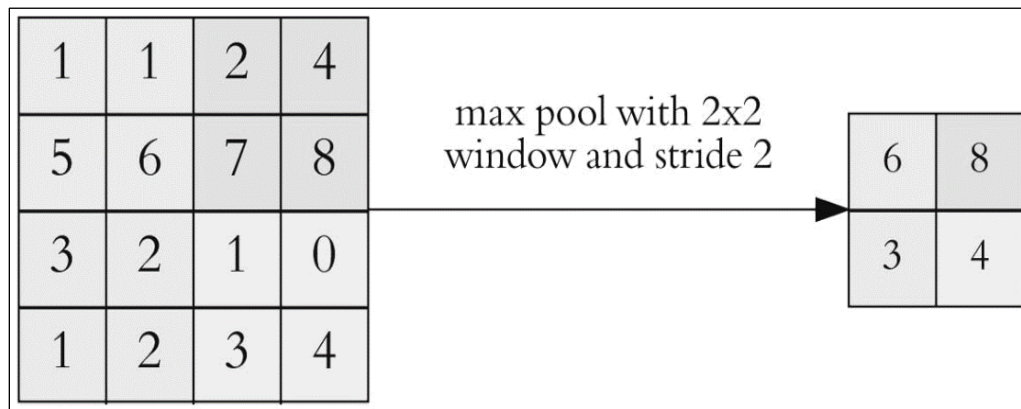


Fig 3 Pooling layer

Now let's work out the feature map dimensions before and after pooling. If the input to the pooling layer has the dimensionality 32x32x10, using the same pooling parameters described above, the result will be a 16x16x10 feature map. Both the height and width of the feature map are halved, but the depth doesn't change because pooling works independently on each depth slice the input.

Fully Connected Layer

Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN. This is why it can be computed as usual by a matrix multiplication followed by a bias effect. The FC layer helps to map the representation between the input and the output. In this, the input image from the previous layers is flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

Activation functions

Activation functions define output of neuron based on a given set of then inputs. A typical activation function is based on conditional probability which will return the value one or zero as an output.

When network input info ip cross the threshold value, the activation functions return value 1 and passes info to the next layers, when network input ip less than the threshold value, then the activation functions return value 0 and the info is not passed on.

- Sigmoid:

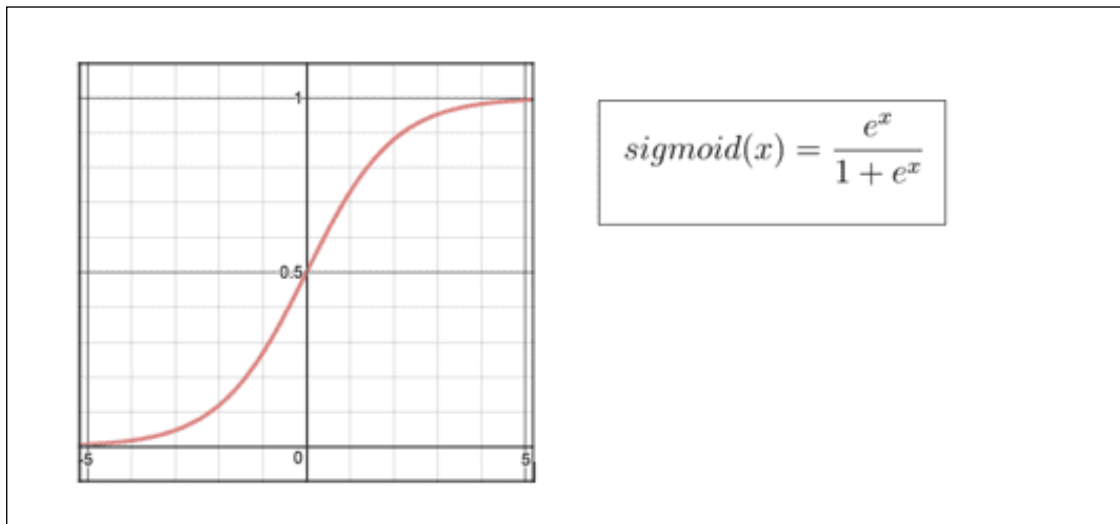


Fig 4 Sigmoid function curve

- Tanh:

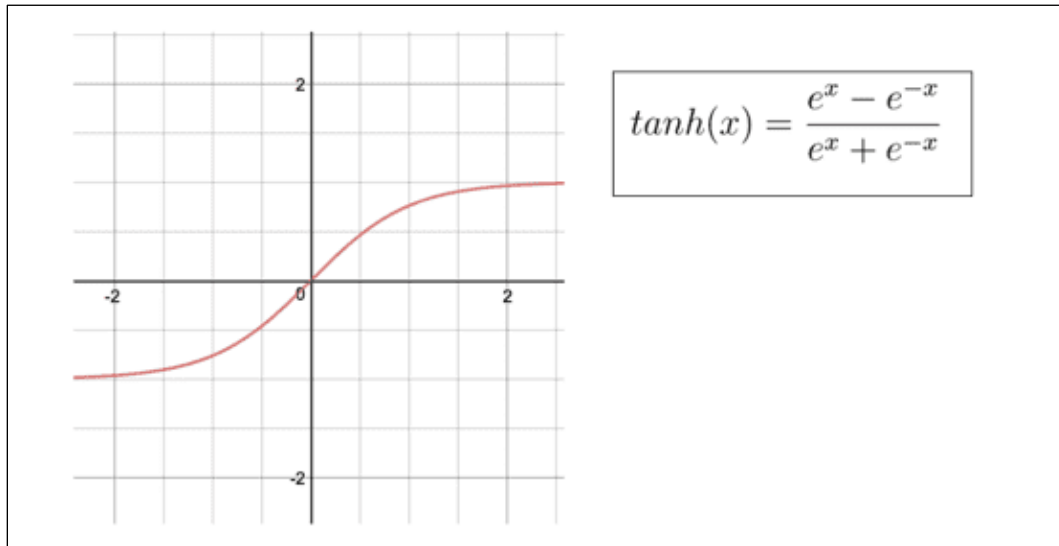


Fig 5 Tan h function curve

- ReLu:

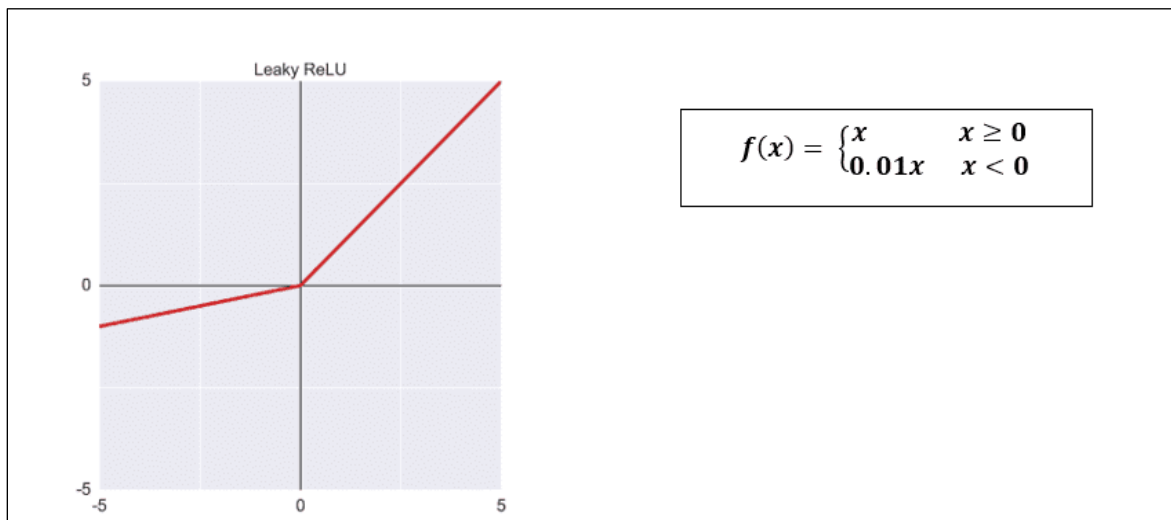


Fig 6 ReLu function curve

- SoftMax

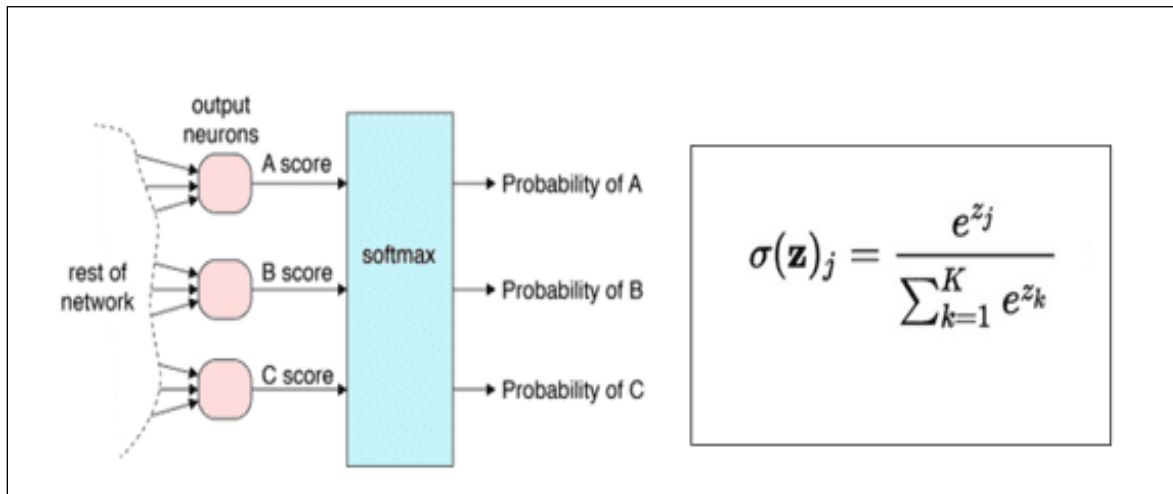


Fig 6 SoftMax function curve

MACHINE LEARNING CLASSIFIERS

LOGISTIC REGRESSION

Primarily used for classification, this model uses sigmoid function. Sigmoid function is represented by an s-shaped curve. This generates the probability of the label \mathbf{x} .

