SMART-INTERNZ INTERNSHIP AI&ML

PROJECT REPORT:

ALZHEMIER'S DISEASE PREDICTION

USING DEEP LEARNING

TEAM NO:

592710

TEAM MEMBERS: VIT-AP

21BCE7016 - NIKITHA SINDE

21BCE7071 – NIKSHEP ROHAN KURAPATI

21BCE7139 – SRI SAI VENKATA DURGA AKARSHA CHINTA

21BCE7151 - SALEMULA SOHITH VENKATA SAI GANESH

Project Report

ALZHEMIER'S DISEASE PREDICTION USING DEEP LEARNING

1. INTRODUCTION

1.1 Project Overview:

Utilizing AI and machine learning, our project aims to predict Alzheimer's disease onset through advanced data analysis. We'll harness diverse datasets, including brain imaging scans, genetic markers, and clinical records, to train predictive models. Leveraging machine learning algorithms, we'll extract patterns and correlations from these datasets.

Our focus lies in developing robust predictive models capable of early detection and risk assessment. Ethical considerations guide our handling of sensitive medical data, ensuring privacy and compliance. The project involves rigorous data preprocessing and feature extraction to enhance model accuracy. Through iterative model training and validation, we aim to achieve high predictive performance. Interpretability tools will aid in understanding model decisions, fostering trust in clinical applications. Continuous refinement and adaptation remain integral for real-world impact in Alzheimer's disease prediction. Collaboration with medical experts ensures alignment with clinical needs and standards.

1.2 Purpose:

Predicting Alzheimer's disease in its early stages holds immense significance on multiple fronts. Firstly, early detection allows for interventions that can significantly impact disease progression. It provides a window of opportunity for targeted therapies, lifestyle modifications, and cognitive interventions that might slow down the advancement of symptoms. Moreover, it empowers individuals and their families with critical information, enabling them to plan for the future, make informed decisions about care, and access support networks that specialize in Alzheimer's care and management. This early awareness can alleviate stress, facilitate better coping mechanisms, and foster a proactive approach to handling the challenges associated with the disease.

Additionally, early prediction aids in optimizing healthcare resource allocation. It allows for better planning within the healthcare system by directing appropriate care to those who need it most urgently. Moreover, early identification provides an opportunity for individuals to participate in clinical trials for experimental treatments or interventions, which not only benefits them directly but also contributes to advancing medical research and the development of more effective therapies for Alzheimer's in the long run.

Ultimately, leveraging deep learning for Alzheimer's prediction holds the promise of revolutionizing early detection, enhancing patient care, and advancing our understanding and management of this neurodegenerative disease.

2. LITERATURE SURVEY

2.1 Existing problem

Alzheimer's disease, a rapidly increasing condition globally, predominantly impacts the elderly. It's an incurable neurodegenerative ailment primarily targeting the brain. The challenge arises from its initial stages, often devoid of noticeable symptoms, leading to delayed or absent early diagnosis, resulting in difficulties for individuals coping with Alzheimer's-related issues. Alzheimer's disease prediction faces formidable challenges rooted in its elusive nature. Early detection remains a critical hurdle, as the disease often manifests insidiously with subtle cognitive changes that are difficult to differentiate from normal aging. Diagnostic tools, primarily reliant on clinical evaluations and cognitive assessments, lack the precision to detect the disease in its initial stages, impeding timely interventions. Moreover, the variability in symptom presentation among individuals adds complexity to creating unified prediction models. This variability demands a nuanced understanding of the disease's progression and complicates the development of accurate algorithms that can effectively forecast its onset and progression.

Compounding these challenges are limitations in data accessibility and quality. Predictive models heavily depend on comprehensive and diverse datasets that capture the multifaceted aspects of Alzheimer's, including genetic predispositions, lifestyle factors, and biomarkers. However, the availability of such datasets is often limited, impacting the accuracy and generalizability of predictive algorithms. Ethical concerns regarding data privacy and security add another layer of complexity, necessitating stringent measures to ensure the responsible handling of sensitive medical information required for effective prediction models. Addressing these multifaceted challenges demands collaborative efforts across disciplines, technological advancements, and a deeper understanding of the disease's underlying mechanisms.

2.2 References

Shahbaz, M., Ali, S., Guergachi, A., Niazi, A. and Umer, A., 2019, July. Classification of Alzheimer's Disease using Machine Learning Techniques. In *Data* (pp. 296-303).

Liu, Siqi, et al. "Early diagnosis of Alzheimer's disease with deep learning." *2014 IEEE 11th international symposium on biomedical imaging (ISBI)*. IEEE, 2014.

Alzheimer's Disease Detection using 3D ResNet-18 on MRI. (2021, May 6). Alzheimer's Disease Detection Using 3D ResNet-18 on MRI - File Exchange - MATLAB Central. https://www.mathworks.com/matlabcentral/fileexchange/88912-alzheimer-s-disease-detection-using-

3d-resnet-18-on-mri

A review of the application of deep learning in the detection of Alzheimer's disease. (2021, December 15). A Review of the Application of Deep Learning in the Detection of Alzheimer's Disease - ScienceDirect. https://doi.org/10.1016/j.ijcce.2021.12.002

Alzheimer Disease Detection Based on Deep Neural Network with Rectified Adam Optimization Technique using MRI Analysis. (n.d.). Alzheimer Disease Detection Based on Deep Neural Network With Rectified Adam Optimization Technique Using MRI Analysis | IEEE Conference Publication IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9339504

