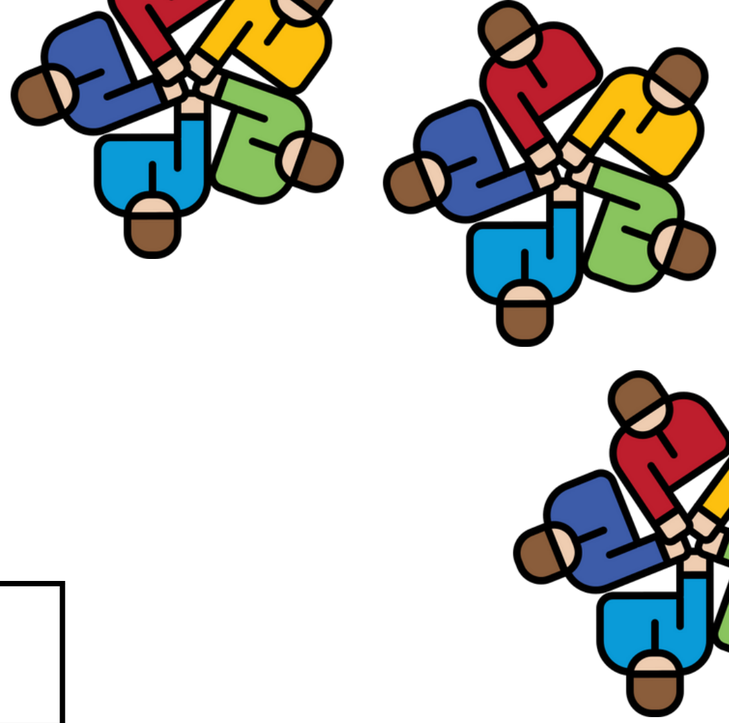
# Enhancing the Efficiency of Diagnosing and Managing Mental Disorders using Machine Learning

SLIIT FACULTY OF COMPUTING

Group ID: 24-25J-322

# Team Members



## Supervisor: Ms. Wishalya Tissera

## Co-Supervisor: Dr. Kapila Dissanayaka

| Student ID | Student Name |
|---|---|
| IT21180934 | Dilshan G.A.M |
| IT21164026 | Wijayasooriya W.A.K.A |
| IT21182846 | Chamaleen D.B.N |
| IT21178368 | Perera K.K.S |

SLIIT FACULTY OF COMPUTING

**Group ID: 24-25J-322**

# INTRODUCTION

According to the World Health Organization **1 in every 8 people** in the world live with a mental disorder.

Mental disorders, significantly impact an individual's cognitive, emotional, and social functioning.

Early diagnosis is crucial for effective intervention, as it can greatly improve long-term outcomes.

# INTRODUCTION

- According to world's Alzheimer's association 1 out of 9 adults (>65) have Alzheimer's.
- Over 700,000 lives have been lost all over the world due to depression. (Global suicide statistics IASP)
- Approximately 1 out of 13 children are being diagnosed with autism. Data and Statistics on Autism Spectrum Disorder | Autism Spectrum Disorder (ASD) | CDC
- Parkinson foundation estimates that by the year 2030 the patients with Parkinson's disease will be rise to 1.2 million (https://www.parkinson.org/understanding-parkinsons/statistics)
- These diseases needed to be identified at the first stages to ensure immediate treatment. That helps patients' family members to plan their lives and keep the situation under control.

# Research Problem

- How to leverage machine and deep learning techniques to enhance mental care?
- Up to what extent machine and deep learning can be used to diagnose and analyze mental disorders like Autism, Alzheimer's, Parkinson's, and Depression?
- How trustworthy and useful are the built models?
- How to monitor patients effectively?

# Research Objectives

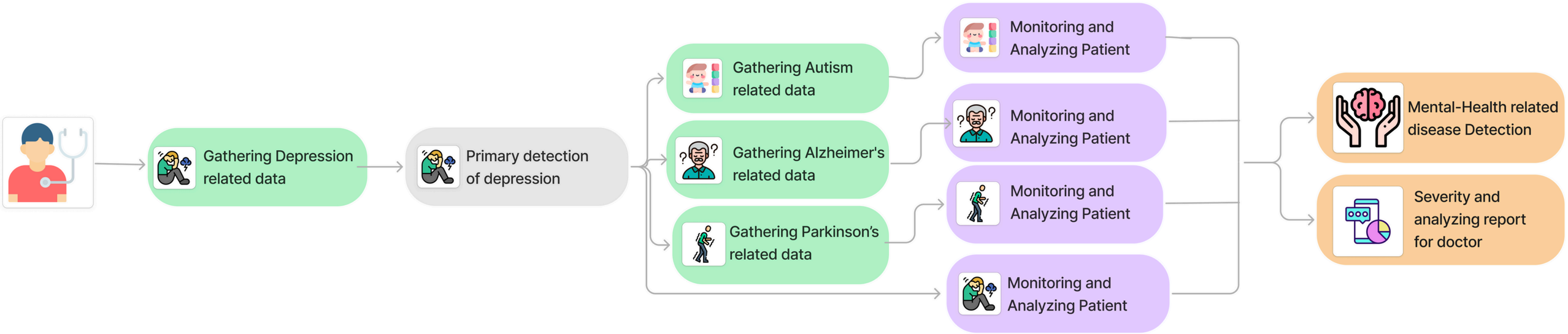
Diagnosing and Analyzing Alzheimer’s Risk through Machine Learning Techniques

Early Detection of Parkinson's: Harnessing Machine Learning for Predictive Risk Modeling.

Early Autism Spectrum Disorder Detection Using Facial Images and Text Analysis

Diagnosing and Analyzing Depression through Text and Facial Images

# Overall System Diagram



# Diagnosing and Analyzing Alzheimer’s Risk through Machine Learning Techniques


## IT21180934 | Dilshan G.A.M

Bachelor of Science (Hons) in Information Technology
Specializing in Data Science

# Background

## Component 1

## What is Alzeihmers?

- Alzheimer's disease is a progressive neurological disorder that leads to memory loss, cognitive decline, and behavioral changes.

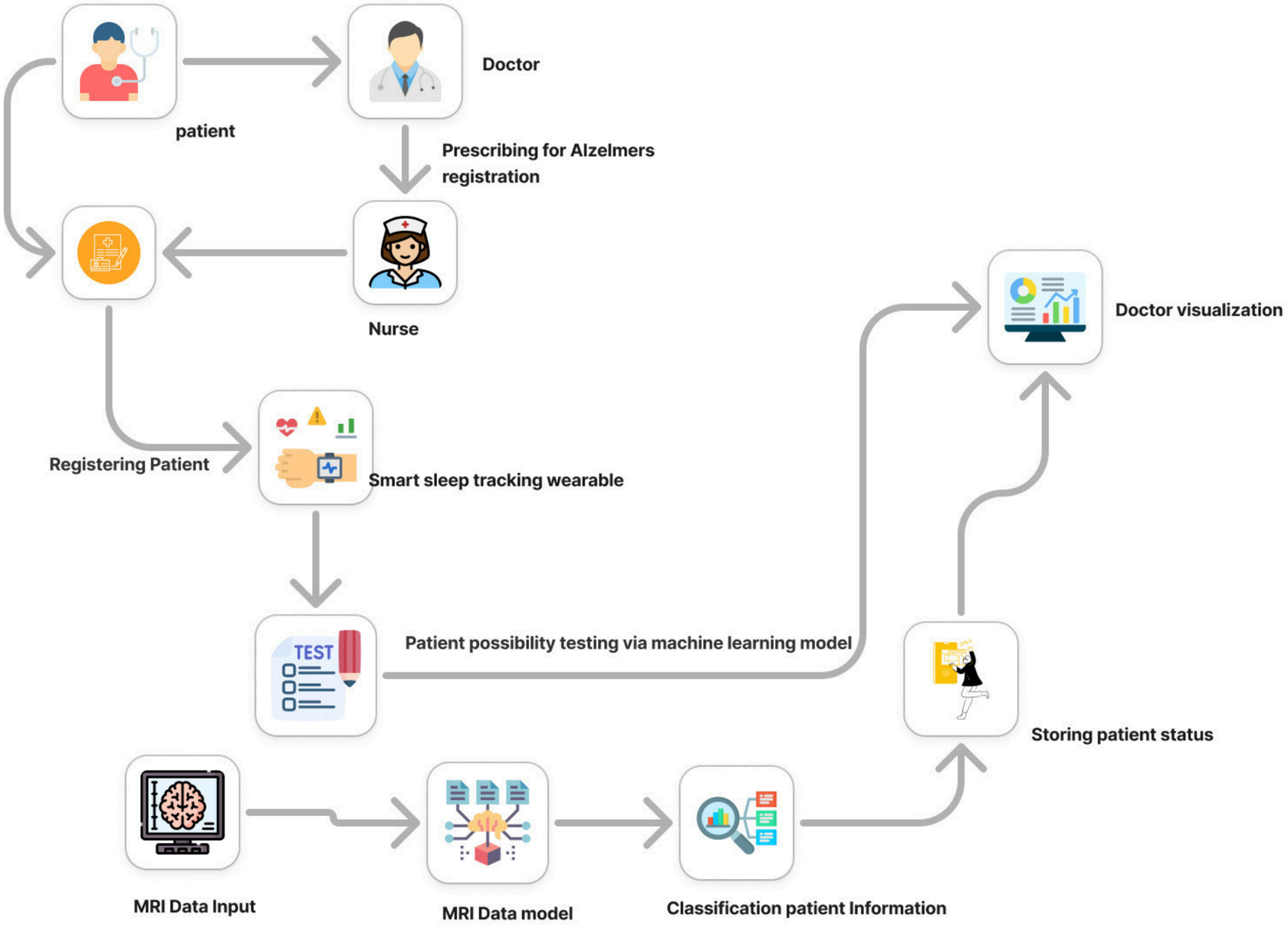
# Research Question

- How to get patient sleep quality measures?
- How to develop a solution to integrate all these components in one application?
- How to feed MRI scan images into the system?
- How to manage the patient status with the medications and treatments through out the time?
- How to make clear visualizations for doctors to evaluate patients' medical history with the treatments?

# Specific & Sub Objectives

- Classifying patients’ severity using MRI scans.
- Identifying possibility of a patient having Alzheimer's using general information.
- Visualizing patient possibilities and levels of the disease through out the time.