Sigmoid function: -

$$\text{Sig}(x) = 1 / (1 + \exp(-x))$$

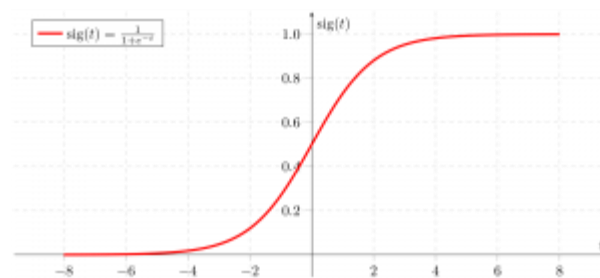


Fig 7 Logistic function

Logistic Regression uses input variables, weights, and output variables. To predict a certain outcome, it usually takes values between 1 and 0. Usually, the values are fitted into the linear regression model and are incorporated with a hypothesis to predict the future values to fit into the respective zones.

The classification made is based on the curve obtained by fitting the values. When applied to the model to predict the indications of Alzheimer's disease, a sigmoid curve will be able to predict the occurrence of the initial biomarkers.

RANDOM FOREST

A subset of supervised learning, this algorithm advances its predecessor Decision Tree by taking a majority voting amongst all the trees; It can tackle the problems of both regression and classification.

The mainstay principle of the decision tree which in turn works under the random tree is that a central node is taken based on the value of Gini index, after which the tree is propagated into further leaves (also known as child nodes). Once the nodes burgeon into a fully developed tree, a decision is made upon the given constraint or a test set. Usually the tree is parsed in different methods such as DFS(depth first search) or BFS(breadth first search). Hence, a final output is generated. A collection of decision trees is known as random forest. The output of random forest is based on the individual trees; to select the final output, usually maximum voting is done but other methods also include minimum voting or average of all the outputs

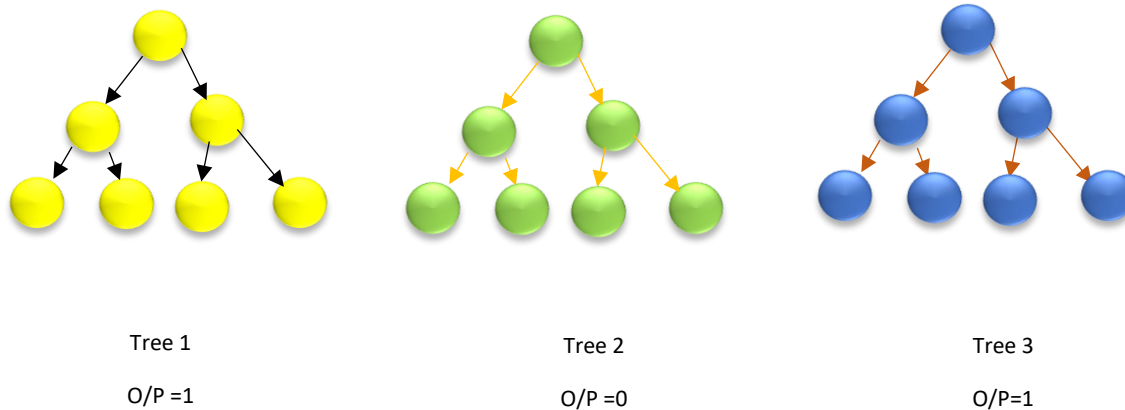


Fig 8 Random forest

Similarly, Random Forest can conclude whether the given dataset is demented or not based on the above process for determining the chances of developing Alzheimer's Disease.

SUPPORT VECTOR MACHINE

A subcategory of supervised learning, this algorithm works surprisingly well with limited data and swiftly classifies the given data.

The main idea of SVM is to create a hyperplane to separate the class labels into categories, this can be done by linearly separating the labels in 2D plane (using a line) or one can increase the dimensions based on the threshold (middle value), linearity and the number of variables. Usually when dealing with an outlier, terms like bias and variance come into picture while classifying a data point in SVM. But a few misclassifications can abet the process; this is called a soft margin. These soft margins are also known as support vectors.

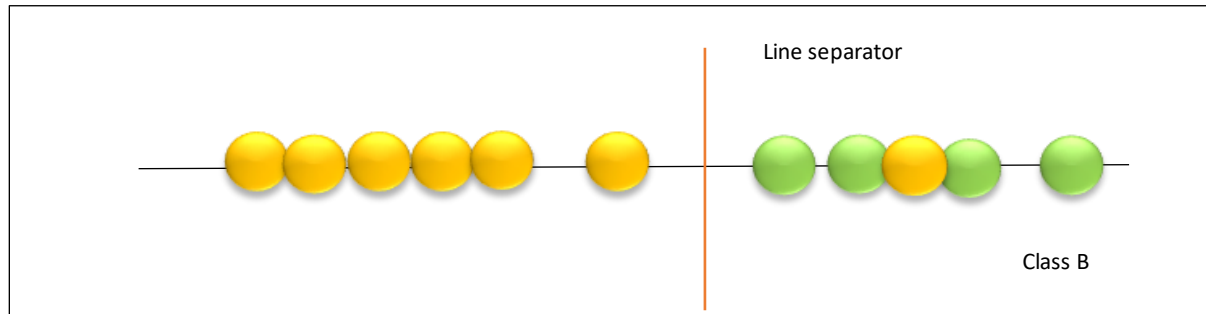


Fig 9 Linear vectors

A kernel function is used to guide the support vector machine which is given as: -

$$z = x^2 + y^2$$

Kernels are initially linear which can work with linear variables but if non-linear kernels are required, they can be created using

$$a \cdot b = xa \cdot xb + ya \cdot yb + za \cdot zb_2$$

$$a \cdot b = xa \cdot xb + ya \cdot yb + (xa^2 + ya^2) \cdot (xb^2 + yb^2)$$

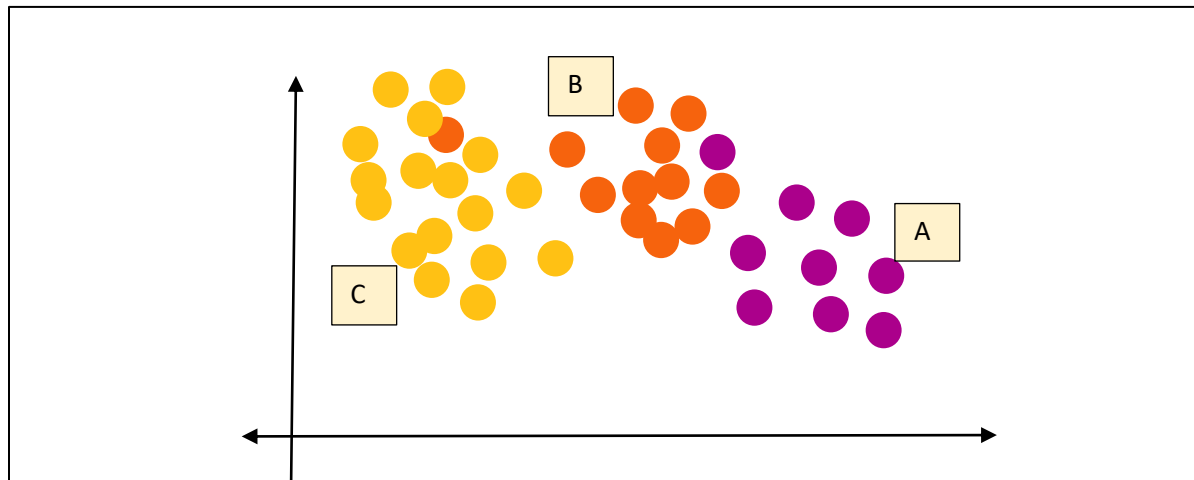


Fig 10 Multi linear vectors

IMPLEMENTATION

IMPORTING LIBRARIES

Here for the classification of Alzheimer's disease we are using Machine Learning classifiers like Logistic Regression, Random Forest and Support Vector Machine. Hence necessary libraries are to be imported. In the similar way for the CNN model also we need to import required packages.