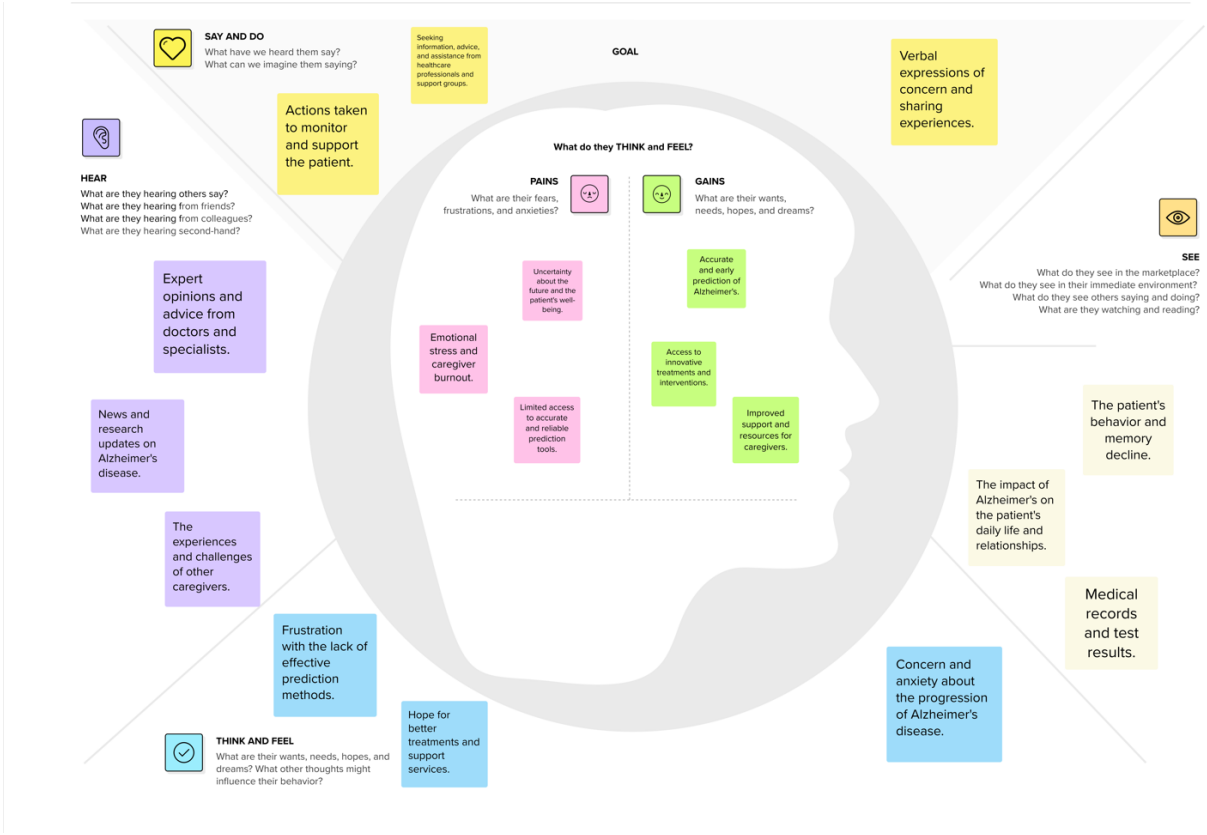
2.3 Problem Statement Definition

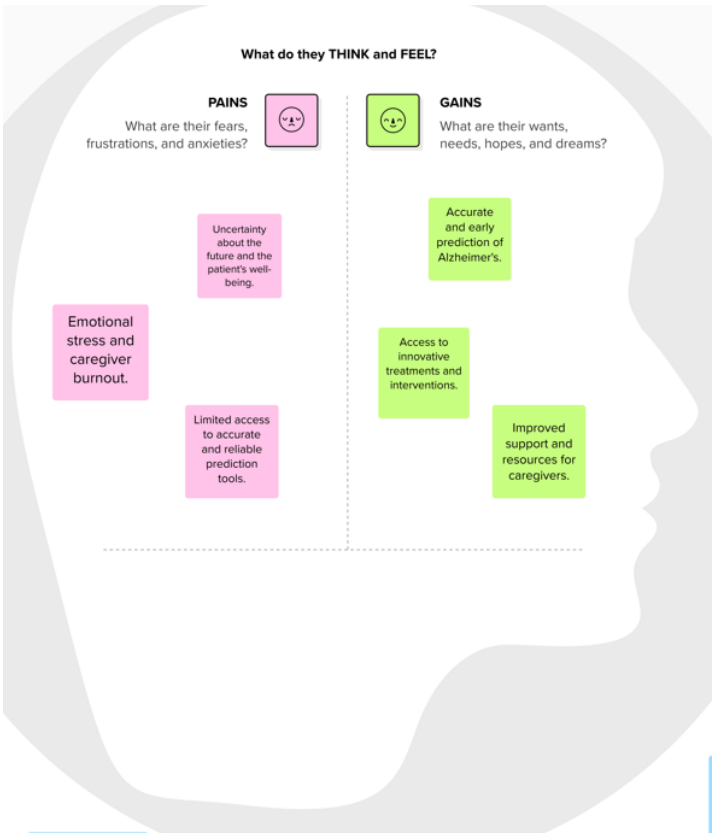
The problem statement "Alzheimer's disease prediction" refers to the challenge of developing accurate and reliable methods to forecast the onset, progression, and potential risk factors associated with Alzheimer's disease. This encompasses the creation of predictive models or algorithms that can identify individuals at risk of developing Alzheimer's or predict the disease's progression in those already diagnosed. The aim is to intervene early, potentially slowing down the disease's advancement or providing personalized care plans.

Early Detection Model for Persons with Probable Alzheimer's disease Using Deep learning algorithm. Alzheimer's disease (AD) is a prevalent and irreversible neurological condition marked by progressive cognitive decline. Over time, patients experience memory loss and deteriorating thinking abilities. Deep learning algorithms, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been employed to analyze various types of data, including neuroimaging, genetic markers, and clinical data, to detect patterns indicative of Alzheimer's disease.