# Methodology

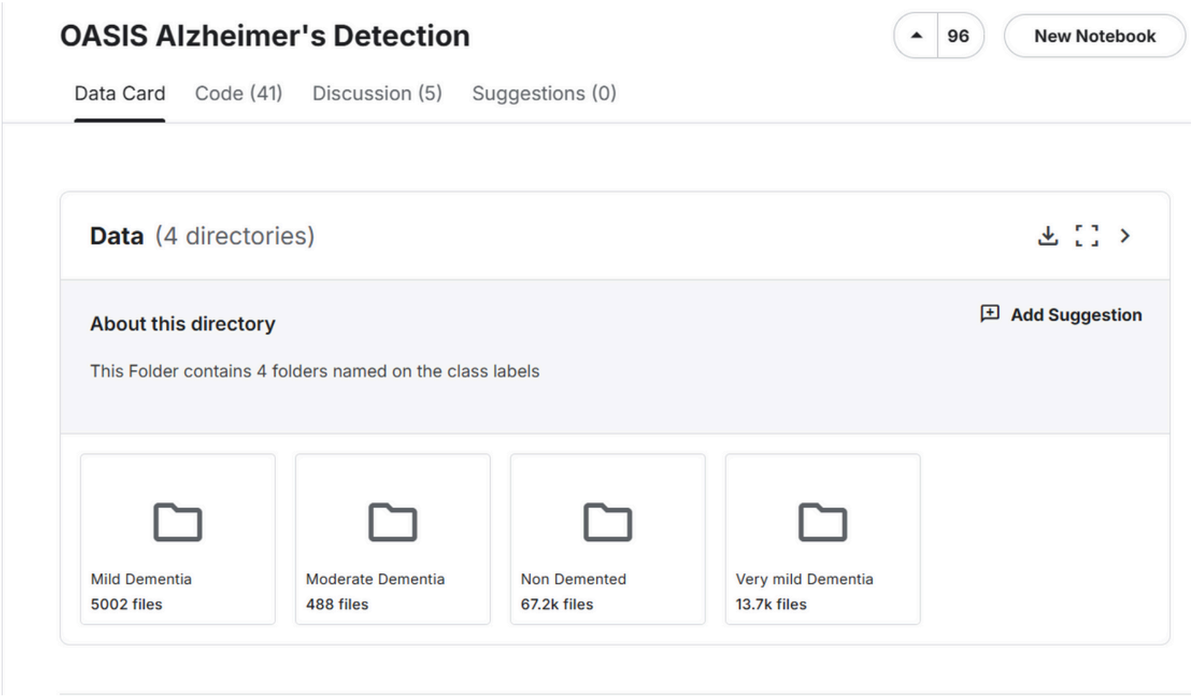


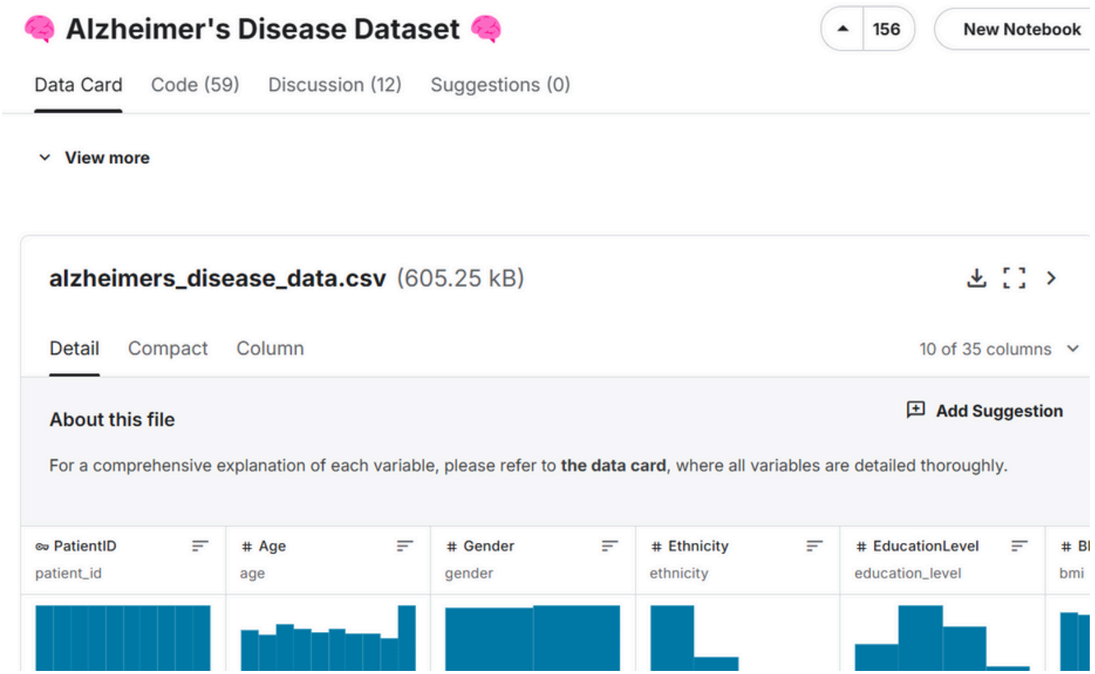
# Completion & What's left?

55%

| Completed | Remaining on PP2 |
|---|---|
| Data Gathering | Evaluating the model |
| Data preprocessing | Increasing the accuracy |
| Model selecting | Implement the hybrid model |
| Basic model training | Finish remaining frontend parts |
| | |

# Data gathering



# Data Preprocessing

```python
# Check for missing values
print("\nMissing Values:")
print(data.isnull().sum())

# Handle missing values (drop rows with missing values for simplicity)
data = data.dropna()

# Encode categorical variables (if any)
label_encoders = {}
for column in data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le

# Feature selection and splitting the dataset
X = data.drop('Diagnosis', axis=1)  # Features (all columns except target)
X = X.drop('PatientID', axis=1)  # Drop the 'Patient ID' column
X= X.drop('DoctorInCharge', axis=1)  # Drop the 'DoctorInCharge' column
```

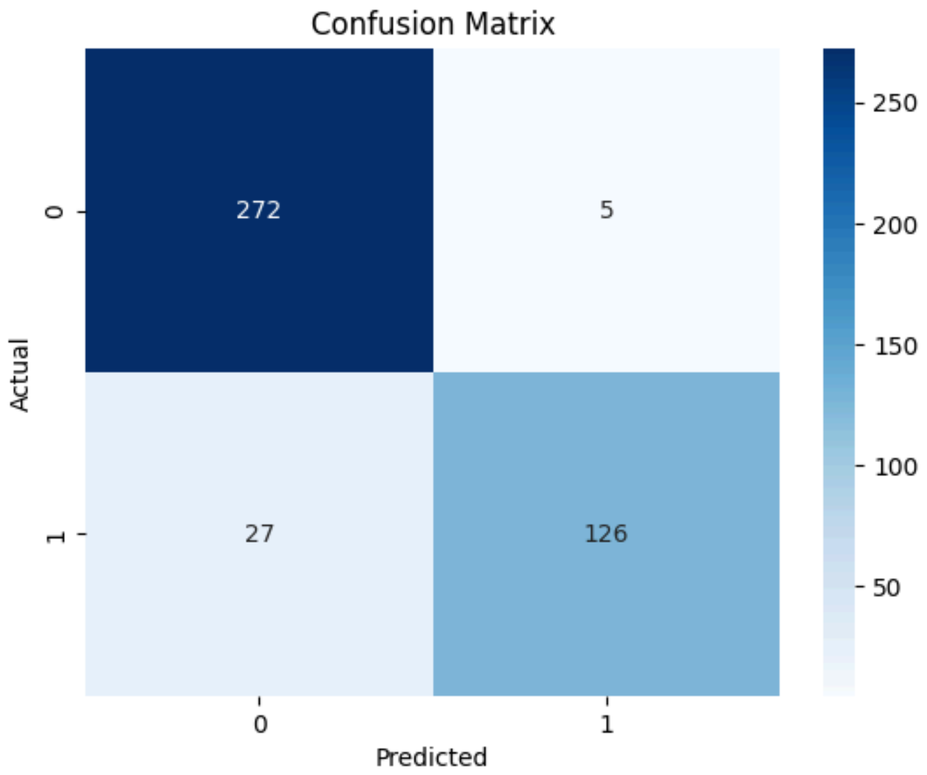
```python
# Import necessary libraries
import numpy as np
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

# Reshape the labels to match the requirements of RandomOverSampler
labels = train_generator.classes
labels = labels.reshape(-1, 1)  # Reshape to 2D array as required by RandomOverSampler

# Initialize the RandomOverSampler
oversampler = RandomOverSampler(random_state=42)

# Get the filenames and labels for oversampling
filenames = np.array(train_generator.filepaths)  # Get filepaths of images

# Perform oversampling
oversampled_filenames, oversampled_labels = oversampler.fit_resample(filenames.reshape(-1, 1), labels)

# Flatten the oversampled_filenames back to a 1D array
oversampled_filenames = oversampled_filenames.flatten()

# Display the new distribution
oversampled_class_counts = Counter(oversampled_labels.flatten())
print(f"Original class counts: {class_counts}")
print(f"New class counts after oversampling: {oversampled_class_counts}")

# Update the train_datagen to use the oversampled data
from tensorflow.keras.preprocessing.image import array_to_img, img_to_array, load_img
```

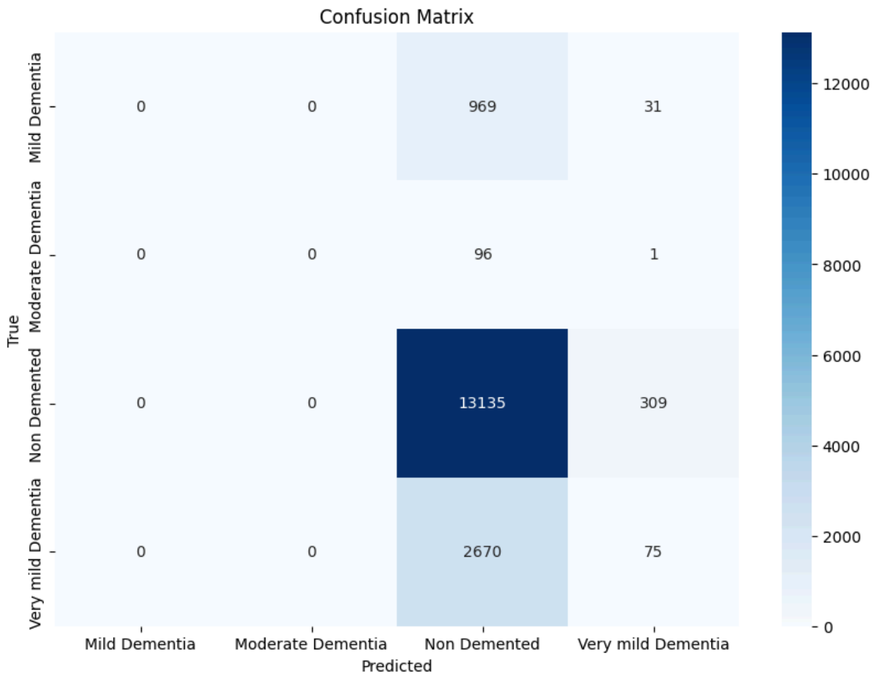
# Model Training

```python
# Build the CNN model
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dropout(0.5),
    Dense(512, activation='relu'),
    Dense(4, activation='softmax')
])

# Compile the model
# Compile the model with the corrected optimizer parameter
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(learning_rate=0.001),
              metrics=['accuracy'])
```

```python
# Scale features for better performance
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Train a Random Forest Classifier
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

# Evaluate the model
y_pred = rf_model.predict(X_test)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

print("\nAccuracy Score:")
print(accuracy_score(y_test, y_pred))

# Feature importance visualization
feature_importances = pd.DataFrame({
    'Feature': X.columns if isinstance(X, pd.DataFrame) else data.drop('Diagnosis', axis=1).columns,
```

# Data validation



# References

[1] *A. Ebrahimi-Ghahnavieh, S. Luo and R. Chiong, "Transfer Learning for Alzheimer's Disease Detection on MRI Images," 2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), Bali, Indonesia, 2019, pp. 133-138*

[2] *C. -E. Kuo, Y. -C. Liu, D. -W. Chang, C. -P. Young, F. -Z. Shaw and S. -F. Liang, "Development and Evaluation of a Wearable Device for Sleep Quality Assessment," in IEEE Transactions on Biomedical Engineering, vol. 64, no. 7, pp. 1547-1557*