For ML classifiers:

```
import pandas as pd
import numpy as np
import seaborn as sns
```

For CNN Model:

```
import pandas as pd
import numpy as np
import seaborn as sns
import tensorflow as tf
```

DATASET LOADING

For ML classifiers:

```
df = pd.read_csv('../input/oasis_longitudinal.csv')
df.head()
```

For CNN Model:

```
base_dir = "../input/alzheimer/Alzheimer_s Dataset"
root_dir = "/"
test_dir = "../input/alzheimer/Alzheimer_s Dataset/test/"
train_dir = "../input/alzheimer/Alzheimer_s Dataset/train/"
work_dir = root_dir + "dataset/"
```

DATA PREPROCESSING

For ML classifier:

Removing rows with missing values

This can be done with the help of a function called dropna

Syntax: pandas.DataFrame.dropna(axis = 0, how = 'any', thresh = None, subset = None, inplace=False)

Imputation

Imputation is a method for replacing missing data with a substitute value while retaining the majority of the dataset's data/information. These strategies are utilized because eliminating data from a dataset every time is impractical and might result in a significant reduction in the dataset's size, which not only raises issues about biasing the dataset but also results in inaccurate analysis.

Splitting Train/Validation/Test Sets

Train Dataset: Used to fit the machine learning model.

Test Dataset: Used to evaluate the fit machine learning model.

Validate Dataset: From the test dataset a part is taken for validation

Cross-validation

To determine the best parameters for each model (Logistic Regression, SVM, Random Forests), we use 5-fold cross-validation. We find the best hyperparameters by accuracy because our performance metric is accuracy. Finally, we compare each model's accuracy, recall, and AUC.

For CNN Model:

Image Augmentation

Image augmentation is a technique of altering the existing data to create some more data for the model training process. There are different types of Augmentation techniques. They are:

- Image Rotation
- Image Shifting
- Image Flipping
- Image Noising
- Image Blurring

PERFORMANCE MEASURES:

The evaluation criteria for measuring the performance of the machine learning algorithm can be called as the performance metrics. The need for metrics arises when one needs to compare the performance of different algorithms. It also weighs on selecting the representation of the results. There is a wide range of performance metrics available; Confusion matrix, accuracy metrics, precision, F1 score etc.

Confusion matrix:

The easiest way of checking the performance of an algorithm is by generating the confusion matrix. It is done by creating a table of four components. True positive, False Positive, True Negative and False Negative.

		Actual	
		1	0
Predicted	1	True positive	False Positive
	0	False negative	True Negative

Let's understand the terms: -

- 1) **True positives (TP)** – When the output of the actual class and the predicted class is 1, it is called a true positive.
- 2) **True Negatives (TN)** - When the output of the actual class and the predicted class is 0, it is a true negative.
- 3) **False Positive (FP)** – When the value of actual class is 0 and the predicted class is 1, it is called False Positive.
- 4) **False Negative (FN)** – when the value of actual class is 1 and the predicted class is 0, it is called False negative.

(i) Accuracy: -

From the confusion matrix, the accuracy of the algorithm can be determined.

It can be defined as the ration of number of correct predictions to the total number of predictions.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

(ii) Precision: -

It can be defined as the number of correct documents returned by the ML model. It is calculated using the confusion matrix as well.

$$\text{Precision} = \frac{TP}{TP+FP}$$

(iii) Recall or sensitivity: -

It is defined as the number of positives returned by the ml model, which can be calculated by using the confusion matrix.

$$\text{Recall} = \frac{TP}{TP+FN}$$

(iv) Specificity: -

In contrast to the sensitivity, specificity is the number of negatives returned by the ml models.

$$\text{Specificity} = \frac{TN}{TN+FP}$$

(v) F1 Score: -

It can be calculated using the harmonic mean of precision and recall. The values that F1 score can take is (1,0); with 1 being the best and 0 being the worst score.

$$F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

GRAPHICAL USER INTERFACE(GUI)

FRONT END DEVELOPMENT

Front end development is mostly focused on what some may coin the "client side" of development. Front end developers will be engaged in analyzing code, design, and debugging applications along with ensuring a seamless user experience. In this project we have used HTML, CSS and JavaScript for front-end website.

HTML

HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications. It is easy, simple and Platform independent language.

CSS

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML.

JavaScript

JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else.

DATABASE CONNECTIVITY

Backend was connected using flask which is small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier.

The model was saved with the help of pickle module.

Python pickle module is used for serializing and de-serializing python object structures. The process to convert any kind of python objects (list, dict, etc.) into byte streams (0s and 1s).

RESULTS

Performance metrics for Logistic Regression Classifier

```
m = 'Logistic Regression (w/ dropna)'
acc.append([m, test_score, test_recall, test_recall, fpr, tpr, thresholds])
```

```
↳ Best accuracy on validation set is: 0.7551920341394027
Best parameter for regularization (C) is: 1
Test accuracy with best C parameter is 0.7777777777777778
Test recall with the best C parameter is 0.6666666666666666
Test AUC with the best C parameter is 0.7976190476190476
```

Fig 12 Performance Metrics for Logistic Regression

Performance metrics for Support Vector Machine Classifier

```
m = 'SVM'
acc.append([m, test_score, test_recall, test_auc, fpr, tpr, thresholds])
```

```
Best accuracy on cross validation set is: 0.7866666666666667
Best parameter for c is: 100
Best parameter for gamma is: 10
Best parameter for kernel is: rbf
Test accuracy with the best parameters is 0.9090909090909091
Test recall with the best parameters is 0.9166666666666666
Test recall with the best parameter is 0.9107142857142856
```

Fig 13 Performance Metrics for Support Vector Machine

Performance metrics for Random Forest Classifier

```
m = 'Random Forest'
acc.append([m, test_score, test_recall, test_auc, fpr, tpr, thresholds])
```

```
Best accuracy on validation set is: 0.8882051282051282
Best parameters of M, d, m are: 14 6 8
Test accuracy with the best parameters is 0.9393939393939394
Test recall with the best parameters is: 0.9166666666666666
Test AUC with the best parameters is: 0.9345238095238094
```