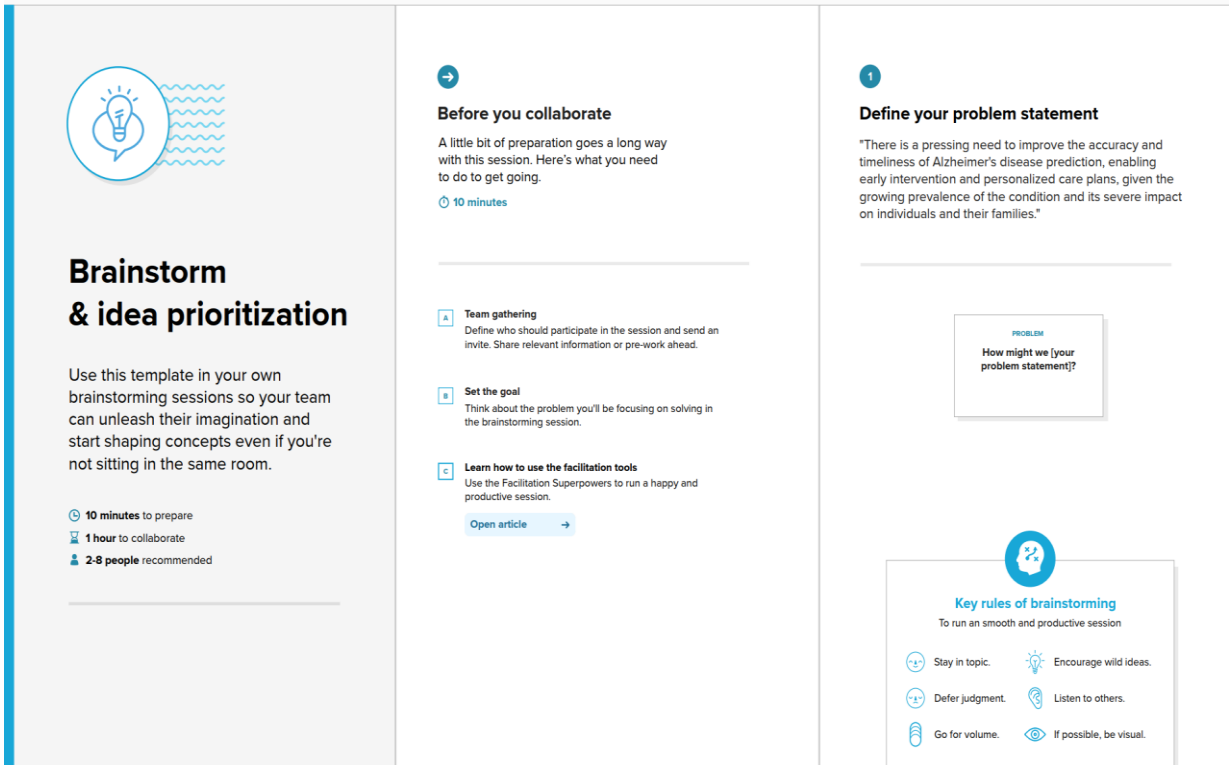
3. IDEATION & PROPOSED SOLUTION

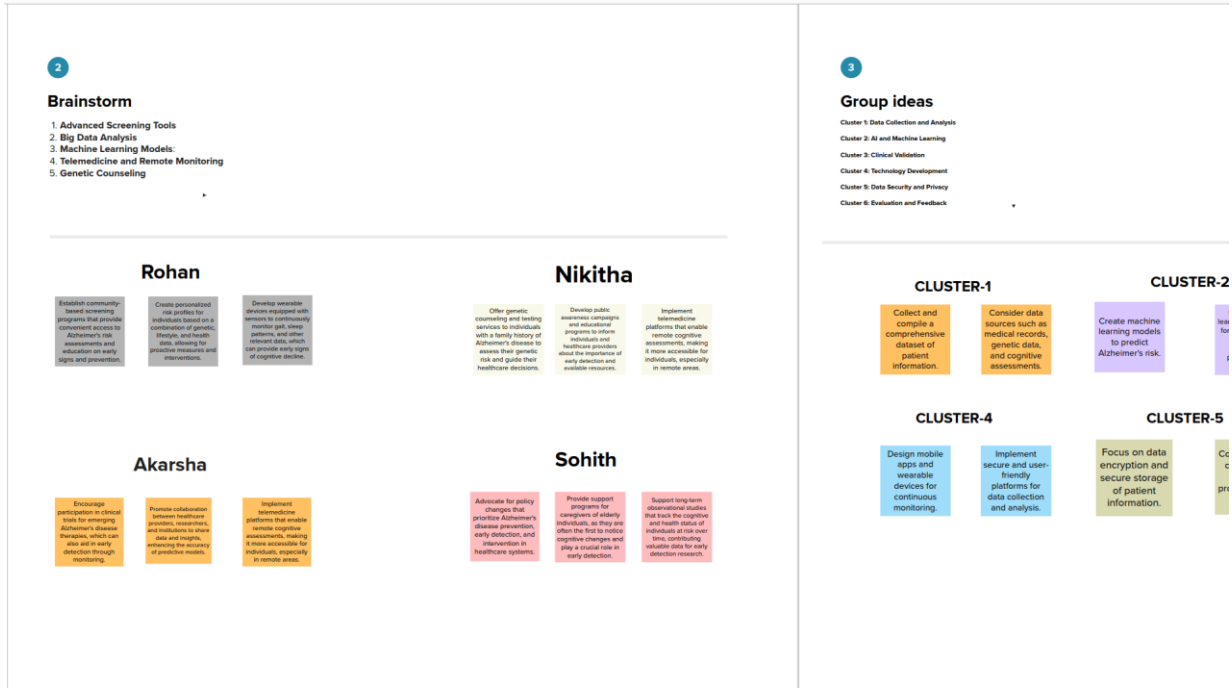
3.1 Empathy Map Canvas

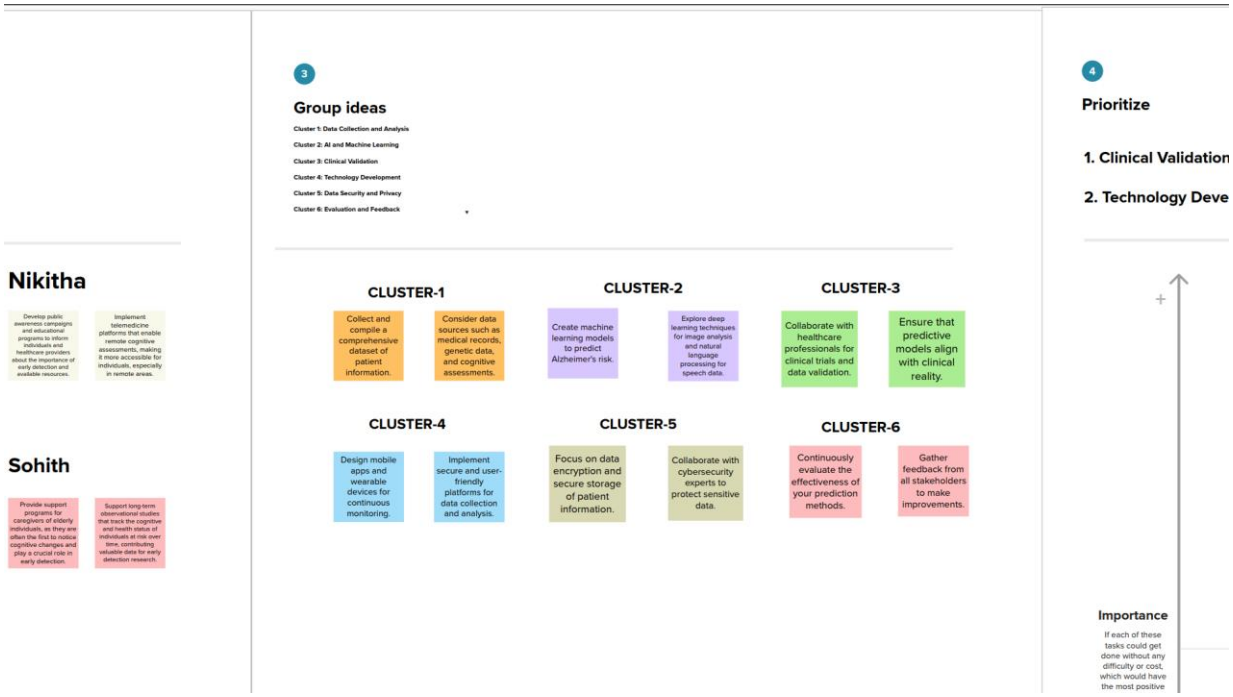


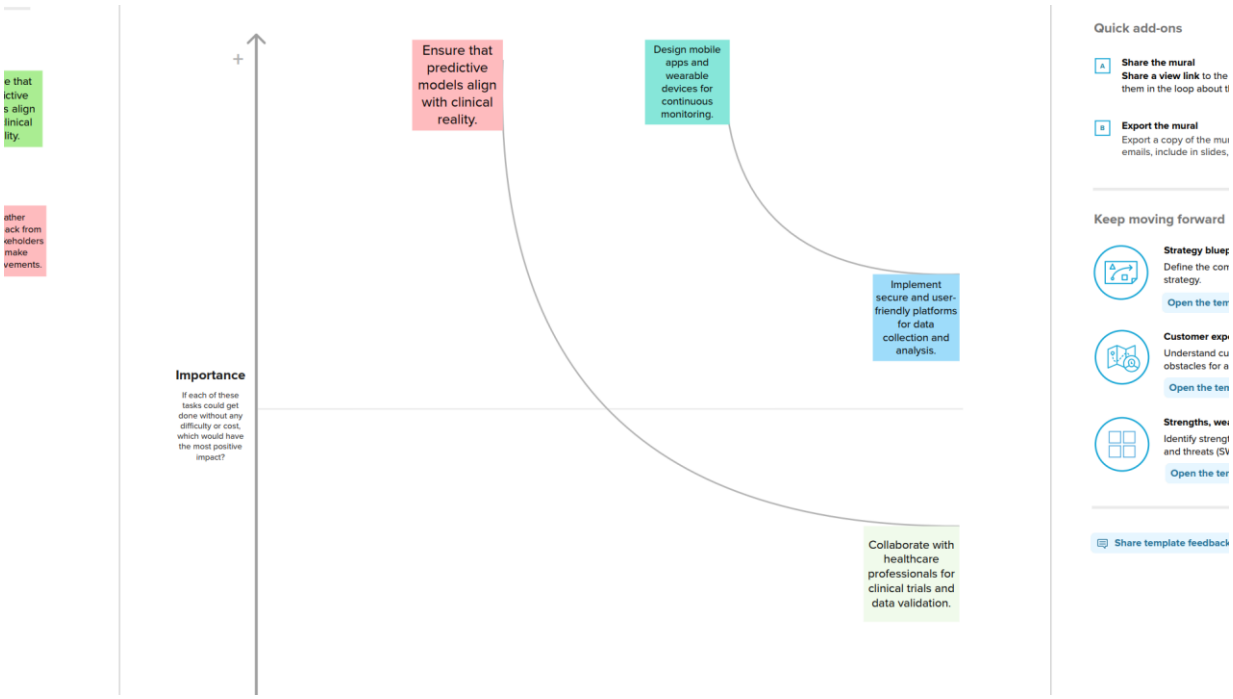


3.2 Ideation & Brainstorming






Nikitha
Develop public awareness campaigns and educational programs to inform individuals and healthcare providers about the importance of early detection and available resources.
Implement telemedicine platforms that enable remote cognitive assessments, making it more accessible for individuals, especially in remote areas.
Sohith
Provide support programs for caregivers of elderly individuals, as they are often the first to notice cognitive changes and play a crucial role in early detection.
Support long-term observational studies that track the cognitive and health status of individuals at risk over time, contributing valuable data for early detection research.
3
Group ideas
Cluster 1: Data Collection and Analysis
Cluster 2: AI and Machine Learning
Cluster 3: Clinical Validation
Cluster 4: Technology Development
Cluster 5: Data Security and Privacy
Cluster 6: Evaluation and Feedback
CLUSTER-1
Collect and compile a comprehensive dataset of patient information.
Consider data sources such as medical records, genetic data, and cognitive assessments.
CLUSTER-2
Create machine learning models to predict Alzheimer's risk.
Explore deep learning techniques for image analysis and natural language processing for speech data.
CLUSTER-3
Collaborate with healthcare professionals for clinical trials and data validation.
Ensure that predictive models align with clinical reality.
CLUSTER-4
Design mobile apps and wearable devices for continuous monitoring.
Implement secure and user-friendly platforms for data collection and analysis.
CLUSTER-5
Focus on data encryption and secure storage of patient information.