[3] *Simon A., Deo M., Selvam V., and Babu R., An overview of machine learning and its applications, International Journal of Electrical Sciences & Engineering. (2016)* ***1****, 22–24.*

# Early Detection of Parkinson's: Harnessing Machine Learning for Predictive Risk Modeling


## IT21164026| Wijayasooriya W.A.K.A

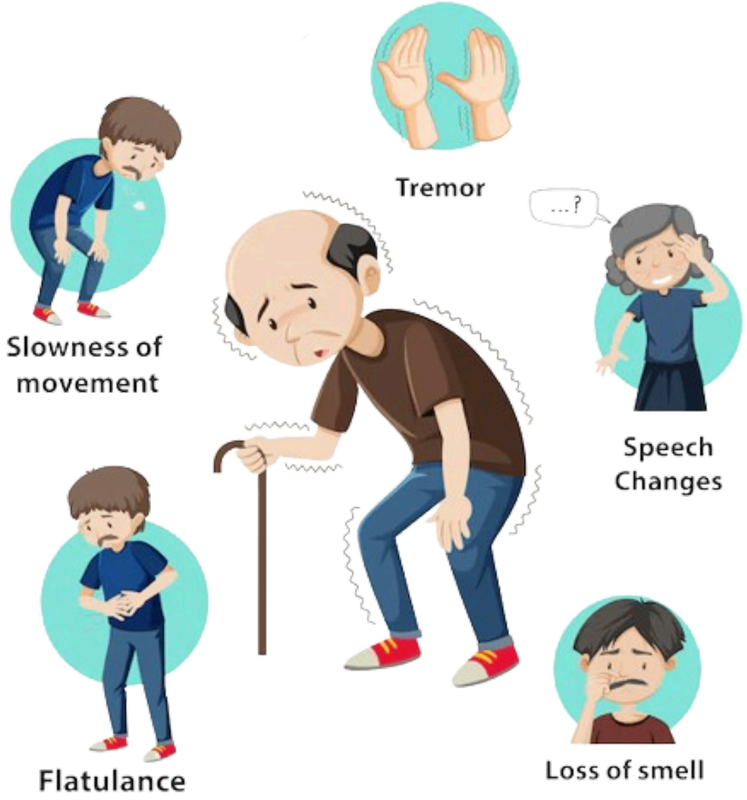
Bachelor of Science (Hons) in Information Technology
Specializing in Data Science

# Introduction & Background

## Component 2

## What is Parkinson’s disease?

- Parkinson's disease is a progressive neurodegenerative disorder affecting millions worldwide. It primarily impacts motor functions due to the loss of dopamine-producing neurons in the brain.



# Separation of Feature Domains

**Each dataset represents a distinct feature domain.**

•Handwriting Samples Model: Focuses on motor-related symptoms (e.g., tremors or irregularities in handwriting)

•DVoice Audio Model: Targets speech-related symptoms (e.g., monotone speech, voice tremors).

•Clinical Data Model: Leverages demographic and lifestyle factors for broader risk prediction.

# Evidence Of Completion

# Predictive Model for Early Identification of Parkinson's Disease Risk Using Clinical Data

## Overview of the Dataset

```python
df = pd.read_csv('../../Datasets/parkinsons_disease_data.csv')
df.head()
```

[4] Python

| | PatientID | Age | Gender | Ethnicity | EducationLevel | BMI | Smoking | AlcoholConsumption | PhysicalActivity | DietQuality | ... | FunctionalAssessment | Tremor | Rigidity | Bradykinesia | PosturalInstability | SpeechPr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 3058 | 85 | 0 | 3 | 1 | 19.619878 | 0 | 5.108241 | 1.380660 | 3.893969 | ... | 1.572427 | 1 | 0 | 0 | 0 | |
| 1 | 3059 | 75 | 0 | 0 | 2 | 16.247339 | 1 | 6.027648 | 8.409804 | 8.513428 | ... | 4.787551 | 0 | 1 | 0 | 1 | |
| 2 | 3060 | 70 | 1 | 0 | 0 | 15.368239 | 0 | 2.242135 | 0.213275 | 6.498805 | ... | 2.130686 | 1 | 0 | 0 | 0 | |
| 3 | 3061 | 52 | 0 | 0 | 0 | 15.454557 | 0 | 5.997788 | 1.375045 | 6.715033 | ... | 3.391288 | 1 | 1 | 1 | 0 | |
| 4 | 3062 | 87 | 0 | 0 | 1 | 18.616042 | 0 | 9.775243 | 1.188607 | 4.657572 | ... | 3.200969 | 0 | 0 | 0 | 1 | |

5 rows × 35 columns

```python
df.shape
```

[6]

```
(2105, 35)
```

# Data Preprocessing

## Handle Missing Values

```
[5]
df.isnull().sum()
```

```
PatientID                 0
Age                       0
Gender                    0
Ethnicity                 0
EducationLevel            0
BMI                       0
Smoking                   0
AlcoholConsumption        0
PhysicalActivity          0
DietQuality               0
SleepQuality              0
```

## Drop Unrelated Columns

```
# drop unrealted columns
df.drop(['PatientID','DoctorInCharge'],axis=1,inplace=True)
df.head()
```

| | Age | Gender | Ethnicity | EducationLevel | BMI | Smoking | AlcoholConsumption | PhysicalActivity | DietQuality | SleepQuality | ... | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 85 | 0 | 3 | 1 | 19.619878 | 0 | 5.108241 | 1.380660 | 3.893969 | 9.283194 | ... | 29 |
| 1 | 75 | 0 | 0 | 2 | 16.247339 | 1 | 6.027648 | 8.409804 | 8.513428 | 5.602470 | ... | 12 |
| 2 | 70 | 1 | 0 | 0 | 15.368239 | 0 | 2.242135 | 0.213275 | 6.498805 | 9.929824 | ... | 29 |
| 3 | 52 | 0 | 0 | 0 | 15.454557 | 0 | 5.997788 | 1.375045 | 6.715033 | 4.196189 | ... | 21 |
| 4 | 87 | 0 | 0 | 1 | 18.616042 | 0 | 9.775243 | 1.188607 | 4.657572 | 9.363925 | ... | 8 |

5 rows × 33 columns

## Label Encoding

```
l encdoing

klearn.preprocessing import LabelEncoder


abelEncoder()
agnosis'] = le.fit_transform(df['Diagnosis'])
```

## Feature Selection

```
[13]
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest

# Assume X is the feature matrix and y is the target
X = df.drop(columns=['Diagnosis'])
y = df['Diagnosis']

# Select top 10 features using chi2
chi2_selector = SelectKBest(chi2, k=10)
X_selected = chi2_selector.fit_transform(X, y)
print("Selected Features:", chi2_selector.get_feature_names_out())
```

```
Selected Features: ['Age' 'AlcoholConsumption' 'CholesterolTriglycerides' 'UPDRS' 'MoCA'
 'FunctionalAssessment' 'Tremor' 'Rigidity' 'Bradykinesia'
 'PosturalInstability']
```

# Model Training

- **Decision Tree**
- **Random Forest**
- **XGBoost**

```python
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# Define hyperparameter grids
param_grids = {
    "Decision Tree": {
        "criterion": ["gini", "entropy"],
        "max_depth": [3, 5, 10, None],
        "min_samples_split": [2, 5, 10],
        "min_samples_leaf": [1, 2, 4]
    },
    "Random Forest": {
        "n_estimators": [100, 200, 300],
        "max_depth": [5, 10, None],
        "min_samples_split": [2, 5, 10],
        "min_samples_leaf": [1, 2, 4]
    },
    "XGBoost": {
        "n_estimators": [100, 200, 300],
        "max_depth": [3, 5, 10],
        "learning_rate": [0.01, 0.1, 0.2],
        "subsample": [0.6, 0.8, 1.0],
        "colsample_bytree": [0.6, 0.8, 1.0]
    }
}
```

[26]

## Best Model

```python
# Find the best model based on Test Accuracy
best_model_name = max(results, key=lambda x: results[x]["Test Accuracy"])
best_model = best_models[best_model_name]

print(f"\nBest Model: {best_model_name} with Test Accuracy: {results[best_model_name]['Te
print(f"Best Hyperparameters for {best_model_name}: {results[best_model_name]['Best Param
```

[28]

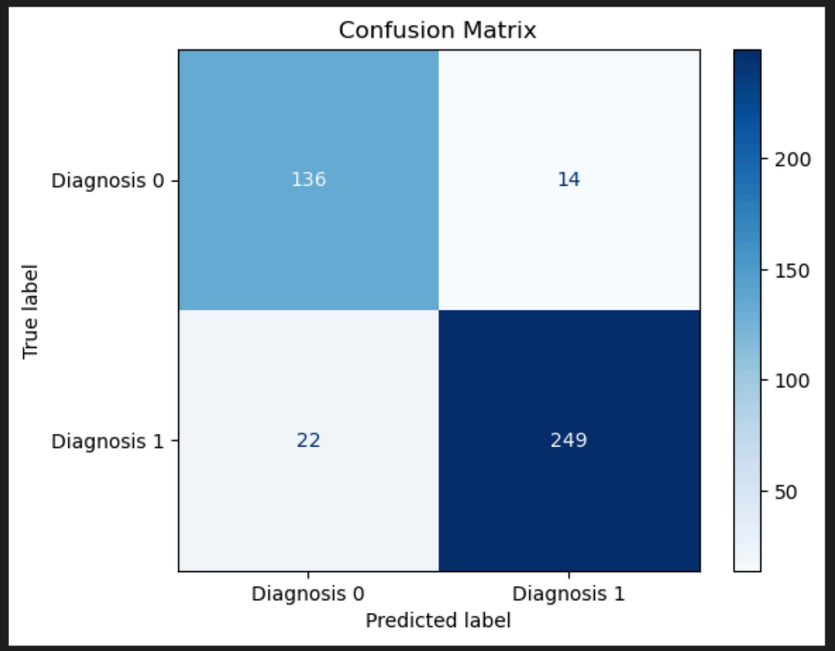
```
...
Best Model: Random Forest with Test Accuracy: 0.9145
Best Hyperparameters for Random Forest: {'max_depth': 10, 'min_samples_leaf': 1, 'min_sample
```

# Performance Evaluation

```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt


y_pred = best_model.predict(X_test)

# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=[0, 1])

# Display confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Diagnosis 0", "Diagnosis 1"])
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()
```

[30]

...