Fig 14 Performance metrics for Random Forest

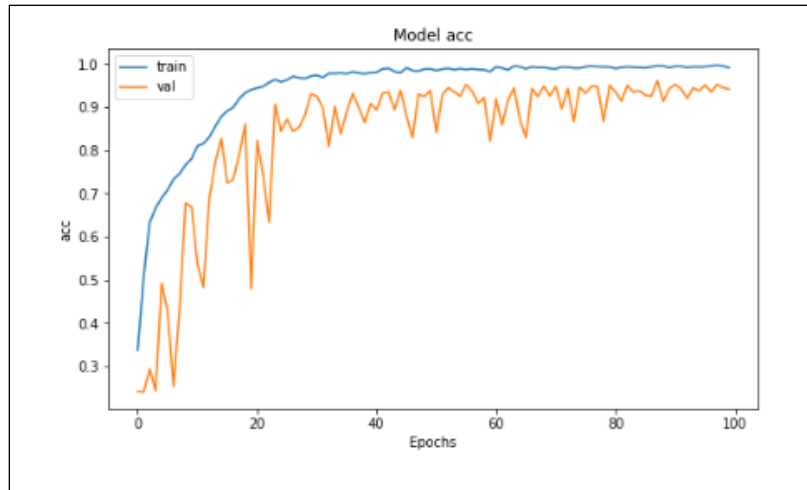


Fig 15 Acc graph for CNN Model

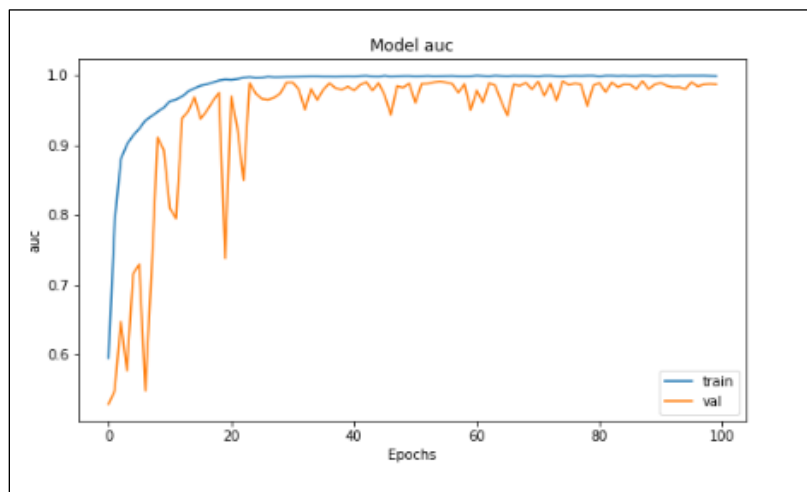


Fig 16 AUC graph for CNN Model

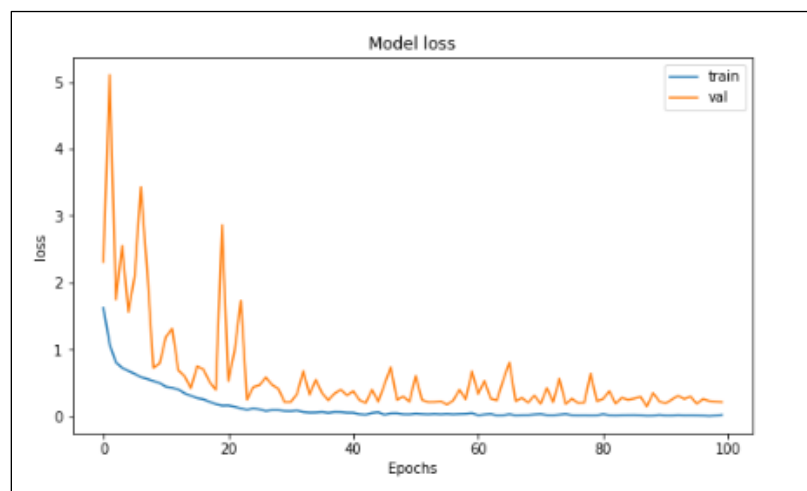


Fig 17 Model Loss graph for CNN

```
#print("Training Accuracy: %.2f%%"%(train_scores[1] * 100))
#print("Validation Accuracy: %.2f%%"%(val_scores[1] * 100))
print("Testing Accuracy: %.2f%%"%(test_scores[1] * 100))
```

2022-04-24 06:06:23.190086: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 951582720 exceeds 10% of free system memory.

80/80 [=====] - 1s 12ms/step - loss: 0.2304 - acc: 0.9457 - auc: 0.9877 - f1_score: 0.9454
Testing Accuracy: 94.57%

Fig 18 Testing Accuracy for CNN Model

```
print(classification_report(test_labels, pred_labels, target_names=CLASSES))
```

	precision	recall	f1-score	support
NonDemented	0.95	0.98	0.97	639
VeryMildDemented	1.00	1.00	1.00	635
MildDemented	0.90	0.93	0.92	662
ModerateDemented	0.93	0.87	0.90	624
micro avg	0.95	0.95	0.95	2560
macro avg	0.95	0.95	0.95	2560
weighted avg	0.95	0.95	0.95	2560
samples avg	0.95	0.95	0.95	2560

Fig 19 Confusion matrix and Classification Report for CNN Model

CATEGORIZATION AND COMPARISON OF ALZHEIMER'S DISEASE USING ML AND CNN

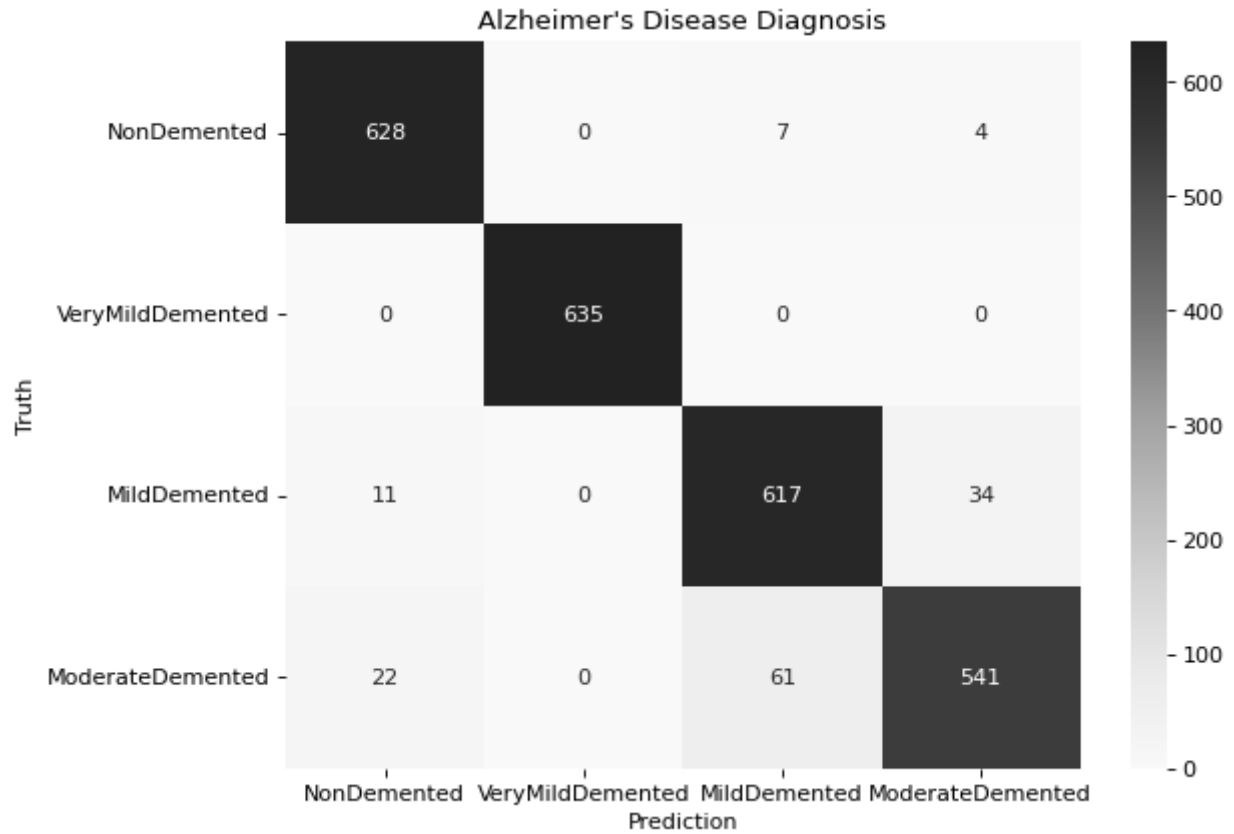


Fig 20 Confusion Matrix

	Precision	Accuracy	F1 score	Recall
SVM	0.910	0.89	0.90	0.88
RF	0.89	0.87	0.850	0.82
LR	0.86	0.82	0.81	0.86
CNN	0.95	0.94	0.97	0.98

Table 2 Values of performance matrix of all algorithms

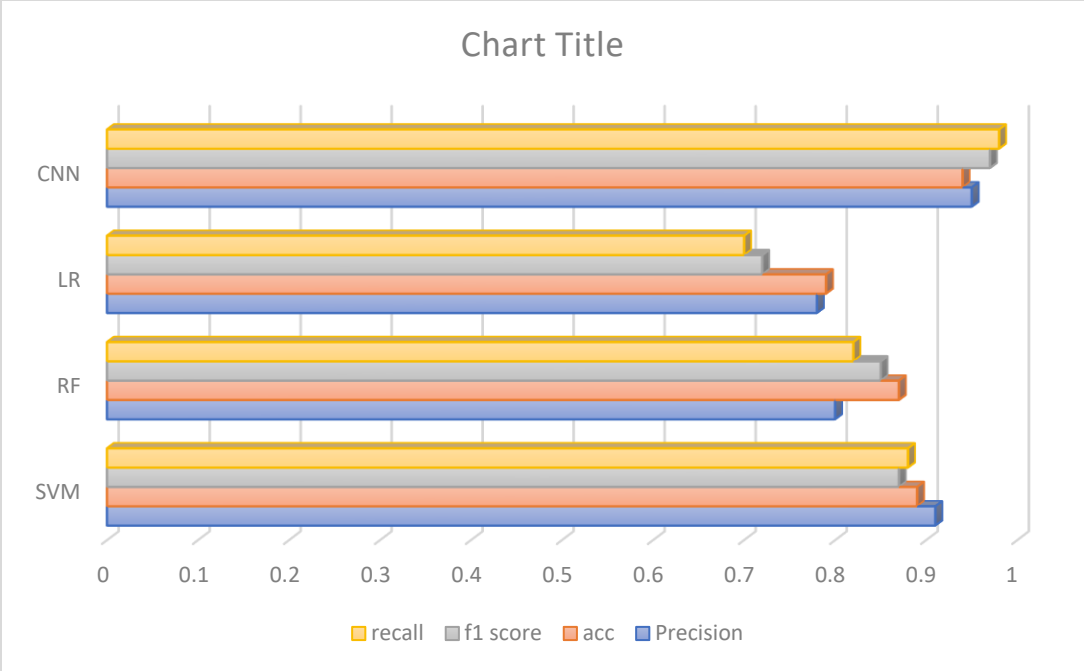
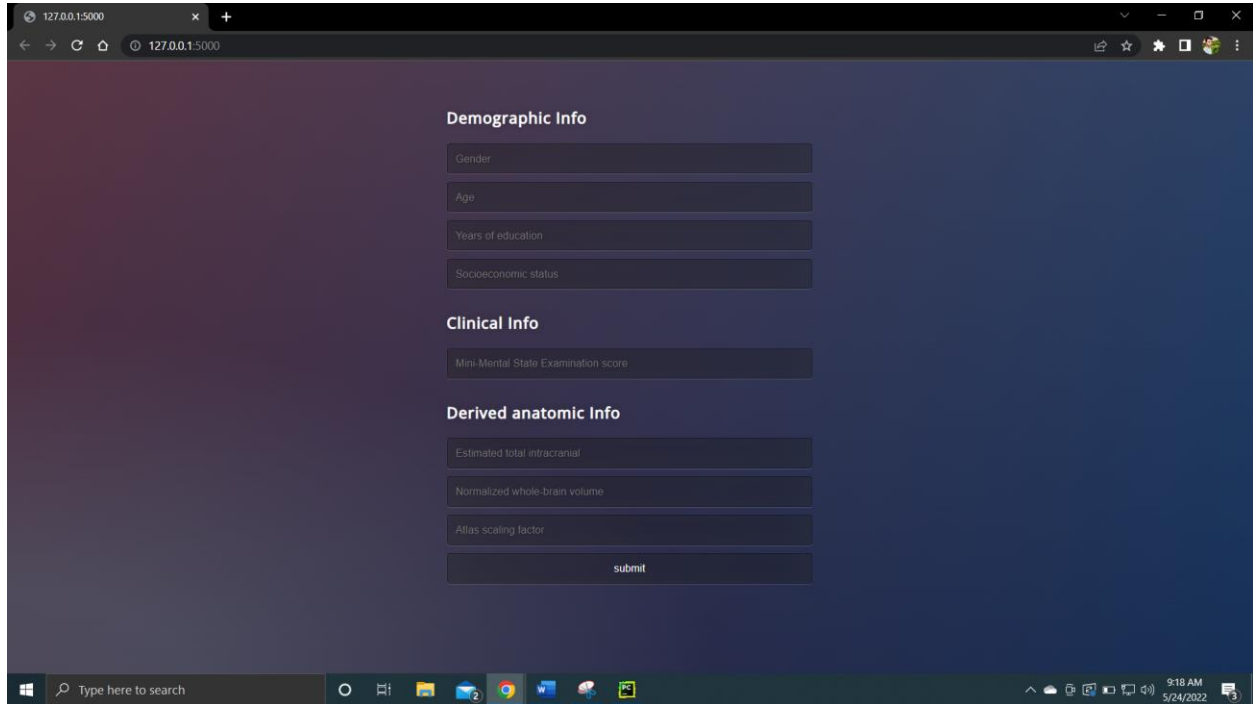