Collaborate with cybersecurity experts to protect sensitive data.
CLUSTER-6
Continuously evaluate the effectiveness of your prediction methods.
Gather feedback from all stakeholders to make improvements.
4
Prioritize
1. Clinical Validation
2. Technology Deve
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive

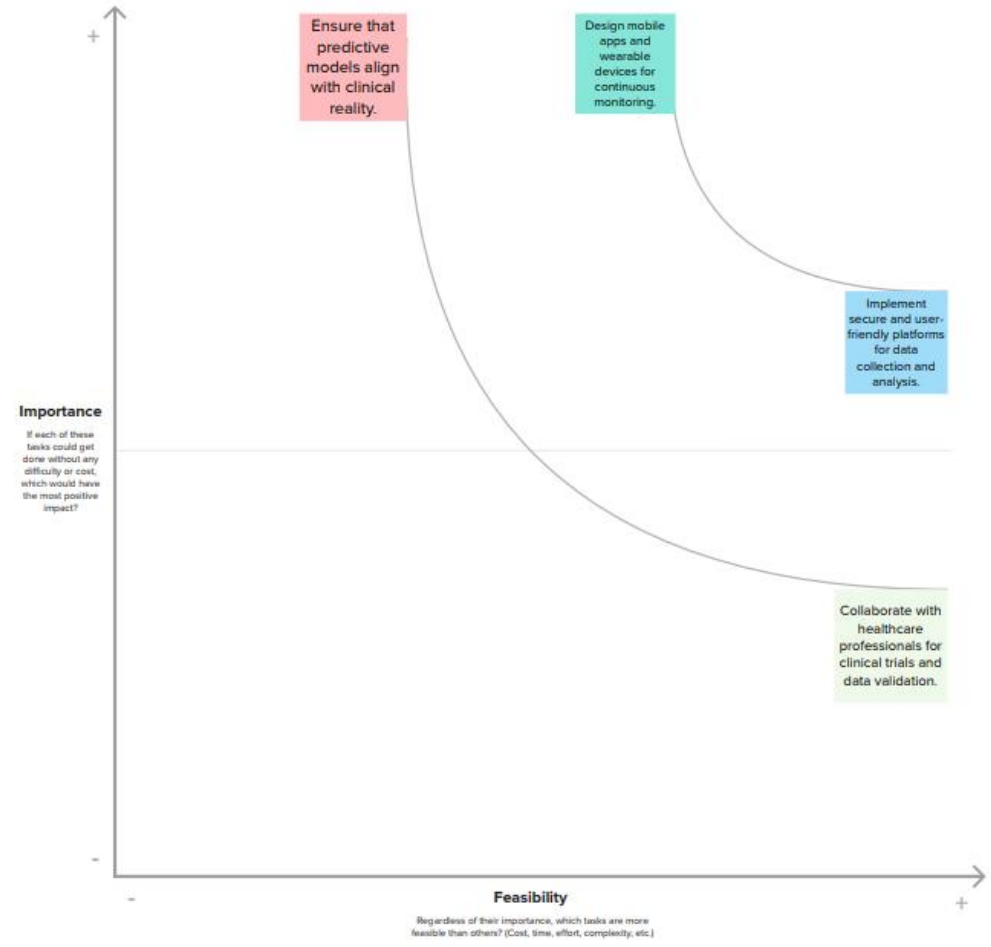

Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Ensure that predictive models align with clinical reality.
Design mobile apps and wearable devices for continuous monitoring.
Implement secure and user-friendly platforms for data collection and analysis.
Collaborate with healthcare professionals for clinical trials and data validation.
Quick add-ons
Share the mural
Share a view link to the
them in the loop about t
Export the mural
Export a copy of the mu
emails, include in slides,
Keep moving forward
Strategy blue
Define the cor
strategy.
Open the ter
Customer exp
Understand cu
obstacles for a
Open the ten
Strengths, we
Identify strengt
and threats (SV
Open the ter
Share template feedback

4

Prioritize

1. Clinical Validation

2. Technology Development



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

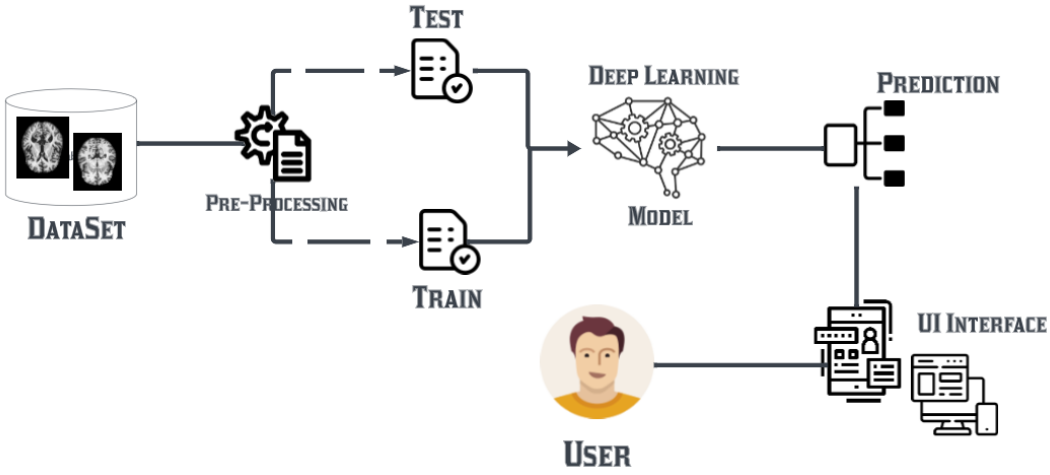
- Data Collection and Integration: The system should gather and integrate diverse data types (genetic, neurological, behavioral) relevant to Alzheimer's prediction.
- Prediction Model Development: Develop algorithms/models to predict the onset or progression of Alzheimer's disease based on collected data.
- User Interface for Data Input: Provide a user-friendly interface for healthcare professionals to input and update patient data securely.
- Real-time Monitoring: Enable real-time or scheduled monitoring of patient data to continuously update predictions.
- Prediction Reporting: Generate reports or alerts indicating the likelihood or risk of Alzheimer's onset or progression for individual patients.