# Speech-Based Predictive Model for Early Parkinson's Disease Risk Identification

## Overview of the Dataset

```
[25]
!ls "/content/drive/My Drive/Kawindu_data_sets/PD_AH - With Parkinson/PD_AH"
```

```
AH_545616858-3A749CBC-3FEB-4D35-820E-E45C3E5B9B6A.wav
AH_545622717-461DFFFE-54AF-42AF-BA78-528BD505D624.wav
AH_545622718-C052AD58-5E6B-4ADC-855C-F76B66BAFA6E.wav
AH_545622719-52C23861-6E0D-41E0-A3D8-9358C28C019B.wav
AH_545622720-E1486AF6-8C95-47EB-829B-4D62698C987A.wav
AH_545622722-3C79DA68-36BB-43A2-B29C-61AEF480E07E.wav
AH_545629296-C2C009C6-8C17-42EA-B6BE-362942FC4692.wav
AH_545643618-82A143AC-B643-4273-A923-C42A83AEEC5F.wav
AH_545648867-CB17D873-1CEA-492A-B5B0-93C7463F516C.wav
AH_545692309-EA8C4DC0-9B2A-4CC7-A490-851A2129A733.wav
AH_545692315-C2972597-9AEC-4060-A186-F1F59340640C.wav
AH_545713221-1E77C030-4558-4A88-B1A2-6AB777ACAE61.wav
AH_545713222-DA13DC3A-F24B-454E-984F-19DF19328D39.wav
AH_545713223-E6D59EE5-4C3F-4B40-AE8F-0657EF94DB66.wav
AH_545713224-1B3708B0-8792-4FEE-B03B-C7CB9CB03D58.wav
AH_545743929-E2EAE1A3-7E46-4DCF-8DB7-37A5CA47DB9D.wav
AH_545753013-FCFF8F46-08FF-4C87-B443-D2039E5DA945.wav
AH_545753014-C68926CC-AB91-49AF-90A6-BB5C434283DB.wav
AH_545753015-58CAA743-BA9A-47E0-B9EF-CC35E9EFB839.wav
AH_545789668-A4F6069C-5E1A-49F5-9EDC-59C6EB833E42.wav
AH_545789670-C297FD53-BF71-4183-86A0-58E5E1EB0DF8.wav
AH_545789671-794D2256-DDFF-4009-8BA8-8A306C8FA14F.wav
AH_545789674-53885025-35F1-48C1-9826-BAAEB8BEAF58.wav
AH_545789675-243F18DB-4432-4C87-B12C-6EEC2D2D30D6.wav
AH_545789677-D381D801-B073-4945-BE0D-E250126EA6B1.wav
...
AH_545841226-C699FC9E-1E0C-474D-A12A-936DD92B8980.wav
AH_545841227-5C77713A-66F1-49D0-BC8A-702C152E668D.wav
AH_545847410-D1BA3BB4-1F61-44CA-ACDE-455A8E97E04B.wav
AH_545880204-EE87D3E2-0D4C-4EAA-ACD7-C3F177AFF62F.wav
```

```
[88]
metadata = pd.read_csv(csv_path)
metadata.head()
```

| | Sample ID | Label | Age | Sex |
|---|---|---|---|---|
| 0 | AH_064F_7AB034C9-72E4-438B-A9B3-AD7FDA1596C5 | HC | 69 | M |
| 1 | AH_114S_A89F3548-0B61-4770-B800-2E26AB3908B6 | HC | 43 | M |
| 2 | AH_121A_BD5BA248-E807-4CB9-8B53-47E7FFE5F8E2 | HC | 18 | F |
| 3 | AH_123G_559F0706-2238-447C-BA39-DB5933BA619D | HC | 28 | M |
| 4 | AH_195B_39DA6A45-F4CC-492A-80D4-FB79049ACC22 | HC | 68 | M |

# Data Preprocessing

## Check the balance of the dataset

```
[103]
#extract the number of roecord with type of label
audio_metadata['Label'].value_counts()
```

...

| Label | count |
|---|---|
| HC | 164 |
| PwPD | 160 |

**dtype:** int64

## Feature Extraction

```
def extract_features(audio_metadata):
    features = []
    labels = []
    for _, row in audio_metadata.iterrows():
        features.append(extract_yamnet_features(row['File Path']))
        labels.append(row['Label'])
    return np.array(features), np.array(labels)

features, labels = extract_features(audio_metadata)
```

## Voice Augmentation

- **Time Stretch**
- **Pitch Shift**

```
# Augment audio with noise, pitch shift, and time stretch
def augment_audio(file_path):
    audio, sr = librosa.load(file_path, sr=None)

    # Augmentation methods
    augmented_audios = [
        audio + np.random.normal(0, 0.02, audio.shape),
        librosa.effects.time_stretch(audio, rate=1.2),
        librosa.effects.pitch_shift(y=audio, sr=sr, n_steps=2),
    ]
    return augmented_audios, sr
```

# Model Training

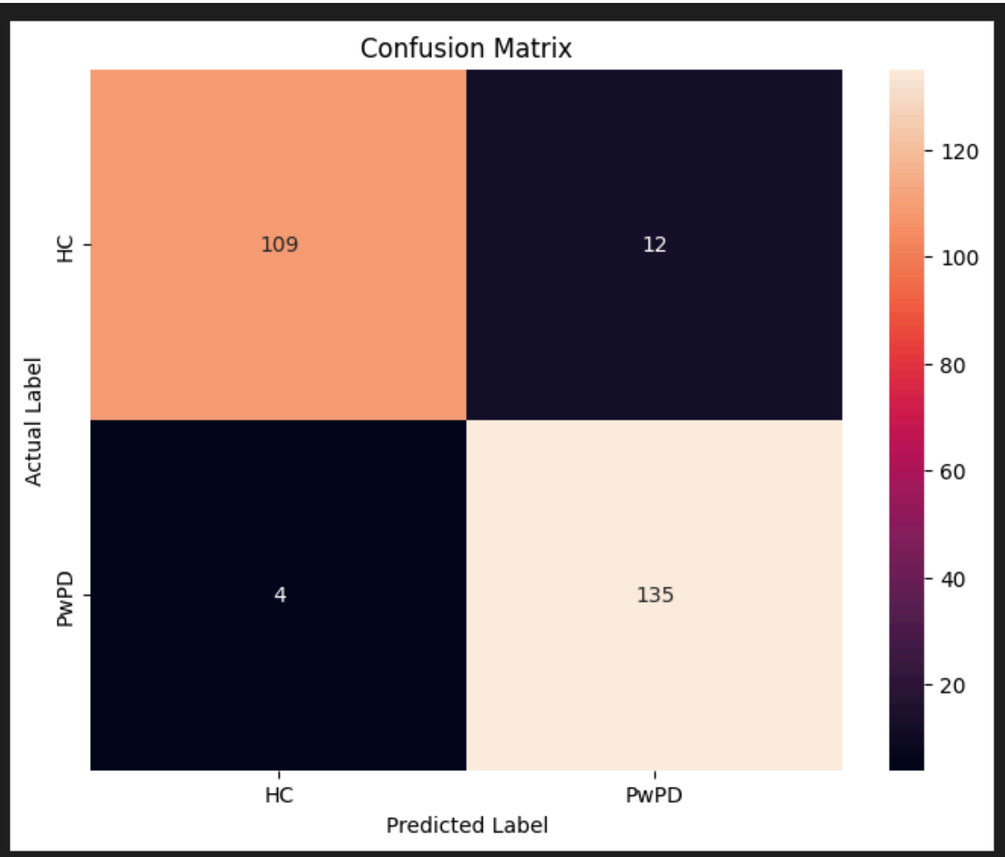
## Used Transfer Learning Model

```python
def create_transfer_learning_model(input_shape):
    model = models.Sequential([
        layers.Dense(128, activation='relu', input_shape=input_shape),
        layers.Dropout(0.5),
        layers.Dense(64, activation='relu'),
        layers.Dense(1, activation='sigmoid')  # Binary classification
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy']
    return model

model = create_transfer_learning_model((features.shape[1],))
```

```python
history = model.fit(
    X_train, y_train,
    epochs=20,
    batch_size=16,
    validation_data=(X_test, y_test)
)
```

```
Epoch 1/20
65/65 [==============================] - 1s 7ms/step - loss: 0.6372 - accuracy: 0.6322 - val_loss:
Epoch 2/20
65/65 [==============================] - 0s 4ms/step - loss: 0.5520 - accuracy: 0.7220 - val_loss:
Epoch 3/20
65/65 [==============================] - 0s 4ms/step - loss: 0.4757 - accuracy: 0.7770 - val_loss:
Epoch 4/20
65/65 [==============================] - 0s 4ms/step - loss: 0.4326 - accuracy: 0.8012 - val_loss:
Epoch 5/20
65/65 [==============================] - 0s 4ms/step - loss: 0.3732 - accuracy: 0.8282 - val_loss:
Epoch 6/20
65/65 [==============================] - 0s 4ms/step - loss: 0.3574 - accuracy: 0.8359 - val_loss:
Epoch 7/20
65/65 [==============================] - 0s 4ms/step - loss: 0.3153 - accuracy: 0.8571 - val_loss:
Epoch 8/20
65/65 [==============================] - 0s 4ms/step - loss: 0.2756 - accuracy: 0.8793 - val_loss:
Epoch 9/20
65/65 [==============================] - 0s 4ms/step - loss: 0.2817 - accuracy: 0.8755 - val_loss:
Epoch 10/20
65/65 [==============================] - 0s 4ms/step - loss: 0.2460 - accuracy: 0.9054 - val_loss:
Epoch 11/20
65/65 [==============================] - 0s 4ms/step - loss: 0.2096 - accuracy: 0.9064 - val_loss:
Epoch 12/20
65/65 [==============================] - 0s 4ms/step - loss: 0.1988 - accuracy: 0.9228 - val_loss:
Epoch 13/20
...
Epoch 19/20
65/65 [==============================] - 0s 4ms/step - loss: 0.1231 - accuracy: 0.9517 - val_loss:
Epoch 20/20
```

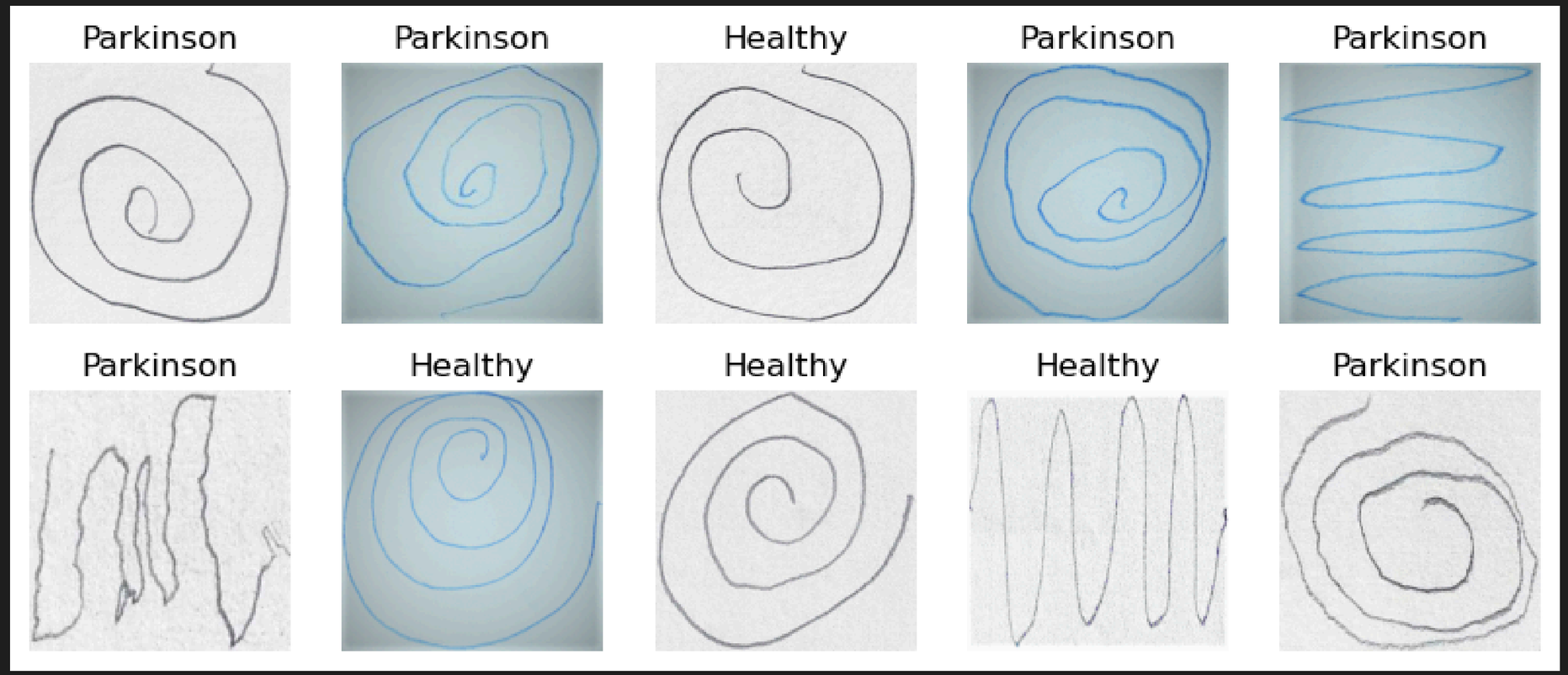
# Performance Evaluation




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| HC | 0.96 | 0.90 | 0.93 | 121 |
| PwPD | 0.92 | 0.97 | 0.94 | 139 |
| accuracy | | | 0.94 | 260 |
| macro avg | 0.94 | 0.94 | 0.94 | 260 |
| weighted avg | 0.94 | 0.94 | 0.94 | 260 |