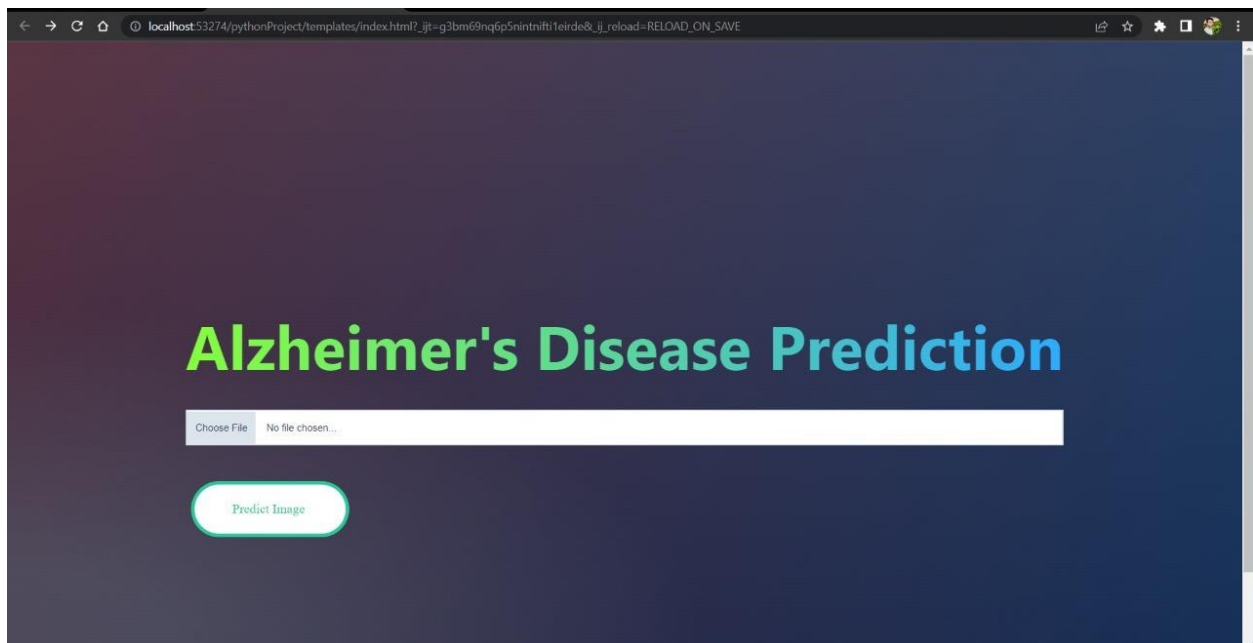
Fig 21 Bar Graph representing all metrics of ML classifiers and CNN Model

GRAPHICAL USER INTERFACE



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000'. The page has a dark blue gradient background. It contains three sections of input fields: 'Demographic Info' with fields for Gender, Age, Years of education, and Socioeconomic status; 'Clinical Info' with a field for Mini-Mental State Examination score; and 'Derived anatomic Info' with fields for Estimated total intracranial, Normalized whole-brain volume, and Atlas scaling factor. A 'submit' button is located at the bottom of the derived anatomic info section. The Windows taskbar is visible at the bottom of the screen.

Fig 22 GUI for ML Classifiers



The screenshot shows a web browser window with the address bar displaying 'localhost:53274/pythonProject/templates/index.html?_ijt=g3bm69nq6p5nintnlt1eirde&_ll_reload=RELOAD_ON_SAVE'. The page has a dark blue gradient background. It features the title 'Alzheimer's Disease Prediction' in large, bold, green and blue text. Below the title is a file upload section with a 'Choose File' button and a text box showing 'No file chosen...'. At the bottom of this section is a green 'Predict Image' button.

Fig 23 GUI for CNN Model

CONCLUSION AND FUTURE SCOPE

The comparison between performances of ML models and CNN model is depicted using the accuracies generated. The values were standardized to make sure that they easily fit in the ML models. Then the dataset has been used to train SVM, logistic regression and random forest models. For evaluation metrics, accuracy, recall, AUC, and confusion matrix have been used. SVM generates higher accuracy than the other ML models. Whereas CNN model has generated more accuracy overall. Hence it can be concluded that CNN is the most efficient model that can be used to predict the occurrence of the early stages of Alzheimer's Disease. This disease effects 1 in 100 people. But then again, the treatment doesn't guarantee that it can be cured. If the ailment advances too much, the patient might get into a zone where they can't even figure out the basic activities.

The system models could be enhanced in the future by employing a larger dataset and more machine learning models like AdaBoost, KNN, Majority Voting, and Bagging. The system's dependability and performance will improve as a result of this. By simply inputting MRI data, the ML system can assist the general public in gaining an understanding of the probability of Dementia in adult patients.

APPENDIX

ml.ipynb

```

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

sns.set()

df = pd.read_csv('oasis_longitudinal.csv')

df.head()

df = df.loc[df['Visit']==1] # use first visit data only because of the analysis we're doing

df = df.reset_index(drop=True) # reset index after filtering first visit data

df['M/F'] = df['M/F'].replace(['F','M'], [0,1]) # M/F column

df['Group'] = df['Group'].replace(['Converted'], ['Demented']) # Target variable

df['Group'] = df['Group'].replace(['Demented', 'Nondemented'], [1,0]) # Target variable

df = df.drop(['MRI ID', 'Visit', 'Hand'], axis=1) # Drop unnecessary columns

# bar drawing function

def bar_chart(feature):

    Demented = df[df['Group']==1][feature].value_counts()

    Nondemented = df[df['Group']==0][feature].value_counts()

    df_bar = pd.DataFrame([Demented,Nondemented])

    df_bar.index = ['Demented','Nondemented']

    df_bar.plot(kind='bar',stacked=True, figsize=(8,5))