4.2 Non-Functional requirements

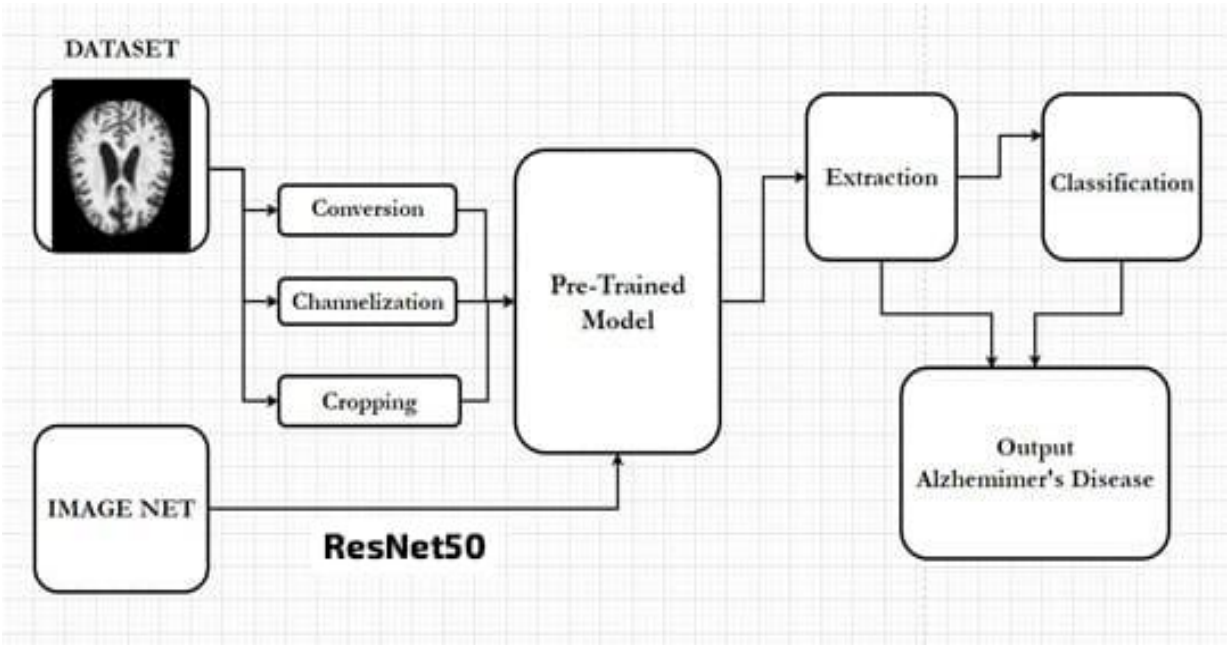
- Security: Ensure robust security measures for patient data, complying with healthcare data privacy regulations (e.g., HIPAA).
- Performance: The system should handle a large volume of data efficiently and provide timely predictions without significant delays.
- Reliability: Maintain high accuracy in predictions while minimizing false positives/negatives to instill confidence in healthcare professionals using the system.
- Scalability: Design the system to accommodate future expansions, allowing for the addition of new data sources or improvements in prediction models.
- Usability: Ensure ease of use for healthcare professionals interacting with the system, providing clear interfaces and intuitive functionalities.
- Interoperability: Enable compatibility with existing healthcare systems or databases to facilitate data exchange and integration.

5 PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

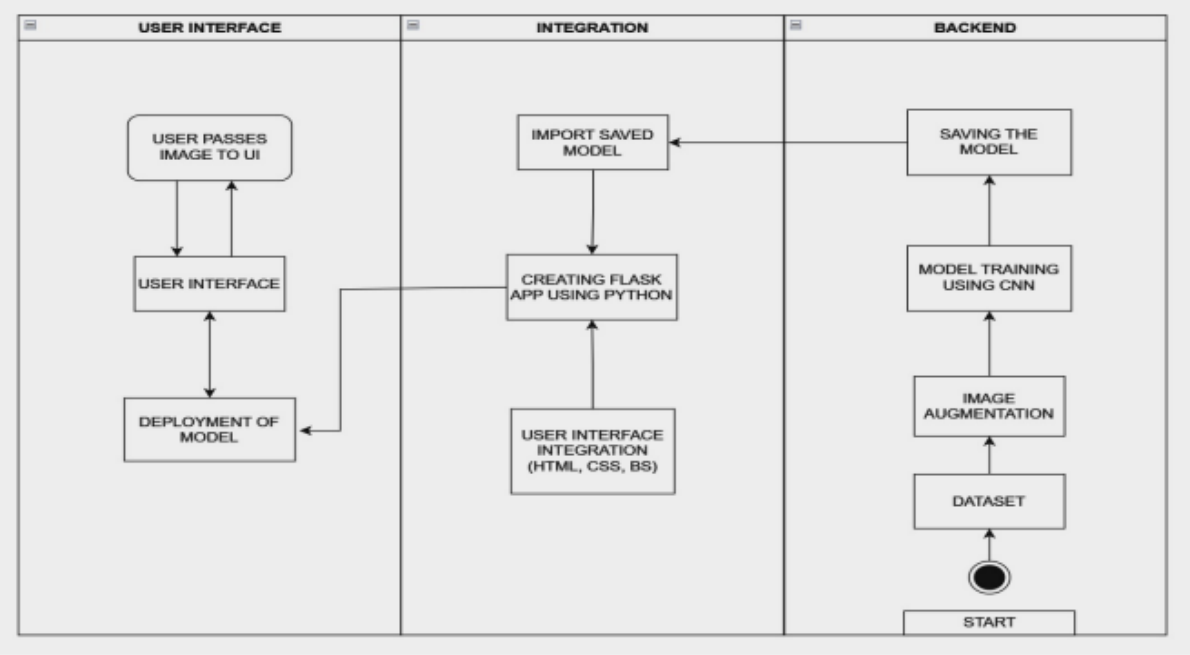


5.2 Solution Architecture



6 PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	3	1 Days	30 Oct 2023	30 Oct 2023	20	3 Nov 2023
Sprint-2	5	4 Days	31 Oct 2023	3 Nov 2023		
Sprint-3	10	5 Days	4 Nov 2023	8 Nov 2023		
Sprint-4	1	10 Days	9 Nov 2023	18 Nov 2023		
Sprint-5	1	4 Days	19 Nov 2023	22 Nov 2023		

6.3 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Project setup & Infrastructure	USN-1	Set up the development environment with the required tools and frameworks to start the garbage classification project.	1	High	Nikitha
Sprint-1	Development environment	USN-2	Gather a diverse dataset of images containing different types of garbage (plastic, paper, glass, organic) for training the deep learning model.	2	High	Nikitha
Sprint-2	Data collection	USN-3	Preprocess the collected dataset by resizing images, normalizing pixel values, and splitting it into training and validation sets	2	High	Rohan
Sprint-2	data preprocessing	USN-4	Explore and evaluate different deep learning architectures (e.g., CNNs) to select the most suitable model for garbage classification.	3	High	Akarsha
Sprint-3	model development	USN-5	train the selected deep learning model using the preprocessed dataset and monitor its performance on the validation set.	4	High	Sohith
Sprint-3	Training	USN-6	implement data augmentation techniques (e.g., rotation, flipping) to improve the model's robustness and accuracy.	6	Medium	Rohan
Sprint-4	model deployment & Integration	USN-7	deploy the trained deep learning model as an API or web service to make it accessible for garbage classification. integrate the model's API into a user-friendly web interface for users to upload images and receive garbage classification results.	1	Medium	Sohith
Sprint-5	Testing & quality assurance	USN-8	conduct thorough testing of the model and web interface to identify and report any issues or bugs. fine-tune the model hyperparameters and optimize its performance based on user feedback and testing results.	1	Medium	Akarsha

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

Dataset: https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images

Alzheimer's disease categories include Non-Dementia, Very-Mild Dementia, Mild Dementia, and Moderate Dementia. Symptoms of AD are contingent upon the disease's stage, with short-term memory loss typically being the initial and prevalent symptom. Language disorders are also frequently observed in AD patients. The concealment of AD symptoms in the early stages presents a significant obstacle to appropriate treatment.