# Hand-Writing Samples Predictive Model for Early Parkinson's Disease Risk Identification

## Over 3000 Images




**IT21164026 | Wijayasooriya W.A.K.A | 24-25J-322**

# Data Preprocessing

## Resize & Normalize Image

```python
from re import sub

train_data = tf.keras.utils.image_dataset_from_directory(
    dir,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    subset='training',
    validation_split=0.1,
    seed=42
)
validation_data = tf.keras.utils.image_dataset_from_directory(
    dir,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    subset='validation',
    validation_split=0.1,
    seed=42
)

test_data = tf.keras.utils.image_dataset_from_directory(
    dir,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    subset='validation',
    validation_split=0.1,
    seed=42
)
```

[4]

```python
train_data = train_data.map(lambda x, y: (x / 255, y))
validation_data = validation_data.map(lambda x, y: (x / 255, y))
test_data = test_data.map(lambda x, y: (x / 255, y))
```

```python
for image,label in train_data.take(1) :
  for i in range(1):
      print(image)
```

```
Tensor(
[[0.91074437 0.91074437 0.91074437]
 [0.9470588  0.9470588  0.9470588 ]
 [0.9229692  0.9229692  0.9229692 ]
 ...
 [0.92579055 0.92579055 0.92579055]
 [0.92655057 0.92655057 0.92655057]
 [0.89371693 0.89371693 0.89371693]]

[[0.93313324 0.93313324 0.93313324]
 [0.9343937  0.9343937  0.9343937 ]
 [0.917447   0.917447   0.917447  ]
 ...
 [0.9155263  0.9155263  0.9155263 ]
 [0.91542614 0.91542614 0.91542614]
 [0.9112244  0.9112244  0.9112244 ]]
```

# Model Training

## Used Transfer Learning Model

```python
base_model = tf.keras.applications.MobileNetV2(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet'
)


base_model.trainable = False


model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])


model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Summary of the model
model.summary()

model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=['accuracy'])
```

```python
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)

model_checkpoint = ModelCheckpoint(
    'best_model.keras',
    monitor='val_loss',
    save_best_only=True,
    verbose=1
)

history = model.fit(train_data,
                    epochs=5,
                    validation_data=validation_data,
                    callbacks=[early_stopping, model_checkpoin
                    batch_size = 32
                    )


model.save('parkinson_detection_model.h5')
```

# Performance Evaluation

## Training Accuracy & Validation Accuracy

```
h 1/5
2 ━━━━━━━━━━━━━━━━━━━━ 0s 622ms/step - accuracy: 0.7383 - loss: 0.5226
h 1: val_loss improved from inf to 0.24858, saving model to best_model.keras
2 ━━━━━━━━━━━━━━━━━━━━ 76s 763ms/step - accuracy: 0.7390 - loss: 0.5217 - val_accuracy: 0.8988 - val_loss: 0.2486
h 2/5
2 ━━━━━━━━━━━━━━━━━━━━ 0s 600ms/step - accuracy: 0.8755 - loss: 0.2946
h 2: val_loss improved from 0.24858 to 0.17169, saving model to best_model.keras
2 ━━━━━━━━━━━━━━━━━━━━ 62s 672ms/step - accuracy: 0.8757 - loss: 0.2943 - val_accuracy: 0.9479 - val_loss: 0.1717
h 3/5
2 ━━━━━━━━━━━━━━━━━━━━ 0s 592ms/step - accuracy: 0.9026 - loss: 0.2295
h 3: val_loss improved from 0.17169 to 0.15361, saving model to best_model.keras
2 ━━━━━━━━━━━━━━━━━━━━ 62s 667ms/step - accuracy: 0.9026 - loss: 0.2294 - val_accuracy: 0.9448 - val_loss: 0.1536
h 4/5
2 ━━━━━━━━━━━━━━━━━━━━ 0s 603ms/step - accuracy: 0.9230 - loss: 0.1991
h 4: val_loss improved from 0.15361 to 0.11844, saving model to best_model.keras
2 ━━━━━━━━━━━━━━━━━━━━ 62s 673ms/step - accuracy: 0.9231 - loss: 0.1990 - val_accuracy: 0.9632 - val_loss: 0.1184
h 5/5
2 ━━━━━━━━━━━━━━━━━━━━ 0s 577ms/step - accuracy: 0.9377 - loss: 0.1572
h 5: val_loss improved from 0.11844 to 0.10694, saving model to best_model.keras
2 ━━━━━━━━━━━━━━━━━━━━ 60s 649ms/step - accuracy: 0.9378 - loss: 0.1571 - val_accuracy: 0.9755 - val_loss: 0.1069
ING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g
```

```
evaluation = model.evaluate(validation_data)
print(f"Validation Loss: {evaluation[0]}, Validation Accuracy: {evaluation[1]}")
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 6s 562ms/step - accuracy: 0.9821 - loss: 0.0953
Validation Loss: 0.10693712532520294, Validation Accuracy: 0.9754601120948792
```

# Completion & What's left?

55%

| Completed | Remaining on PP2 |
|---|---|
| Data Gathering | Model Enhancement |
| Clinical Data Model Building | Multimodel Approach |
| Speech Data Model Building | Front-end Development |
| Handwriting Data Model Building | Data Visualization Report |
| Integrated the Backend | Novelty Part |

# References

[1] *Prashanth, R., & Roy, S. D. (2018). Early detection of Parkinson's disease through patient questionnaire and predictive modelling. International journal of medical informatics, 119, 75-87.*

[2] *Govindu, A., & Palwe, S. (2023). Early detection of Parkinson's disease using machine learning. Procedia Computer Science, 218, 249-261.*

[3] *Wang, W., Lee, J., Harrou, F., & Sun, Y. (2020). Early detection of Parkinson's disease using deep learning and machine learning. IEEE Access, 8, 147635-147646.*

[4] *Shaban, M. (2023). Deep learning for Parkinson's disease diagnosis: a short survey. Computers, 12(3), 58.*

# Early Autism Spectum Disorder Detection Using Facial Images and Text Analysis

## IT21182846 | Chamaleen D.B.N

Bachelor of Science (Hons) in Information Technology
Specializing in Data Science

# Introduction & Background

## Component 3

### What is Autism Disorder?

Autism spectrum disorder is a complex neurodevelopment condition characterized by diverse behavioral, communication, and social challenges. Its prevalence is significant globally, affecting approximately one in 160 children according to the World Health Organization. ASD diagnosis typically occurs around the age of 4.

# Introduction & Background

## What is ASD?

A neurodevelopmental disorder affecting social interaction, communication, and behavior.

## Why Early Identification of ASD?

- Reduces Clinical Burden – Helps doctors focus on real cases, avoiding unnecessary tests.
- Faster Support – Early diagnosis allows quicker therapy and better outcomes.
- Lower Costs – Reduces long-term medical and education expenses.
- Better Life Quality – Early help improves social and learning skills.

# Research Question?

- How to identify autism patient face detections in an application?
- How to develop self-manageable platform?
- How can we collect more and better-quality data to improve  ML models for early ASD diagnosis?
- How to gather text base questions' answers?

# Research Question?

### Challenges in Existing ASD Detection Methods

- Delayed Diagnosis – Traditional screening takes time and requires expert evaluation.
- Questionnaire answers may not always be accurate – People can misunderstand or give biased responses.
- Text-only data may miss key signs – Facial expressions and behaviors provide important clues that text responses can't capture.

### Why Combine Facial Images & Questionnaire Data?

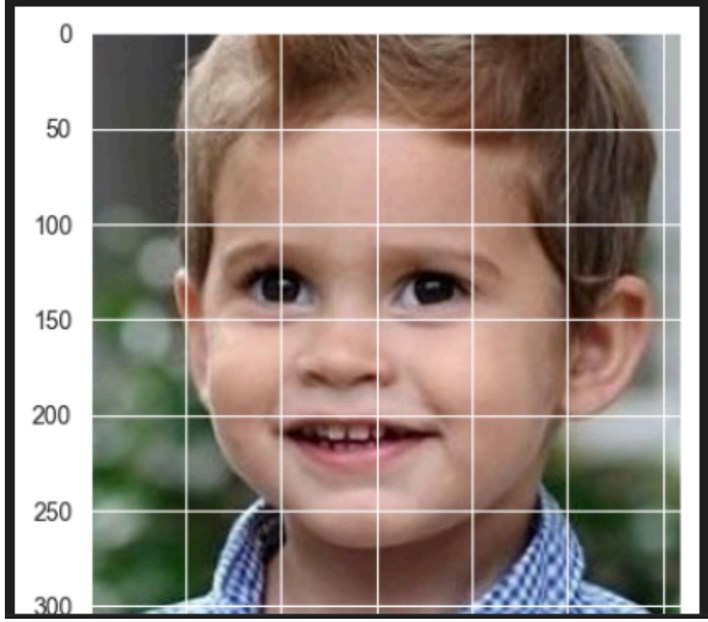
- Improves Accuracy – Combining both methods gives a more reliable diagnosis.
- Captures More Features – Facial expressions and behavior patterns complement questionnaire responses.
- Faster Screening – Reduces dependency on clinical visits and speeds up early intervention.