```

```
# Gender and Group ( Femal=0, Male=1)

bar_chart('M/F')

plt.xlabel('Group')

plt.ylabel('Number of patients')

plt.legend()

plt.title('Gender and Demented rate')

#MMSE : Mini Mental State Examination

# Nondemented = 0, Demented =1

# Nondemented has higher test result ranging from 25 to 30.

#Min 17 ,MAX 30

facet= sns.FacetGrid(df,hue="Group", aspect=3)

facet.map(sns.kdeplot,'MMSE',shade= True)

facet.set(xlim=(0, df['MMSE'].max()))

facet.add_legend()

plt.xlim(15,30)

#bar_chart('ASF') = Atlas Scaling Factor

facet= sns.FacetGrid(df,hue="Group", aspect=3)

facet.map(sns.kdeplot,'ASF',shade= True)

facet.set(xlim=(0, df['ASF'].max()))

facet.add_legend()

plt.xlim(0.5, 2)

#eTIV = Estimated Total Intracranial Volume

facet= sns.FacetGrid(df,hue="Group", aspect=3)
```

```

facet.map(sns.kdeplot,'eTIV',shade= True)

facet.set(xlim=(0, df['eTIV'].max()))

facet.add_legend()

plt.xlim(900, 2100)

#'nWBV' = Normalized Whole Brain Volume

# Nondemented = 0, Demented =1

facet= sns.FacetGrid(df,hue="Group", aspect=3)

facet.map(sns.kdeplot,'nWBV',shade= True)

facet.set(xlim=(0, df['nWBV'].max()))

facet.add_legend()

plt.xlim(0.6,0.9)

#AGE. Nondemented =0, Demented =0

facet= sns.FacetGrid(df,hue="Group", aspect=3)

facet.map(sns.kdeplot,'Age',shade= True)

facet.set(xlim=(0, df['Age'].max()))

facet.add_legend()

plt.xlim(50,100)

#'EDUC' = Years of Education

# Nondemented = 0, Demented =1

facet= sns.FacetGrid(df,hue="Group", aspect=3)

facet.map(sns.kdeplot,'EDUC',shade= True)

facet.set(xlim=(df['EDUC'].min(), df['EDUC'].max()))

facet.add_legend()

```

```
plt.ylim(0, 0.16)

# Check missing values by each column

pd.isnull(df).sum()

# The column, SES has 8 missing values

# Dropped the 8 rows with missing values in the column, SES

df_dropna = df.dropna(axis=0, how='any')

pd.isnull(df_dropna).sum()

df_dropna['Group'].value_counts()

# Draw scatter plot between EDUC and SES

x = df['EDUC']

y = df['SES']

ses_not_null_index = y[~y.isnull()].index

x = x[ses_not_null_index]

y = y[ses_not_null_index]

# Draw trend line in red

z = np.polyfit(x, y, 1)

p = np.poly1d(z)

plt.plot(x, y, 'go', x, p(x), "r--")

plt.xlabel('Education Level(EDUC)')

plt.ylabel('Social Economic Status(SES)')

plt.show()

df.groupby(['EDUC'])['SES'].median()

df["SES"].fillna(df.groupby("EDUC")["SES"].transform("median"), inplace=True)
```



```
# I confirm there're no more missing values and all the 150 data were used.

pd.isnull(df['SES']).value_counts()

from sklearn.model_selection import train_test_split

from sklearn import preprocessing

from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import cross_val_score

# Dataset with imputation

Y = df['Group'].values # Target for the model

X = df[['M/F', 'Age', 'EDUC', 'SES', 'MMSE', 'eTIV', 'nWBV', 'ASF']] # Features we use

# splitting into three sets

X_trainval, X_test, Y_trainval, Y_test = train_test_split(

    X, Y, random_state=0)

# Feature scaling

scaler = MinMaxScaler().fit(X_trainval)

X_trainval_scaled = scaler.transform(X_trainval)

X_test_scaled = scaler.transform(X_test)

# Dataset after dropping missing value rows

Y = df_dropna['Group'].values # Target for the model

X = df_dropna[['M/F', 'Age', 'EDUC', 'SES', 'MMSE', 'eTIV', 'nWBV', 'ASF']] # Features we use

# splitting into three sets

X_trainval_dna, X_test_dna, Y_trainval_dna, Y_test_dna = train_test_split(

    X, Y, random_state=0)
```

```
# Feature scaling

scaler = MinMaxScaler().fit(X_trainval_dna)

X_trainval_scaled_dna = scaler.transform(X_trainval_dna)

X_test_scaled_dna = scaler.transform(X_test_dna)

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import confusion_matrix, accuracy_score, recall_score, roc_curve, auc

acc = [] # list to store all performance metric

# Dataset after dropping missing value rows

best_score=0

kfold=5 # set the number of folds

for c in [0.001, 0.1, 1, 10, 150]:

    logRegModel = LogisticRegression(C=c)

    # perform cross-validation

    scores = cross_val_score(logRegModel, X_trainval_scaled_dna, Y_trainval_dna, cv=kfold,
scoring='accuracy')

    # compute mean cross-validation accuracy

    score = np.mean(scores)

    # Find the best parameters and score

    if score > best_score:

        best_score = score
```

```

best_parameters = c

# rebuild a model on the combined training and validation set

SelectedLogRegModel = LogisticRegression(C=best_parameters).fit(X_trainval_scaled_dna,
Y_trainval_dna)

test_score = SelectedLogRegModel.score(X_test_scaled_dna, Y_test_dna)

PredictedOutput = SelectedLogRegModel.predict(X_test_scaled)

test_recall = recall_score(Y_test, PredictedOutput, pos_label=1)

fpr, tpr, thresholds = roc_curve(Y_test, PredictedOutput, pos_label=1)

test_auc = auc(fpr, tpr)

print("Best accuracy on validation set is:", best_score)

print("Best parameter for regularization (C) is: ", best_parameters)

print("Test accuracy with best C parameter is", test_score)

print("Test recall with the best C parameter is", test_recall)

print("Test AUC with the best C parameter is", test_auc)

m = 'Logistic Regression (w/ dropna)'

acc.append([m, test_score, test_recall, test_recall, fpr, tpr, thresholds])

best_score = 0

for c_paramter in [0.001, 0.01, 0.1, 1, 10, 100, 1000]: #iterate over the values we need to try for
the parameter C

    for gamma_paramter in [0.001, 0.01, 0.1, 1, 10, 100, 1000]: #iterate over the values we need to
try for the parameter gamma

        for k_parameter in ['rbf', 'linear', 'poly', 'sigmoid']: # iterate over the values we need to try for
the kernel parameter

            svmModel = SVC(kernel=k_parameter, C=c_paramter, gamma=gamma_paramter)
#define the model

            # perform cross-validation

```

```
scores = cross_val_score(svmModel, X_trainval_scaled, Y_trainval, cv=kfolds,
scoring='accuracy')

# the training set will be split internally into training and cross validation

# compute mean cross-validation accuracy

score = np.mean(scores)

# if we got a better score, store the score and parameters

if score > best_score:

    best_score = score #store the score

    best_parameter_c = c_paramter #store the parameter c

    best_parameter_gamma = gamma_paramter #store the parameter gamma

    best_parameter_k = k_parameter

# rebuild a model with best parameters to get score

SelectedSVMmodel = SVC(C=best_parameter_c, gamma=best_parameter_gamma,
kernel=best_parameter_k).fit(X_trainval_scaled, Y_trainval)

test_score = SelectedSVMmodel.score(X_test_scaled, Y_test)

PredictedOutput = SelectedSVMmodel.predict(X_test_scaled)

test_recall = recall_score(Y_test, PredictedOutput, pos_label=1)

fpr, tpr, thresholds = roc_curve(Y_test, PredictedOutput, pos_label=1)

test_auc = auc(fpr, tpr)

print("Best accuracy on cross validation set is:", best_score)

print("Best parameter for c is: ", best_parameter_c)

print("Best parameter for gamma is: ", best_parameter_gamma)

print("Best parameter for kernel is: ", best_parameter_k)

print("Test accuracy with the best parameters is", test_score)
```

```
print("Test recall with the best parameters is", test_recall)

print("Test recall with the best parameter is", test_auc)

m = 'SVM'

acc.append([m, test_score, test_recall, test_auc, fpr, tpr, thresholds])

best_score = 0

for M in range(2, 15, 2): # combines M trees

    for d in range(1, 9): # maximum number of features considered at each split

        for m in range(1, 9): # maximum depth of the tree

            # train the model

            # n_jobs(4) is the number of parallel computing

            forestModel = RandomForestClassifier(n_estimators=M, max_features=d, n_jobs=4,

                                                max_depth=m, random_state=0)

            # perform cross-validation

            scores = cross_val_score(forestModel, X_trainval_scaled, Y_trainval, cv=kfolds,

                                     scoring='accuracy')

            # compute mean cross-validation accuracy

            score = np.mean(scores)

            # if we got a better score, store the score and parameters

            if score > best_score:

                best_score = score

                best_M = M

                best_d = d

                best_m = m
```

```
# Rebuild a model on the combined training and validation set

SelectedRFModel = RandomForestClassifier(n_estimators=M, max_features=d,

                                         max_depth=m, random_state=0).fit(X_trainval_scaled, Y_trainval )

PredictedOutput = SelectedRFModel.predict(X_test_scaled)

test_score = SelectedRFModel.score(X_test_scaled, Y_test)

test_recall = recall_score(Y_test, PredictedOutput, pos_label=1)

fpr, tpr, thresholds = roc_curve(Y_test, PredictedOutput, pos_label=1)

test_auc = auc(fpr, tpr)

print("Best accuracy on validation set is:", best_score)

print("Best parameters of M, d, m are: ", best_M, best_d, best_m)

print("Test accuracy with the best parameters is", test_score)

print("Test recall with the best parameters is:", test_recall)

print("Test AUC with the best parameters is:", test_auc)

m = 'Random Forest'

acc.append([m, test_score, test_recall, test_auc, fpr, tpr, thresholds])

print("Feature importance: ")

np.array([X.columns.values.tolist(), list(SelectedRFModel.feature_importances_)]).T

# Performance Metric for each model

result = pd.DataFrame(acc, columns=['Model', 'Accuracy', 'Recall', 'AUC', 'FPR', 'TPR', 'TH'])

result[['Model', 'Accuracy', 'Recall', 'AUC']]
```