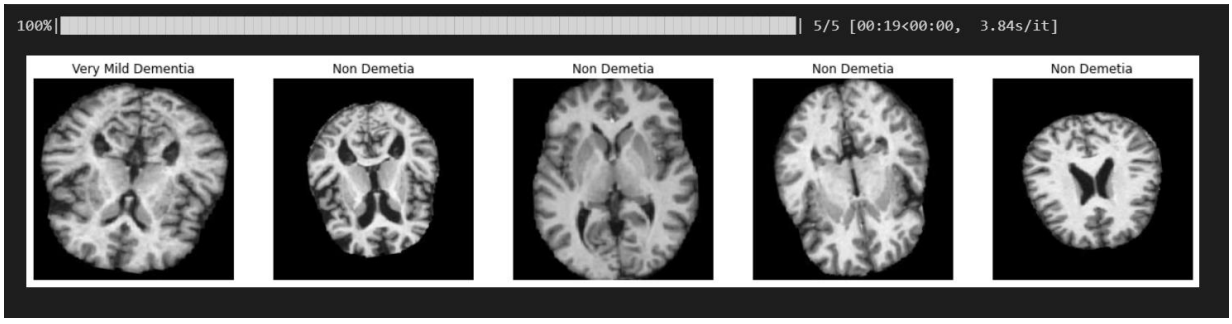
- **Non-Dementia:** Typically, individuals exhibit normal cognitive function or mild impairment that doesn't meet dementia criteria. Deep learning aims to differentiate this stage, identifying those with subtle cognitive changes.
- **Very-Mild Dementia:** Characterized by subtle cognitive alterations that minimally effect daily life. Deep learning models target early markers in data to signal the onset of

cognitive decline at this stage.

- **Mild Dementia:** Notable cognitive decline impacts memory, language, and daily activities. Deep learning focuses on identifying specific patterns in patient data, especially neuroimaging, correlating with these changes.
- **Moderate Dementia:** Advanced cognitive decline significantly impairs daily functioning and independence. Deep learning strives to pinpoint severe patterns or biomarkers associated with this stage, aiding in precise classification and potential interventions.

A multi-classification approach is used to classify samples into these four stages. Additionally, separate binary classifications are performed between each pair of classes to further distinguish and classify samples within the AD spectrum.

```
# Images views from different classes
fig, ax = plt.subplots(nrows = 1, ncols = 5, figsize=(20,20))

for i in tqdm(range(0,5)):
    rand1 = np.random.randint(len(train_dataset))
    rand2 = np.random.randint(100)
    ax[i].imshow(train_dataset[rand1][0][rand2])
    ax[i].axis('off')
    a = train_dataset[rand1][1][rand2]
    if a[0] == 1:
        ax[i].set_title('Mild Dementia')
    elif a[1] == 1:
        ax[i].set_title('Moderate Dementia')
    elif a[2] == 1:
        ax[i].set_title('Non Demetia')
    elif a[3] == 1:
        ax[i].set_title('Very Mild Dementia')
```



8. PERFORMANCE TESTING

a. Performance Metrics

```
model.evaluate(test_dataset)
```

```
10/10 [==============================] - 91s 9s/step - loss: 1.1231 - auc: 0.7804

[1.1231095790863037, 0.780361533164978]
```

```
# Training the model using number of epochs
model_history=model.fit(train_dataset,
                        validation_data=valid_dataset,
                        epochs = 1, # Number of epochs can be changed
                        callbacks = callback_list,
                        verbose = 1)
```

```
33/33 [==============================] - 516s 16s/step - loss: 1.2512 - auc: 0.7773 - val_loss: 1.0808 - val_auc: 0.8090

Epoch 00001: val_auc improved from 0.77528 to 0.80895, saving model to .\best_weights.hdf5
```

```
Epoch 1/30
33/33 [==============================] - ETA: 0s - loss: 1.5367 - auc: 0.7390
Epoch 1: val_auc improved from -inf to 0.58925, saving model to .\best_weights.hdf5
33/33 [==============================] - 336s 10s/step - loss: 1.5367 - auc: 0.7390 - val_loss: 2.9766 - val_auc: 0.5893
Epoch 2/30
33/33 [==============================] - ETA: 0s - loss: 1.2218 - auc: 0.7859
Epoch 2: val_auc improved from 0.58925 to 0.69831, saving model to .\best_weights.hdf5
33/33 [==============================] - 317s 10s/step - loss: 1.2218 - auc: 0.7859 - val_loss: 1.3996 - val_auc: 0.6983
Epoch 3/30
33/33 [==============================] - ETA: 0s - loss: 1.1527 - auc: 0.7999
Epoch 3: val_auc did not improve from 0.69831
33/33 [==============================] - 305s 9s/step - loss: 1.1527 - auc: 0.7999 - val_loss: 1.2900 - val_auc: 0.6431
Epoch 4/30
33/33 [==============================] - ETA: 0s - loss: 1.1105 - auc: 0.7986
Epoch 4: val_auc did not improve from 0.69831
33/33 [==============================] - 304s 9s/step - loss: 1.1105 - auc: 0.7986 - val_loss: 1.3262 - val_auc: 0.6571
Epoch 5/30
33/33 [==============================] - ETA: 0s - loss: 1.0633 - auc: 0.8100
Epoch 5: val_auc improved from 0.69831 to 0.75601, saving model to .\best_weights.hdf5
33/33 [==============================] - 303s 9s/step - loss: 1.0633 - auc: 0.8100 - val_loss: 1.1851 - val_auc: 0.7560
Epoch 6/30
33/33 [==============================] - ETA: 0s - loss: 1.0483 - auc: 0.8123
Epoch 6: val_auc did not improve from 0.75601
33/33 [==============================] - 296s 9s/step - loss: 1.0483 - auc: 0.8123 - val_loss: 1.1452 - val_auc: 0.7508
Epoch 7/30
33/33 [==============================] - ETA: 0s - loss: 1.0214 - auc: 0.8177
Epoch 7: val_auc did not improve from 0.75601
33/33 [==============================] - 294s 9s/step - loss: 1.0214 - auc: 0.8177 - val_loss: 1.1412 - val_auc: 0.7516
```

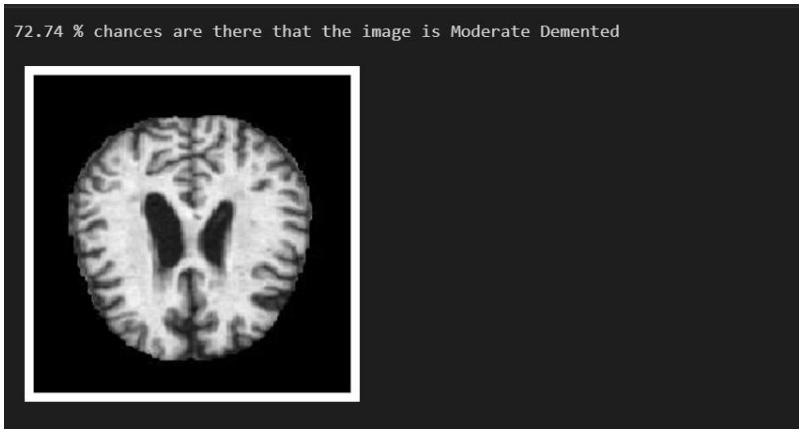
9. RESULTS

a. Output Screenshots

```
# Test Case No.1: Non-Dementia

dic = test_dataset.class_indices
idc = {k:v for v, k in dic.items()}

img = load_img(r'C:\Users\Jeath\Downloads\Alzheimer-Disease-Diagnosis-main\Alzheimer-Disease-Diagnosis-main\Alz
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)
#answer = model.predict_classes(img)
answer = (model.predict(img) > 0.5).astype("int32")
#probability = round(np.max(model.predict_proba(img)*100),2)

probability = round(np.max(model.predict(img)*100),2)

#predict_classes=np.argmax(predict_prob,axis=1)

print(probability, '% chances are there that the image is Non-Dementia')
```

```
55.45 % chances are there that the image is Non-Dementia
```

```
# Test Case 3: Moderate Demented

dic = test_dataset.class_indices
idc = {k:v for v, k in dic.items()}

img = load_img(r'C:\Users\Jeath\Downloads\Alzheimer-Disease-Diagnosis-main\Alzheimer-Disease-Diagnosis-main\Alzheimer
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)
#answer = model.predict_classes(img)
answer = (model.predict(img) > 0.5).astype("int32")
#probability = round(np.max(model.predict_proba(img)*100),2)

probability = round(np.max(model.predict(img)*100),2)

#predict_classes=np.argmax(predict_prob,axis=1)

print(probability, '% chances are there that the image is Moderate Demented')
```