# Evidence of Completion

# Facial Image-Based Predictive Model for Early ASD Detection

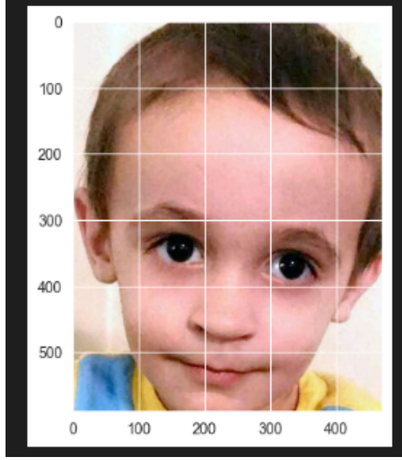
## Overview of the Dataset

```
# Plot first 3 images from train_imgs
import matplotlib.image as mpimg
for ima in train_imgs[0:3]:
    img=mpimg.imread(ima)
    imgplot = plt.imshow(img)
    plt.show()
```

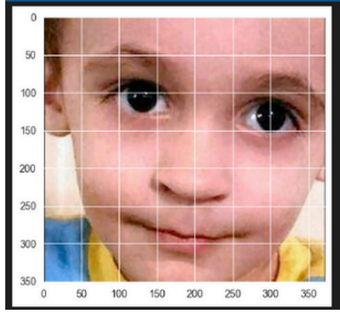


# Data Preprocessing

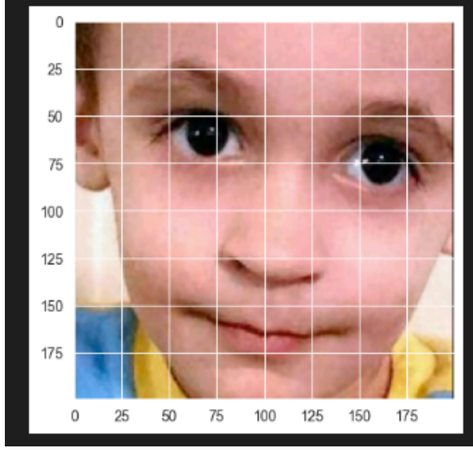
```
sample2=[]
def sampledata():
    for i in training_data[0:2]:
        img=cv2.imread(i)
        sample=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        imgplot = plt.imshow(sample)
        plt.show()
        sample2.append(sample)
    return sample2
sample_data=sampledata()
```



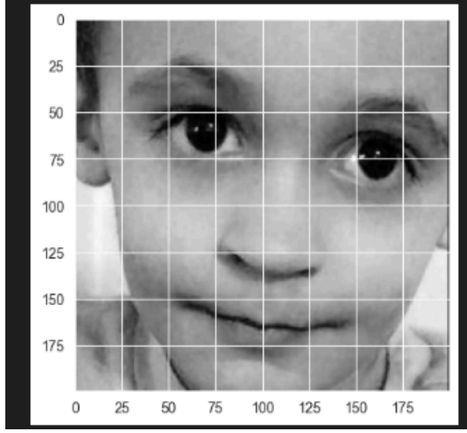

```
detectedimages=[]
def cropping_based_face_detection(samdata):
    detector = dlib.get_frontal_face_detector()
    greyimg=cv2.cvtColor(samdata, cv2.COLOR_BGR2GRAY)
    detectedimg = detector(greyimg)
    #print("found {} face(s)".format(len(detectedimg)))
    cropped = None  # Initialize cropped to None outside the loop
    for face in detectedimg:
        x1 = face.left()
        y1 = face.top()
        x2 = face.right()
        y2 = face.bottom()
        img=cv2.rectangle(samdata, (x1, y1), (x2, y2), (0, 0, 0), 2)
        #imgplot = plt.imshow(img)
        #plt.show()
        cropped=img[face.top():face.bottom(),face.left():face.right()]
        imgplot = plt.imshow(cropped)
        plt.show()
        detectedimages.append(cropped) # Move append inside loop to en
    # If no faces are detected, append the original image
    if cropped is None:
        detectedimages.append(samdata)
    return detectedimages
for i in sample_data:
    det_img=cropping_based_face_detection(i)
```



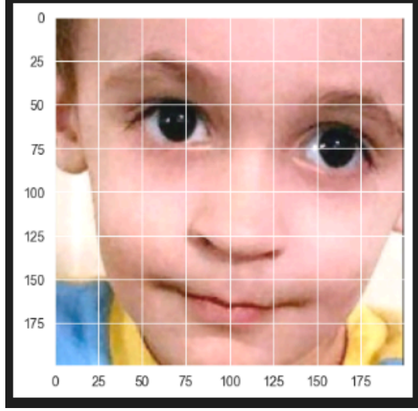

```
resizedimg=[]
def ReSize(detimg):
    img=cv2.resize(detimg,(200,200))
    imgplot = plt.imshow(img)
    plt.show()
    resizedimg.append(img)
    return resizedimg

for i in det_img:
    res_img=ReSize(i)
```



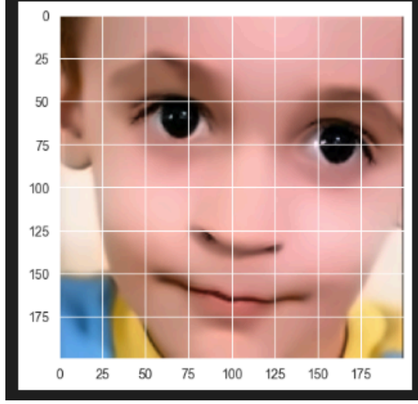
```python
grey=[]
def gray(resimg):
    img=cv2.cvtColor(resimg, cv2.COLOR_BGR2GRAY)
    I=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    imgplot = plt.imshow(I)
    plt.show()
    grey.append(I)
    return grey

for i in res_img:
    grey_img=gray(i)
```




```python
bc=[]
def Brightnessandcontrast(resimg):
    Bright = 20
    Contrast = 1
    img=cv2.cvtColor(resimg,cv2.COLOR_BGR2RGB)
    adjusted = cv2.convertScaleAbs(img, alpha=Contrast, beta=Bright)
    rgbadjusted=cv2.cvtColor(adjusted,cv2.COLOR_BGR2RGB)
    imgplot = plt.imshow(rgbadjusted)
    plt.show()
    bc.append(rgbadjusted)
    return bc

for i in res_img:
    b_c=Brightnessandcontrast(i)
```




```python
blurred=[]
def noise_reduction(resimg):
    img=cv2.cvtColor(resimg,cv2.COLOR_BGR2RGB)
    filteredimg = cv2.fastNlMeansDenoisingColored(img,None,10,10,7,21)
    rgbfilteredimg=cv2.cvtColor(filteredimg,cv2.COLOR_BGR2RGB)
    imgplot = plt.imshow(rgbfilteredimg)
    plt.show()
    blurred.append(rgbfilteredimg)
    return blurred

for i in res_img:
    blur_img=noise_reduction(i)
```



```python
# Set the dimensions for images
nrows = 150
ncolumns  = 150
channels = 3

# Read and process the images: Function returns X,y. X - list of resized images, y - list of labels for the images

def read_and_process_image(list_of_images):
    X = []
    y = []

    for image in list_of_images:
        X.append(cv2.resize(cv2.imread(image, cv2.IMREAD_COLOR), (nrows, ncolumns), interpolation = cv2.INTER_CUBIC))
        if 'Non_Autistic' in image:
            y.append(0)
        else:
            y.append(1)

    return X,y
```

```python
# Get resized images and labels from train data
X_train, y_train = read_and_process_image(train_imgs)

# Delete train images to save space
del train_imgs
gc.collect()
```

```
40520
```

# Model Architecture

## Facial Image Model

```python
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import (GlobalAveragePooling2D, Dense, Dropout, BatchNormalization,
                                     GaussianNoise, Input, MultiHeadAttention, Reshape)
from tensorflow.keras.optimizers import Adam

def create_vgg16_model(input_shape):

    inputs = Input(shape=input_shape)

    base_model = VGG16(weights='imagenet', input_tensor=inputs, include_top=False)

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

    x = base_model.output

    height, width, channels = 7, 7, 512
    x = Reshape((height * width, channels))(x)

    attention_output = MultiHeadAttention(num_heads=8, key_dim=channels)(x, x)
    attention_output = Reshape((height, width, channels))(attention_output)

    x = GaussianNoise(0.25)(attention_output)

    x = GlobalAveragePooling2D()(x)

    x = Dense(512, activation='relu')(x)
    x = BatchNormalization()(x)
    x = GaussianNoise(0.25)(x)
    x = Dropout(0.25)(x)

    outputs = Dense(4, activation='softmax')(x)

    model = Model(inputs=inputs, outputs=outputs)

    return model

input_shape = (224, 224, 3)

cnn_model = create_vgg16_model(input_shape)

cnn_model.compile(optimizer=Adam(learning_rate=0.0001),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
```

```python
from tensorflow.keras.applications import VGG19
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout, BatchNormalization, GaussianNoise, Input, MultiHeadAttention, Reshape
from tensorflow.keras.optimizers import Adam
import tensorflow as tf

def create_vgg19_model(input_shape):

    inputs = Input(shape=input_shape)

    base_model = VGG19(weights='imagenet', input_tensor=inputs, include_top=False)

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

    x = base_model.output

    height, width, channels = 7, 7, 512
    x = Reshape((height * width, channels))(x)

    attention_output = MultiHeadAttention(num_heads=8, key_dim=channels)(x, x)

    attention_output = Reshape((height, width, channels))(attention_output)

    x = GaussianNoise(0.25)(attention_output)

    x = GlobalAveragePooling2D()(x)

    x = Dense(512, activation='relu')(x)
    x = BatchNormalization()(x)
    x = GaussianNoise(0.25)(x)
    x = Dropout(0.25)(x)

    outputs = Dense(4, activation='softmax')(x)

    model = Model(inputs=inputs, outputs=outputs)

    return model

input_shape = (224, 224, 3)

cnn_model = create_vgg19_model(input_shape)

cnn_model.compile(optimizer=Adam(learning_rate=0.00001),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
```

```python
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import (GlobalAveragePooling2D, Dense, Dropout, GaussianNoise,
                                     Input, MultiHeadAttention, Reshape)
from tensorflow.keras.optimizers import Adam

def create_lightweight_model(input_shape):
    inputs = Input(shape=input_shape)
    base_model = MobileNetV2(weights='imagenet', input_tensor=inputs, include_top=False)
    for layer in base_model.layers:
        layer.trainable = False
    x = base_model.output
    height, width, channels = x.shape[1], x.shape[2], x.shape[3]
    x = Reshape((height * width, channels))(x)
    attention_output = MultiHeadAttention(num_heads=4, key_dim=channels)(x, x)
    attention_output = Reshape((height, width, channels))(attention_output)
    x = GaussianNoise(0.2)(attention_output)
    x = GlobalAveragePooling2D()(x)
    x = Dense(256, activation='relu')(x)
    x = GaussianNoise(0.2)(x)
    x = Dropout(0.3)(x)
    outputs = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=inputs, outputs=outputs)
    return model

input_shape = (224, 224, 3)
cnn_model = create_lightweight_model(input_shape)
cnn_model.compile(optimizer=Adam(learning_rate=0.0001),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
```

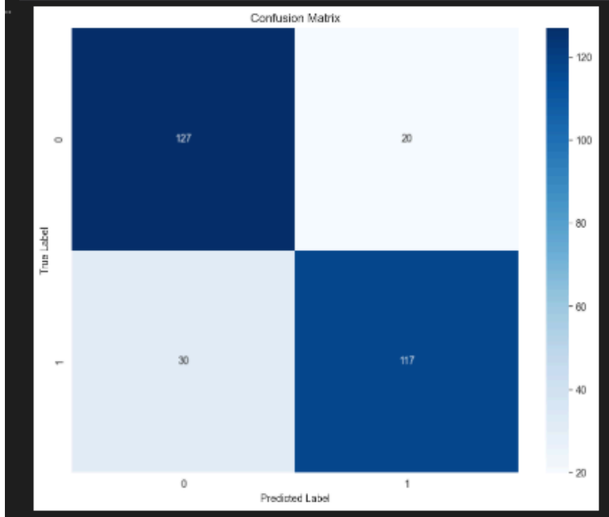
```python
# Check the lengths of test_labels and predicted_labels
print(f"Length of test_labels: {len(test_labels)}")
print(f"Length of predicted_labels: {len(predicted_labels)}")

# Ensure the lengths are consistent
if len(test_labels) != len(predicted_labels):
    # Slice predicted_labels to match test_labels, if necessary
    predicted_labels = predicted_labels[:len(test_labels)]

# Generate and print the classification report
from sklearn.metrics import classification_report
report = classification_report(test_labels, predicted_labels, target_names=list(test_gen_new.class_indices.keys()))
print(report)
```

```
Length of test_labels: 294
Length of predicted_labels: 1176
              precision    recall  f1-score   support

           0       0.50      1.00      0.67       147
           1       0.00      0.00      0.00       147

    accuracy                           0.50       294
   macro avg       0.25      0.50      0.33       294
weighted avg       0.25      0.50      0.33       294
```

```python
report = classification_report(test_labels, predicted_classes, target_names=list(test_gen_new.class_indices.keys()))
print(report)
```

```
              precision    recall  f1-score   support

           0       0.81      0.86      0.84       147
           1       0.85      0.80      0.82       147

    accuracy                           0.83       294
   macro avg       0.83      0.83      0.83       294
weighted avg       0.83      0.83      0.83       294
```