cnn.ipynb

```
import numpy as np

import pandas as pd

import seaborn as sns

import tensorflow as tf

import matplotlib.pyplot as plt

import os

from distutils.dir_util import copy_tree, remove_tree

from PIL import Image

from random import randint

from imblearn.over_sampling import SMOTE

from sklearn.model_selection import train_test_split

from sklearn.metrics import matthews_corrcoef as MCC

from sklearn.metrics import balanced_accuracy_score as BAS

from sklearn.metrics import classification_report, confusion_matrix

import tensorflow_addons as tfa

from keras.utils.vis_utils import plot_model

from tensorflow.keras import Sequential, Input

from tensorflow.keras.layers import Dense, Dropout

from tensorflow.keras.layers import Conv2D, Flatten

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras.applications.inception_v3 import InceptionV3
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator as IDG

from tensorflow.keras.layers import SeparableConv2D, BatchNormalization, MaxPool2D

print("TensorFlow Version:", tf.__version__)

base_dir = "../input/alzheimer/Alzheimer_s Dataset"

root_dir = "/"

test_dir = "../input/alzheimer/Alzheimer_s Dataset/test/"

train_dir = "../input/alzheimer/Alzheimer_s Dataset/train/"

work_dir = root_dir + "dataset/"

if os.path.exists(work_dir):

    remove_tree(work_dir)

os.mkdir(work_dir)

copy_tree(train_dir, work_dir)

copy_tree(test_dir, work_dir)

print("Working Directory Contents:", os.listdir(work_dir))

WORK_DIR = './dataset/'

CLASSES = [ 'NonDemented',

            'VeryMildDemented',

            'MildDemented',

            'ModerateDemented']

IMG_SIZE = 176

IMAGE_SIZE = [176, 176]

DIM = (IMG_SIZE, IMG_SIZE)

#Performing Image Augmentation to have more data samples
```


ZOOM = [.99, 1.01]

BRIGHT_RANGE = [0.8, 1.2]

HORZ_FLIP = True

FILL_MODE = "constant"

DATA_FORMAT = "channels_last"

work_dr = IDG(rescale = 1./255, brightness_range=BRIGHT_RANGE, zoom_range=ZOOM, data_format=DATA_FORMAT, fill_mode=FILL_MODE, horizontal_flip=HORZ_FLIP)

train_data_gen = work_dr.flow_from_directory(directory=WORK_DIR, target_size=DIM, batch_size=6500, shuffle=False)

def show_images(generator,y_pred=None):

"""

Input: An image generator,predicted labels (optional)

Output: Displays a grid of 9 images with lables

"""

get image lables

labels =dict(zip([0,1,2,3], CLASSES))

get a batch of images

x,y = generator.next()

display a grid of 9 images

plt.figure(figsize=(10, 10))

if y_pred is None:

for i in range(9):

ax = plt.subplot(3, 3, i + 1)

idx = randint(0, 6400)

```

plt.imshow(x[idx])

plt.axis("off")

plt.title("Class: {}".format(labels[np.argmax(y[idx])]))

else:

    for i in range(9):

        ax = plt.subplot(3, 3, i + 1)

        plt.imshow(x[i])

        plt.axis("off")

        plt.title("Actual: {} \n Predicted: {}".format(labels[np.argmax(y[i])], labels[y_pred[i]]))

# Display Train Images

show_images(train_data_gen)

#Retrieving the data from the ImageDataGenerator iterator

train_data, train_labels = train_data_gen.next()

#Getting to know the dimensions of our dataset

print(train_data.shape, train_labels.shape)

#Performing over-sampling of the data, since the classes are imbalanced

sm = SMOTE(random_state=42)

train_data, train_labels = sm.fit_resample(train_data.reshape(-1, IMG_SIZE * IMG_SIZE * 3),
train_labels)

train_data = train_data.reshape(-1, IMG_SIZE, IMG_SIZE, 3)

print(train_data.shape, train_labels.shape)

#Splitting the data into train, test, and validation sets

train_data, test_data, train_labels, test_labels = train_test_split(train_data, train_labels, test_size =
0.2, random_state=42)

```

```
train_data, val_data, train_labels, val_labels = train_test_split(train_data, train_labels, test_size = 0.2, random_state=42)
```

```
def conv_block(filters, act='relu'):
```

```
    """Defining a Convolutional NN block for a Sequential CNN model. """
```

```
    block = Sequential()
```

```
    block.add(Conv2D(filters, 3, activation=act, padding='same'))
```

```
    block.add(Conv2D(filters, 3, activation=act, padding='same'))
```

```
    block.add(BatchNormalization())
```

```
    block.add(MaxPool2D())
```

```
    return block
```

```
def dense_block(units, dropout_rate, act='relu'):
```

```
    """Defining a Dense NN block for a Sequential CNN model. """
```

```
    block = Sequential()
```

```
    block.add(Dense(units, activation=act))
```

```
    block.add(BatchNormalization())
```

```
    block.add(Dropout(dropout_rate))
```

```
    return block
```

```
def construct_model(act='relu'):
```

```
    """Constructing a Sequential CNN architecture for performing the classification task. """
```

```
    model = Sequential([
```

```
        Input(shape=(*IMAGE_SIZE, 3)),
```

```
        Conv2D(16, 3, activation=act, padding='same'),
```

```
        Conv2D(16, 3, activation=act, padding='same'),
```

```
        MaxPool2D(),
```

```
conv_block(32),
conv_block(64),
conv_block(128),
Dropout(0.2),
conv_block(256),
Dropout(0.2),
Flatten(),
dense_block(512, 0.7),
dense_block(128, 0.5),
dense_block(64, 0.3),
Dense(4, activation='softmax')
], name = "cnn_model")

return model

#Defining a custom callback function to stop training our model when accuracy goes above 99%
class MyCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if logs.get('val_acc') > 0.99:
            print("\nReached accuracy threshold! Terminating training.")
            self.model.stop_training = True

my_callback = MyCallback()

#EarlyStopping callback to make sure model is always learning
early_stopping = EarlyStopping(monitor='val_loss', patience=2)

#Defining other parameters for our CNN model
```

```
model = construct_model()

METRICS = [tf.keras.metrics.CategoricalAccuracy(name='acc'),

            tf.keras.metrics.AUC(name='auc'),

            tfa.metrics.F1Score(num_classes=4)]

CALLBACKS = [my_callback]

model.compile(optimizer='adam',

              loss=tf.losses.CategoricalCrossentropy(),

              metrics=METRICS)

model.summary()

#Fit the training data to the model and validate it using the validation data

EPOCHS = 50

history = model.fit(train_data, train_labels, validation_data=(val_data, val_labels),
                    callbacks=CALLBACKS, epochs=EPOCHS)

#Plotting the trend of the metrics during training

fig, ax = plt.subplots(1, 3, figsize = (30, 5))

ax = ax.ravel()

for i, metric in enumerate(["acc", "auc", "loss"]):

    ax[i].plot(history.history[metric])

    ax[i].plot(history.history["val_" + metric])

    ax[i].set_title("Model {}".format(metric))

    ax[i].set_xlabel("Epochs")

    ax[i].set_ylabel(metric)

    ax[i].legend(["train", "val"])

#Evaluating the model on the data
```

```
#train_scores = model.evaluate(train_data, train_labels)

#val_scores = model.evaluate(val_data, val_labels)

test_scores = model.evaluate(test_data, test_labels)

#print("Training Accuracy: %.2f%%"%(train_scores[1] * 100))

#print("Validation Accuracy: %.2f%%"%(val_scores[1] * 100))

print("Testing Accuracy: %.2f%%"%(test_scores[1] * 100))

#Predicting the test data

pred_labels = model.predict(test_data)

#Print the classification report of the tested data

#Since the labels are softmax arrays, we need to roundoff to have it in the form of 0s and 1s,

#similar to the test_labels

def roundoff(arr):

    """To round off according to the argmax of each predicted label array. """

    arr[np.argmax(arr) != arr.max()] = 0

    arr[np.argmax(arr) == arr.max()] = 1

    return arr

for labels in pred_labels:

    labels = roundoff(labels)

print(classification_report(test_labels, pred_labels, target_names=CLASSES))

#Plot the confusion matrix to understand the classification in detail

pred_ls = np.argmax(pred_labels, axis=1)

test_ls = np.argmax(test_labels, axis=1)

conf_arr = confusion_matrix(test_ls, pred_ls)
```

```
plt.figure(figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k')

ax = sns.heatmap(conf_arr, cmap='Greens', annot=True, fmt='d', xticklabels=CLASSES,
yticklabels=CLASSES)

plt.title('Alzheimer\'s Disease Diagnosis')

plt.xlabel('Prediction')

plt.ylabel('Truth')

plt.show(ax)

#Printing some other classification metrics

print("Balanced Accuracy Score: { } %".format(round(BAS(test_ls, pred_ls) * 100, 2)))

print("Matthew's Correlation Coefficient: { } %".format(round(MCC(test_ls, pred_ls) * 100, 2)))

#Saving the model for future use

pickle.dump(model, open('alzheimer_model.pkl', 'wb'))

pretrained_model = tf.keras.models.load_model(model_dir)

#Check its architecture

plot_model(pretrained_model, to_file=work_dir + "model_plot.png", show_shapes=True,
show_layer_names=True)
```