10. ADVANTAGES & DISADVANTAGES

Advantages:

- **Early Detection Potential:** Deep learning models excel at identifying subtle patterns within complex data, enabling the detection of early-stage biomarkers or indicators of Alzheimer's disease before noticeable symptoms appear.
- **Transfer Learning Capabilities:** Pre-trained ResNet50 models on large datasets (e.g., ImageNet) can be fine-tuned with Alzheimer's-specific data. This transfer learning helps leverage the knowledge gained from broader datasets, enhancing performance in disease prediction tasks.
- **Accurate Predictive Capabilities:** Leveraging convolutional neural networks (CNNs) and other deep learning architectures, these models can process diverse datasets, including neuroimaging scans and genetic information, leading to more precise and accurate predictions compared to traditional methods.
- **Robustness to Overfitting:** The skip connections in ResNet50 aid in mitigating overfitting issues, allowing the model to generalize well even with a limited dataset, which is crucial in medical applications where extensive datasets might be unavailable.
- **Potential for Automation:** These models hold the potential to automate certain aspects of diagnosis and prediction, easing the burden on healthcare professionals and improving efficiency in healthcare delivery.

Disadvantages:

- **Complexity and Interpretability:** Deep learning models, especially deep neural networks, are often viewed as "black boxes," making it challenging to interpret the reasoning behind their predictions. This lack of interpretability can hinder trust and understanding, crucial in medical decision-making.
- **Overfitting and Generalization**: deep learning models can overfit to the training data, resulting in poor generalization to new, unseen data. This is a concern in healthcare where models need to perform reliably on diverse patient populations.
- **Deployment Challenges:** Translating ResNet50 models into practical clinical use might be challenging due to their computational demands and the need for specialized expertise in deploying and maintaining such systems within healthcare environments.

11. CONCLUSION

It has been established that Alzheimer's disease as an incurable neurodegenerative condition that notably impacts memory functions, particularly among the elderly. The sheer volume of patients makes manual diagnosis impractical, leading to potential errors by healthcare specialists due to time constraints and the complexity of assessment. While various diagnostic procedures exist, there's an urgent need for a precise and prompt diagnostic solution. The proposed model advocates a deep learning approach utilizing ResNet-50 and CNN architectures for diagnosing and categorizing Alzheimer's disease.

Within this model, Alzheimer's disease was classified into four categories: Non-Dementia, Very Mild-Dementia, Mild Dementia, and Moderate Dementia. During both training and testing phases, the ResNet50 method showcased superior performance. This proposed method holds promise for real-time analysis and classification of Alzheimer's disease. Future plans involve expanding disease detection using larger datasets and employing diverse measures to enhance the system's accuracy.

12. FUTURE SCOPE

Deep learning is the field of constable's development. Accuracy can be improved because of the advancements in deep learning and machine learning. Very wide multi modal approaches provide a more comprehensive view of the disease progression. We can also evolve in the monitoring of the person vitals and also helps in the personalized medicine checking their can also evolve in the monitoring of the person vitals and also helps in the personalized medicine checking their stages. Large scale collaborative datasets can be among institutions for further detailed research.

- Combining sparse regression and deep learning (DL) methods for diagnosing Alzheimer's disease (AD) can be effective. Additionally, one promising technique for AD diagnosis is the manifold-based learning method.
- To improve the overall performance of AD diagnosis, data augmentation and scaling techniques are valuable. These methods can enhance the state-of-the-art performance by increasing the diversity and volume of data, making the models more robust and accurate.
- One of the challenges in Alzheimer's disease research is the difficulty in collecting brain-balanced and sufficient data. Obtaining a well-balanced dataset that adequately represents the different aspects and stages of the disease can be challenging. Additionally, gathering a sufficient amount of data that is large enough to train accurate and robust machine learning models remains a significant hurdle in the field. These challenges can impact the development and performance of AI models for Alzheimer's disease diagnosis and research.

13. APPENDIX

Source Code

Mount Goggle Drive:

```
from google.colab import drive
drive.mount('/content/drive')
```

Import the Required libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import skimage.io
import os
import tqdm
import glob
import tensorflow

from tqdm import tqdm
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split

from skimage.io import imread, imshow
from skimage.transform import resize
from skimage.color import grey2rgb

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import InputLayer, BatchNormalization, Dropout,
Flatten, Dense, Activation, MaxPool2D, Conv2D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.applications.densenet import DenseNet169
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

Import View Pre-Processing Dataset:

```
# Data Augmentation
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   rotation_range=30,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   vertical_flip=True,
                                   validation_split = 0.2)

valid_datagen = ImageDataGenerator(rescale = 1./255,
                                   validation_split = 0.2)

test_datagen  = ImageDataGenerator(rescale = 1./255)
```

```
Found 4098 images belonging to 4 classes.
```

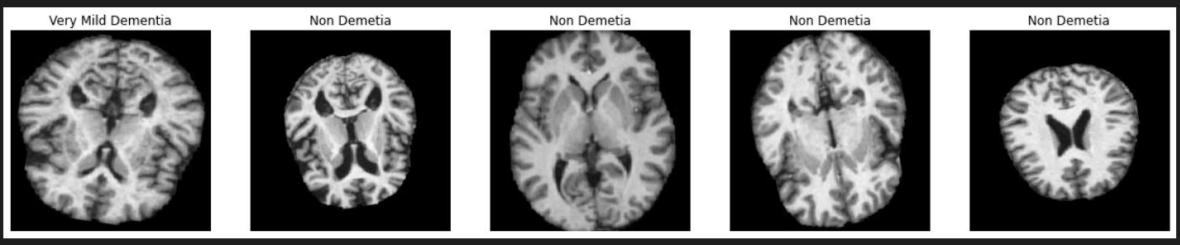
```
# Data Validation
valid_dataset = valid_datagen.flow_from_directory(directory =
r'C:\Users\Jeath\Downloads\Alzheimer-Disease-Diagnosis-main\Alzheimer-Disease-
Diagnosis-main\Alzheimer_Dataset\train',
                                                  target_size = (224,224),
```

```
                                                class_mode = 'categorical',
                                                subset = 'validation',
                                                batch_size = 128)
```

```
Found 1023 images belonging to 4 classes.
```

```
# Images views from different classes
fig, ax = plt.subplots(nrows = 1, ncols = 5, figsize=(20,20))

for i in tqdm(range(0,5)):
    rand1 = np.random.randint(len(train_dataset))
    rand2 = np.random.randint(100)
    ax[i].imshow(train_dataset[rand1][0][rand2])
    ax[i].axis('off')
    a = train_dataset[rand1][1][rand2]
    if a[0] == 1:
        ax[i].set_title('Mild Dementia')
    elif a[1] == 1:
        ax[i].set_title('Moderate Dementia')
    elif a[2] == 1:
        ax[i].set_title('Non Dementia')
    elif a[3] == 1:
        ax[i].set_title('Very Mild Dementia')
```

```
100%| 5/5 [00:19<00:00,  3.84s/it]
```