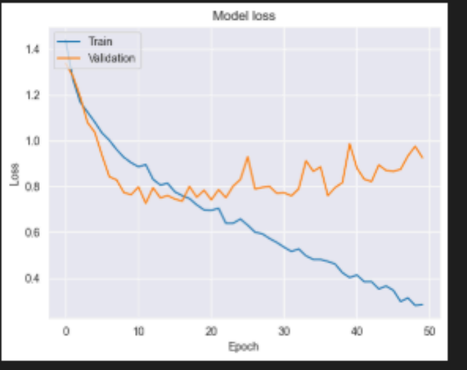
```python
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=list(test_gen_new.class_indices.keys()), yticklabels=list(test_gen_new.class_indices.keys()))
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```




```python
history = cnn_model.fit(
    train_gen_new,
    validation_data=valid_gen_new,
    epochs=50,
    # callbacks=[early_stopping],
    verbose=1
)
```

```
Epoch 1/50
74/74 ━━━━━━━━━━━━━━━━━━━━ 741s 10s/step - accuracy: 0.3356 - loss: 1.5210 - val_accuracy: 0.5204 - val_loss: 1.3363
Epoch 2/50
74/74 ━━━━━━━━━━━━━━━━━━━━ 645s 9s/step - accuracy: 0.4949 - loss: 1.3008 - val_accuracy: 0.7347 - val_loss: 1.2799
Epoch 3/50
74/74 ━━━━━━━━━━━━━━━━━━━━ 569s 8s/step - accuracy: 0.5672 - loss: 1.1979 - val_accuracy: 0.7619 - val_loss: 1.1899
Epoch 4/50
74/74 ━━━━━━━━━━━━━━━━━━━━ 505s 7s/step - accuracy: 0.6046 - loss: 1.1386 - val_accuracy: 0.7755 - val_loss: 1.0784
Epoch 5/50
74/74 ━━━━━━━━━━━━━━━━━━━━ 532s 7s/step - accuracy: 0.6333 - loss: 1.0526 - val_accuracy: 0.7891 - val_loss: 1.0359
Epoch 6/50
74/74 ━━━━━━━━━━━━━━━━━━━━ 514s 7s/step - accuracy: 0.6531 - loss: 1.0482 - val_accuracy: 0.8061 - val_loss: 0.9349
```

```python
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

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



# Text-Based Model

## Overview of the Dataset

| | Case_No | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | Age_Mons | Qchat-10-Score | Sex | Ethnicity | Jaundice | Family_mem_with_ASD | Who completed the test | Class/ASD Traits |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 28 | 3 | f | middle eastern | yes | no | family member | No |
| 1 | 2 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 36 | 4 | m | White European | yes | no | family member | Yes |
| 2 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 36 | 4 | m | middle eastern | yes | no | family member | Yes |
| 3 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 24 | 10 | m | Hispanic | no | no | family member | Yes |
| 4 | 5 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 20 | 9 | f | White European | no | yes | family member | Yes |
| 5 | 6 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 21 | 8 | m | black | no | no | family member | Yes |
| 6 | 7 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 33 | 5 | m | asian | yes | no | family member | Yes |
| 7 | 8 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 33 | 6 | m | asian | yes | no | family member | Yes |
| 8 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 36 | 2 | m | asian | no | no | family member | No |
| 9 | 10 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 22 | 8 | m | south asian | no | no | Health Care Professional | Yes |
| 10 | 11 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 36 | 6 | m | Hispanic | yes | yes | family member | Yes |
| 11 | 12 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 17 | 8 | m | middle eastern | yes | no | family member | Yes |
| 12 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 25 | 0 | f | middle eastern | yes | no | family member | No |
| 13 | 14 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 15 | 7 | f | middle eastern | yes | no | family member | Yes |
| 14 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 18 | 0 | m | middle eastern | no | no | family member | No |
| 15 | 16 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 12 | 7 | m | black | no | no | family member | Yes |
| 16 | 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 36 | 0 | m | middle eastern | no | yes | family member | No |
| 17 | 18 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 12 | 8 | f | middle eastern | yes | no | family member | Yes |
| 18 | 19 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 29 | 3 | f | middle eastern | no | no | family member | No |
| 19 | 20 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 12 | 7 | f | black | no | no | family member | Yes |

Table 1: Details of variables mapping to the Q-Chat-10 screening methods

| **Variable in Dataset** | **Corresponding Q-chat-10-Toddler Features** |
|---|---|
| A1 | Does your child look at you when you call his/her name? |
| A2 | How easy is it for you to get eye contact with your child? |
| A3 | Does your child point to indicate that s/he wants something? (e.g. a toy that is out of reach) |
| A4 | Does your child point to share interest with you? (e.g. poin9ng at an interes9ng sight) |
| A5 | Does your child pretend? (e.g. care for dolls, talk on a toy phone) |
| A6 | Does your child follow where you're looking? |
| A7 | If you or someone else in the family is visibly upset, does your child show signs of wan9ng to comfort them? (e.g. stroking hair, hugging them) |
| A8 | Would you describe your child's first words as: |
| A9 | Does your child use simple gestures? (e.g. wave goodbye) |
| A10 | Does your child stare at nothing with no apparent purpose? |

# Data Preprocessing

## Handling Missing Values & Data Inspection

```
[ ] # 0. Initial Data Inspection and Missing Value Handling

    # Replace '?' with 'others' in 'relation' and 'ethnicity' columns if they exist
    for col in ['relation', 'ethnicity']:
        if col in df.columns:
            df[col] = df[col].replace('?', 'others')
```

## Encoding Categorical Features

```
[ ] # 1. Data Preprocessing

    # a. Convert categorical features to numerical
    for col in ['Sex', 'Jaundice', 'Family_mem_with_ASD', 'Class/ASD Traits ']: #add space at the end
        le = LabelEncoder()
        # Check if the column exists before transforming it
        if col in df.columns:  # Check if the column name is present
            df[col] = le.fit_transform(df[col])
        else:
            print(f"Warning: Column '{col}' not found in the DataFrame.") # This will print a warning if the column is not found
```

## Handling Class Imbalance with SMOTE (Synthetic Minority Over-sampling Technique)

```
# c. Handle Class Imbalance with SMOTE
smote = SMOTE(random_state=42)
X_train, y_train = smote.fit_resample(X_train, y_train)
```

## Feature Engineering: Q-Chat-10 Score Calculation

```
[ ] # 2. Feature Engineering: Q-Chat-10 Score Calculation

    def calculate_qchat_score(row):
        """Calculates the Q-Chat-10 score based on the specified rules."""
        score = 0
        for i in range(1, 10):  # A1 to A9
            if row[f'A{i}'] == 1:
                score += 1
        if row['A10'] == 1:
            score += 1
        return score

    df['Qchat_Score'] = df.apply(calculate_qchat_score, axis=1)

    # Create a binary ASD Traits variable based on the Qchat_Score
    df['ASD_Traits'] = df['Qchat_Score'].apply(lambda x: 1 if x > 3 else 0)
```

## Feature Selection Using P-values

```
[ ] # 3. Feature Selection Based on P-values

    def calculate_p_value(feature, target):
        """Calculates the p-value for a feature using the Chi-Square test."""
        contingency_table = pd.crosstab(feature, target)
        chi2, p, dof, expected = chi2_contingency(contingency_table)
        return p

    # Calculate p-values for 'used_app_before' and 'Sex' (if they exist)
    features_to_evaluate = ['used_app_before', 'Sex']  # Use 'Sex' instead of 'gender'

    p_values = {}
    for feature in features_to_evaluate:
        if feature in df.columns:
            p = calculate_p_value(df[feature], df['ASD_Traits'])
            p_values[feature] = p
            print(f"P-value for {feature}: {p}")
```

# Model Training

```python
[ ] # 7. Model Training and Evaluation
    models = {
        'GaussianNB': GaussianNB(),
        'LogisticRegression': LogisticRegression(random_state=42),
        'RandomForest': RandomForestClassifier(random_state=42),
        'GradientBoosting': GradientBoostingClassifier(random_state=42),
        'SVC': SVC(random_state=42, probability=True),  # Enable probability for ROC
        'KNeighbors': KNeighborsClassifier(),
        'MLP': MLPClassifier(random_state=42, max_iter=300)
    }
```

```python
[ ] # Function to train, evaluate, and visualize results
    def train_evaluate_model(model, name, X_train, y_train, X_test, y_test):
        print(f"Training and Evaluating {name}...")
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        y_prob = model.predict_proba(X_test)[:, 1]  # Probabilities for ROC Curve

        print(f"Classification Report for {name}:\n", classification_report(y_test, y_pred))
        print(f"Confusion Matrix for {name}:\n", confusion_matrix(y_test, y_pred))
```

## Best Model

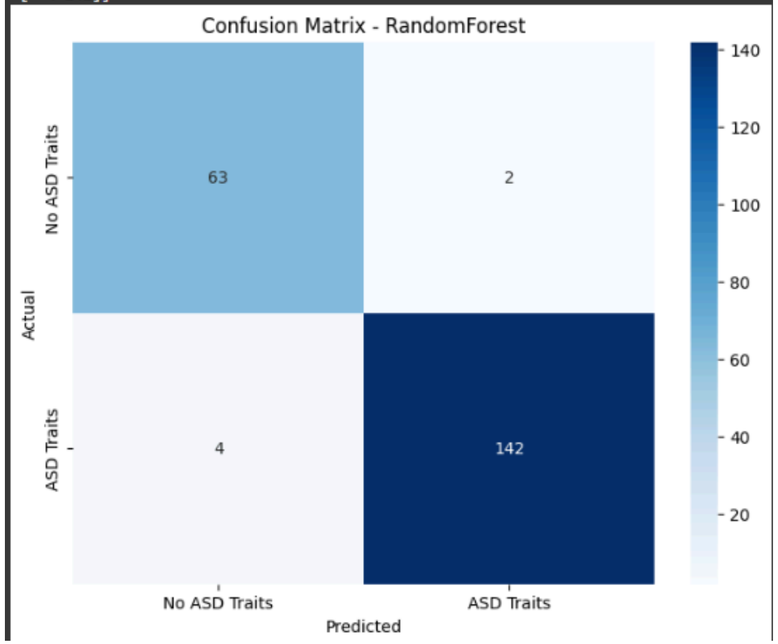
```python
# 7. Find the Best Model (Based on Test Accuracy)
best_model_name = None
best_accuracy = 0
best_model = None

for name, model in models.items():
    model.fit(X_train_selected, y_train)
    y_pred = model.predict(X_test_selected)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"{name} Accuracy: {accuracy}")

    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_model_name = name
        best_model = model

print(f"\nBest Model: {best_model_name} with Test Accuracy = {best_accuracy:.4f}")
```

```
Training and Evaluating RandomForest...
Classification Report for RandomForest:
               precision    recall  f1-score   support

           0       0.94      0.97      0.95        65
           1       0.99      0.97      0.98       146

    accuracy                           0.97       211
   macro avg       0.96      0.97      0.97       211
weighted avg       0.97      0.97      0.97       211

Confusion Matrix for RandomForest:
 [[ 63   2]
 [  4 142]]
```