index.html

```
<html>

<head>

</head>

<body>

<div class="login">

    <form action="/predict" method="POST">

        <h3>Demographic Info</h3>

        <input type="text" name="Gender" placeholder="Gender" required="required" />
```

```

        <!--<input type="text" name="Handedness" placeholder="Handedness" required="required"
/>--!>

        <input type="text" name="Age" placeholder="Age" required="required" />

        <input type="text" name="Years of education" placeholder="Years of education"
required="required" />

        <input type="text" name="Socioeconomic status" placeholder="Socioeconomic status"
required="required" />

        <h3>Clinical Info</h3>

        <input type="text" name="Mini-Mental State Examination score" placeholder="Mini-Mental
State Examination score" required="required" />

        <!--<input type="text" name="Clinical Dementia Rating" placeholder="Clinical Dementia
Rating" required="required" />--!>

        <h3>Derived anatomic Info</h3>

        <input type="text" name="Estimated total intracranial" placeholder="Estimated total
intracranial" required="required" />

        <input type="text" name="Normalized whole-brain volume" placeholder="Normalized
whole-brain volume" required="required" />

        <input type="text" name="Atlas scaling factor" placeholder="Atlas scaling factor"
required="required" />

        <input type="submit" name="Submit" value="submit" />

    </form>

</div>

</body>

</html>

```

Output.html

```

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Title</title>

</head>

```



```
<body>

{% if (answer) < 0.5 %}
<h1> no alzeimier </h1>
    {{ answer }}
{%else%}
<h1> alzheimer</h1>
    {{ answer }}
{%endif%}
</body>
</html>
```

app.py

```
from flask import Flask, render_template, request
import pickle
from sklearn.preprocessing import StandardScaler
import numpy as np
model = pickle.load(open('random_forest_model.pkl','rb'))
scaler = pickle.load(open('scaler.pkl', 'rb'))
app = Flask(__name__)
@app.route('/')
def home_page():
    return render_template('index.html')
@app.route('/predict', methods = ['POST'])
def predict():
    gender = request.form['Gender']
    #handedness = request.form['Handedness']
    age = request.form['Age']
    year_of_education = request.form['Years of education']
```

```
socio_economic_status = request.form['Socioeconomic status']
mini_mental_state_examination_score = request.form['Mini-Mental State Examination score']
#clinical_dementia_rating = request.form['Clinical Dementia Rating']
estimated_total_intracranial = request.form['Estimated total intracranial']
normalized_whole_brain_volume = request.form['Normalized whole-brain volume']
atlas_scaling_factor = request.form['Atlas scaling factor']
#input_data = [[np.array()]]
input_data = np.array([[gender, age, year_of_education,
                        socio_economic_status, mini_mental_state_examination_score,
estimated_total_intracranial,
                        normalized_whole_brain_volume, atlas_scaling_factor]])
pred = model.predict(input_data)
pred1 = scaler.transform(input_data)
print(pred)
return render_template('output.html', answer = pred1)

if __name__ == '__main__':
    app.run(debug = True)
```

REFERENCES

- 1 Tanveer, M., Richhariya, B., Khan, R. U., Rashid, A. H., Khanna, P., Prasad, M., & Lin, C. T. (2020). Machine learning techniques for the diagnosis of Alzheimer's disease. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 16(1s), 1–35.
- 2 Bron, E. E., Klein, S., Jiskoot, L. C., Linders, J., van Swieten, J. C., van der Flier, W. M., & van der Lugt, A. (2021). Cross-cohort generalizability of deep and conventional machine learning for MRI-based diagnosis and prediction of Alzheimer's disease. *NeuroImage: Clinical*, 31, 102712.
- 3 Kim, J. P., Kim, J., Park, Y. H., Park, S. N., Seo, S. W., & Seong, J.-K. (2019). Machine learning based hierarchical classification of frontotemporal dementia and Alzheimer's disease. *NeuroImage: Clinical*, 23, 101811.
- 4 Liu, M., Zhang, J., Nie, D., & Shen, D. (2018). Anatomical landmark based deep feature representation for MR images in brain disease diagnosis. *IEEE Journal of Biomedical and Health Informatics*, 22(5), 1476–1485.
- 5 Naik, B., Mehta, A., & Shah, M. (2020). Denouements of Machine Learning and Multimodal Diagnostic Classification of Alzheimer's disease. *Visual Computing for Industry, Biomedicine, and Art*, 3(1)
- 6 Mofrad, S. A., & Lundervold, A. S. (2021). A predictive framework based on brain volume trajectories enabling early detection of Alzheimer's disease. *Computerized Medical Imaging and Graphics*, 90, 101910.
- 7 Martinez-Murcia, F. J., Ortiz, A., Gorriz, & Castillo-Barnes, D. (2020). Studying the manifold structure of Alzheimer's disease: A deep learning approach using convolutional autoencoders. *IEEE Journal of Biomedical and Health Informatics*, 24(1), 17–26.
- 8 Al-Shoukry, S., Rassem, T. H., & Makbol, N. M. (2020). Alzheimer's diseases detection by using Deep Learning Algorithms: A mini-review. *IEEE Access*, 8, 77131–77141.
- 9 Ghoraani, B., L. N., Hssayeni, M. D., Rosenfeld, A., Tolea, M. I., & Galvin, J. E. (2021). Detection of mild cognitive impairment and Alzheimer's disease using dual-task gait assessments and machine learning. *Biomedical Signal Processing and Control*, 64, 102249.
- 10 Bi, X.-an, Hu, X., Wu, H., & Wang, Y. (2020). Multimodal Data Analysis of Alzheimer's

- disease based on Clustering Evolutionary Random Forest. *IEEE Journal of Biomedical and Health Informatics*, 24(10), 2973–2983.
- 11 Hazarika, R. A., Abraham, A., Sur, S. N., Maji, A. K., & Kandar, D. (2021). Different techniques for Alzheimer's disease classification using brain images: A study. *International Journal of Multimedia Information Retrieval*, 10(4), 199–218.
 - 12 Kruthika, K. R., Rajeswari, M., & Maheshappa, H. D. (2019). Multistage classifier-based approach for Alzheimer's disease prediction and retrieval. *Informatics in Medicine Unlocked*, 14, 34–42.
 - 13 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(2).
 - 14 Khan, N. M., Abraham, N., & Hon, M. (2019). Transfer learning with intelligent training data selection for prediction of Alzheimer's disease. *IEEE Access*, 7, 72726–72735.
 - 15 Fan, Z., Xu, F., Qi, X., Li, C., & Yao, L. (2019). Classification of Alzheimer's disease based on Brain MRI and machine learning. *Neural Computing and Applications*, 32(7), 1927–1936.
 - 16 Kishore, P., Usha Kumari, C., Kumar, M. N. V. S. S., & Pavani, T. (2021). Detection and analysis of alzheimer's disease using various machine learning algorithms. *Materials Today: Proceedings*, 45, 1502–1508.
 - 17 Albright, J. (2019). Forecasting the progression of alzheimer's disease using neural networks and a novel preprocessing algorithm. *Alzheimer's & Dementia: Translational Research & Clinical Interventions*, 5(1), 483–491.
 - 18 Noor, M. B., Zenia, N. Z., Kaiser, M. S., Mamun, S. A., & Mahmud, M. (2020). Application of deep learning in detecting neurological disorders from Magnetic Resonance Images: A survey on the detection of alzheimer's disease, parkinson's disease and schizophrenia. *Brain Informatics*, 7(1).
 - 19 Y., Bahgat, W. M., & Badawy, M. (2021). A CNN based framework for classification of alzheimer's disease. *Neural Computing and Applications*, 33(16), 10415–10428.
 - 20 Acharya, U. R., Fernandes, S. L., WeiKoh, J. E., Ciaccio, E. J., Fabell, M. K., Tanik, U. J., Rajinikanth, V., & Yeong, C. H. (2019). Automated detection of alzheimer's disease using brain MRI images– a study with various feature extraction techniques. *Journal of Medical Systems*, 43(9).