Build the model:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
import skimage.io
import os
import tqdm
import glob
import tensorflow

from tqdm import tqdm
from sklearn.utils import shuffle
from sklearn import metrics
```

```
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split

from skimage.io import imread, imshow
from skimage.transform import resize
from skimage.color import grey2rgb

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import InputLayer, BatchNormalization, Dropout,
Flatten, Dense, Activation, MaxPool2D, Conv2D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.utils import to_categorical
from keras import optimizers

from keras.callbacks import Callback,ModelCheckpoint
from keras.models import Sequential,load_model
from keras.layers import Dense, Dropout
from keras.wrappers.scikit_learn import KerasClassifier
import keras.backend as K

#import tensorflow_addons as tfa
#from tensorflow.keras.metrics import Metric
#from tensorflow_addons.utils.types import AcceptableDTypes, FloatTensorLike
from typeguard import typechecked
from typing import Optional
```

```
# Model Initialization

base_model = ResNet50(input_shape=(224,224,3),
                      include_top=False,
                      weights="imagenet")
```

```
# Freezing Layers

for layer in base_model.layers:
    layer.trainable=False
```

```
# Building Model and adjust the required parameters

model=Sequential()
model.add(base_model)
model.add(Dropout(0.5))
model.add(Flatten())
```

```
model.add(BatchNormalization())
model.add(Dense(2048,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1024,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(4,activation='softmax'))
```

```
# Model Summary

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 7, 7, 2048)        23587712
_________________________________________________________________
dropout (Dropout)            (None, 7, 7, 2048)        0
_________________________________________________________________
flatten (Flatten)            (None, 100352)            0
_________________________________________________________________
batch_normalization (BatchNo (None, 100352)            401408
_________________________________________________________________
dense (Dense)                (None, 2048)              205522944
_________________________________________________________________
batch_normalization_1 (Batch (None, 2048)              8192
_________________________________________________________________
activation (Activation)      (None, 2048)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2098176
_________________________________________________________________
batch_normalization_2 (Batch (None, 1024)              4096
_________________________________________________________________
activation_1 (Activation)    (None, 1024)              0
...
Total params: 231,626,628
Trainable params: 207,832,068
Non-trainable params: 23,794,560
_________________________________________________________________
```

```
# Model Compile

OPT    = tensorflow.keras.optimizers.Adam(lr=0.001)

model.compile(loss='categorical_crossentropy',
              metrics=[tensorflow.keras.metrics.AUC(name = 'auc')],
              optimizer=OPT)
```

```
# Defining Callbacks

filepath = './best_weights.hdf5'

earlystopping = EarlyStopping(monitor = 'val_auc',
                              mode = 'max' ,
                              patience = 15,
                              verbose = 1)

checkpoint    = ModelCheckpoint(filepath,
                                monitor = 'val_auc',
                                mode='max',
                                save_best_only=True,
                                verbose = 1)


callback_list = [earlystopping, checkpoint]
```

```
# Training the model using number of epochs
model_history=model.fit(train_dataset,
                        validation_data=valid_dataset,
                        epochs = 1, # Number of epochs can be changed
                        callbacks = callback_list,
                        verbose = 1)
```

```
33/33 [==============================] - 516s 16s/step - loss: 1.2512 - auc: 0.7773 - val_loss: 1.0808 - val_auc: 0.8090

Epoch 00001: val_auc improved from 0.77528 to 0.80895, saving model to .\best_weights.hdf5
```

Model Evaluation:

```
# Summarize history for loss function

plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left', bbox_to_anchor=(1,1))
plt.show()
```

```
# Summarize history for loss and value accuracy

plt.plot(model_history.history['auc'])
plt.plot(model_history.history['val_auc'])
plt.title('Model AUC')
plt.ylabel('AUC')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left', bbox_to_anchor=(1,1))
plt.show()
```

```
# Test Model using the Test Dataset

test_dataset  = test_datagen.flow_from_directory(directory =
r'C:\Users\Jeath\Downloads\Alzheimer-Disease-Diagnosis-main\Alzheimer-Disease-
Diagnosis-main\Alzheimer_Dataset\test',
                                                 target_size = (224,224),
                                                 class_mode = 'categorical',
                                                 batch_size = 128)
```

```
Found 1279 images belonging to 4 classes.
```

```
# Evaluating Loss and AUC

model.evaluate(test_dataset)
```

```
10/10 [==============================] - 91s 9s/step - loss: 1.1231 - auc: 0.7804

[1.1231095790863037, 0.780361533164978]
```

```
# Test Case No.1: Non-Dementia

dic = test_dataset.class_indices
idc = {k:v for v, k in dic.items()}

img = load_img(r'C:\Users\Jeath\Downloads\Alzheimer-Disease-Diagnosis-
main\Alzheimer-Disease-Diagnosis-main\Alzheimer_Dataset\test\NonDemented\26
(100).jpg', target_size = (224,224,3))
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)
#answer = model.predict_classes(img)
answer = (model.predict(img) > 0.5).astype("int32")
#probability = round(np.max(model.predict_proba(img)*100),2)

probability = round(np.max(model.predict(img)*100),2)
```

```
#predict_classes=np.argmax(predict_prob,axis=1)

print(probability, '% chances are there that the image is Non-Dementia')
```

```
55.45 % chances are there that the image is Non-Dementia
```

```
# Test Case 2: Mild Demented

dic = test_dataset.class_indices
idc = {k:v for v, k in dic.items()}

img = load_img(r'C:\Users\Jeath\Downloads\Alzheimer-Disease-Diagnosis-
main\Alzheimer-Disease-Diagnosis-main\Alzheimer_Dataset\test/MildDemented/26
(20).jpg', target_size = (224,224,3))
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)
#answer = model.predict_classes(img)
answer = (model.predict(img) > 0.5).astype("int32")
#probability = round(np.max(model.predict_proba(img)*100),2)

probability = round(np.max(model.predict(img)*100),2)

#predict_classes=np.argmax(predict_prob,axis=1)

print(probability, '% chances are there that the image is Mild Demented')
```

```
49.92 % chances are there that the image is Mild Demented
```

```
# Test Case 3: Moderate Demented

dic = test_dataset.class_indices
idc = {k:v for v, k in dic.items()}

img = load_img(r'C:\Users\Jeath\Downloads\Alzheimer-Disease-Diagnosis-
main\Alzheimer-Disease-Diagnosis-
main\Alzheimer_Dataset\test\ModerateDemented\29.jpg', target_size =
(224,224,3))
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)
#answer = model.predict_classes(img)
answer = (model.predict(img) > 0.5).astype("int32")
#probability = round(np.max(model.predict_proba(img)*100),2)

probability = round(np.max(model.predict(img)*100),2)

#predict_classes=np.argmax(predict_prob,axis=1)

print(probability, '% chances are there that the image is Moderate Demented')
```

```
72.74 % chances are there that the image is Moderate Demented
```

```
# Test Case 4: Very Mild Demented

dic = test_dataset.class_indices
idc = {k:v for v, k in dic.items()}

img = load_img(r'C:\Users\Jeath\Downloads\Alzheimer-Disease-Diagnosis-
main\Alzheimer-Disease-Diagnosis-main\Alzheimer_Dataset\test/MildDemented/26
(22).jpg', target_size = (224,224,3))
img = img_to_array(img)
img = img/255
imshow(img)
plt.axis('off')
img = np.expand_dims(img,axis=0)
#answer = model.predict_classes(img)
answer = (model.predict(img) > 0.5).astype("int32")
#probability = round(np.max(model.predict_proba(img)*100),2)

probability = round(np.max(model.predict(img)*100),2)

#predict_classes=np.argmax(predict_prob,axis=1)

print(probability, '% chances are there that the image is Moderate Demented')
```

GitHub Link:

https://github.com/smartinternz02/SI-GuidedProject-610458-1698824657.git