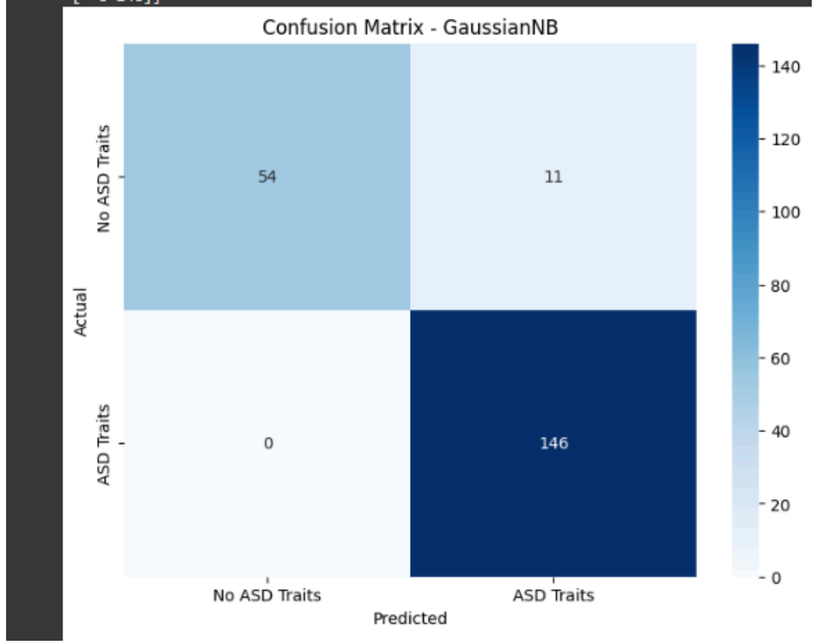
# Performance Evaluation

```
[ ] # Train and evaluate each model using selected features
    print("Evaluating models using selected features:")
    for name, model in models.items():
        train_evaluate_model(model, name, X_train_selected, y_train, X_test_selected, y_test)
```

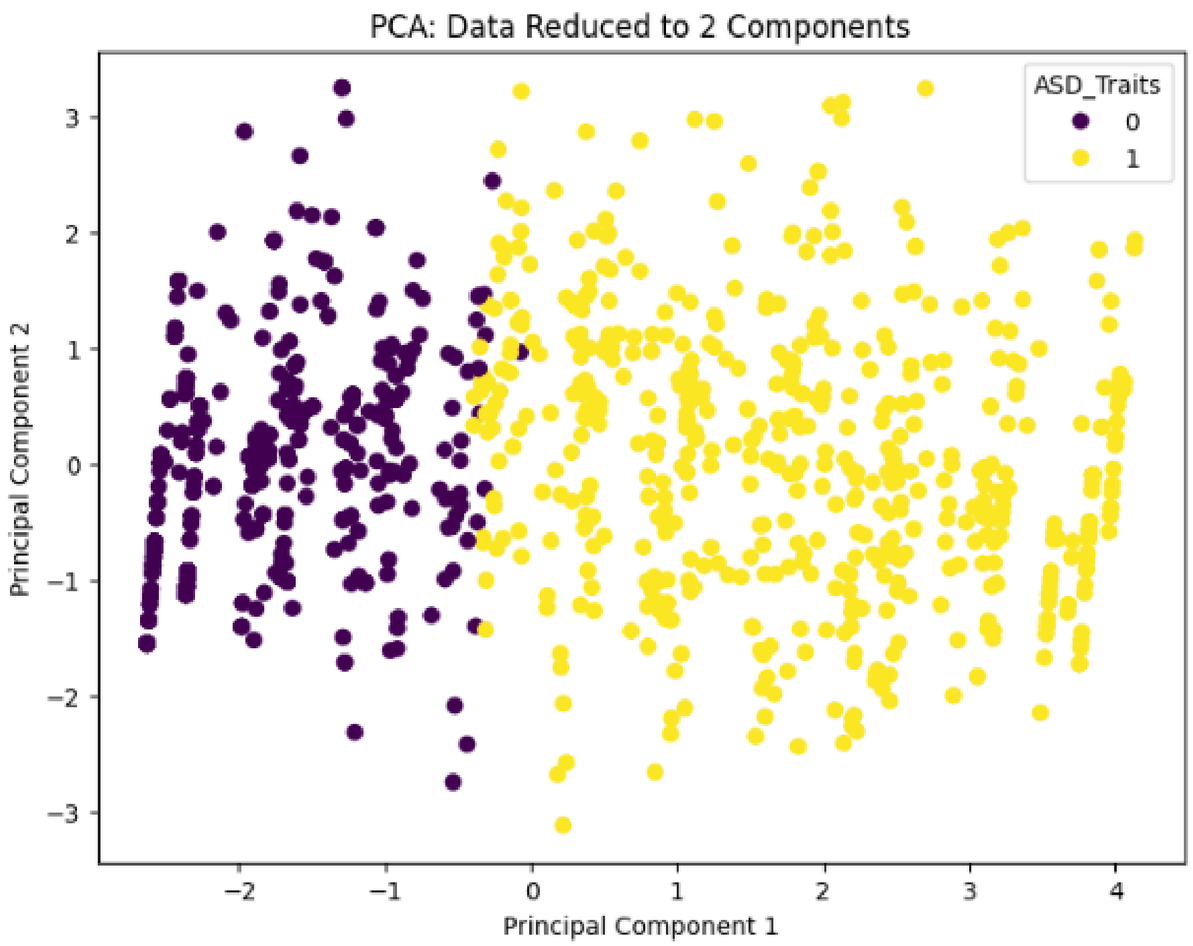
```
Training and Evaluating RandomForest...
Classification Report for RandomForest:
              precision    recall  f1-score   support

           0       0.94      0.97      0.95        65
           1       0.99      0.97      0.98       146

    accuracy                           0.97       211
   macro avg       0.96      0.97      0.97       211
weighted avg       0.97      0.97      0.97       211

Confusion Matrix for RandomForest:
 [[ 63   2]
 [  4 142]]
```




```
Evaluating models using selected features:
Training and Evaluating GaussianNB...
Classification Report for GaussianNB:
              precision    recall  f1-score   support

           0       1.00      0.83      0.91        65
           1       0.93      1.00      0.96       146

    accuracy                           0.95       211
   macro avg       0.96      0.92      0.94       211
weighted avg       0.95      0.95      0.95       211

Confusion Matrix for GaussianNB:
 [[ 54  11]
 [  0 146]]
```



# Performance Evaluation




This plot shows how PCA reduces data to two main features, making it easier to visualize patterns. The two colors represent ASD and non-ASD traits, showing clear separation for classification.

# Fusion Strategy

## Loading and Training Machine Learning Models: Image Classification & Random Forest

```python
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from tensorflow.keras.models import load_model
import joblib

# Step 1: Load Image Model
def load_image_model(model_path):
    try:
        model = load_model(model_path)
        print("Image classification model loaded successfully.")
        return model
    except Exception as e:
        print(f"Error loading image model: {e}")
        raise

# Step 2: Load or Train Random Forest Model
def load_rf_model(model_path, train_data=None, train_labels=None):
    try:
        model = joblib.load(model_path)
        if not hasattr(model, "predict_proba"):
            raise AttributeError("Loaded model does not support predict_proba.")
        print("Random Forest model loaded successfully.")
        return model
    except Exception as e:
        print(f"Error loading Random Forest model: {e}")
        if train_data is not None and train_labels is not None:
            print("Training a new Random Forest model...")
            model = RandomForestClassifier(random_state=42, n_jobs=-1)  # Using all available CPUs for training
            model.fit(train_data, train_labels)
            joblib.dump(model, model_path)
            print(f"New Random Forest model saved to {model_path}.")
            return model
        else:
            raise ValueError("Training data must be provided to create a new model.")
```

```python
# Step 3: Combine Predictions
def combine_models_predictions(image_model, rf_model, image_data, excel_data, weight1=0.8, weight2=0.2):
    # Get predictions from the image model
    image_preds = image_model.predict(image_data)
    image_probs = np.max(image_preds, axis=1)  # Maximum probabilities for each sample

    # Ensure excel_data is in the correct format for the Random Forest model
    if isinstance(excel_data, pd.DataFrame):
        excel_data = excel_data.to_numpy()

    # Get predictions from the Random Forest model
    rf_probs = rf_model.predict_proba(excel_data)[:, 1]  # Probabilities for the positive class

    # Combine probabilities using the weighted average
    combined_probs = weight1 * image_probs + weight2 * rf_probs
    combined_labels = (combined_probs > 0.5).astype(int)  # Threshold at 0.5 for binary classification

    return combined_labels, combined_probs

# Step 4: Main Script
if __name__ == "__main__":
    # Paths to models
    image_model_path = "/content/drive/MyDrive/model_file/image_classification_model.h5"
    rf_model_path = "/content/drive/MyDrive/model_file/random_forest_model_new_001111.pkl"

    # Load Image Model
    image_model = load_image_model(image_model_path)

    # Dummy data for Random Forest training (if needed)
    excel_data_train = np.random.rand(100, 5)  # Replace with your actual training data
    excel_labels_train = np.random.randint(0, 2, size=100)  # Binary labels

    # Load or Train Random Forest Model
    rf_model = load_rf_model(rf_model_path, train_data=excel_data_train, train_labels=excel_labels_train)

    # Dummy data for prediction
    image_data = np.random.rand(10, 224, 224, 3)  # Replace with preprocessed image data
    excel_data_test = pd.DataFrame(np.random.rand(10, 5))  # Replace with actual test data
    true_labels = np.random.randint(0, 2, size=10)  # Replace with true labels for evaluation

    # Combine Predictions
    final_labels, final_probs = combine_models_predictions(
        image_model, rf_model, image_data, excel_data_test, weight1=0.6, weight2=0.4
    )

    # Evaluate Combined Model
    print("\nClassification Report:")
    print(classification_report(true_labels, final_labels))

    # Optional: Print Predictions
    print("\nFinal Combined Probabilities:", final_probs)
    print("Final Combined Labels:", final_labels)
```

# Model Evaluation and Classification Report for Autism Detection

```
Image classification model loaded successfully.
Random Forest model loaded successfully.
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 885ms/step

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         2
           1       0.89      1.00      0.94         8

    accuracy                           0.90        10
   macro avg       0.94      0.75      0.80        10
weighted avg       0.91      0.90      0.89        10


Final Combined Probabilities: [0.72417166 0.56683133 0.6237322  0.74413697 0.73466976 0.75462638
 0.80142146 0.49375303 0.78116273 0.61686944]
Final Combined Labels: [1 1 1 1 1 1 1 0 1 1]
```

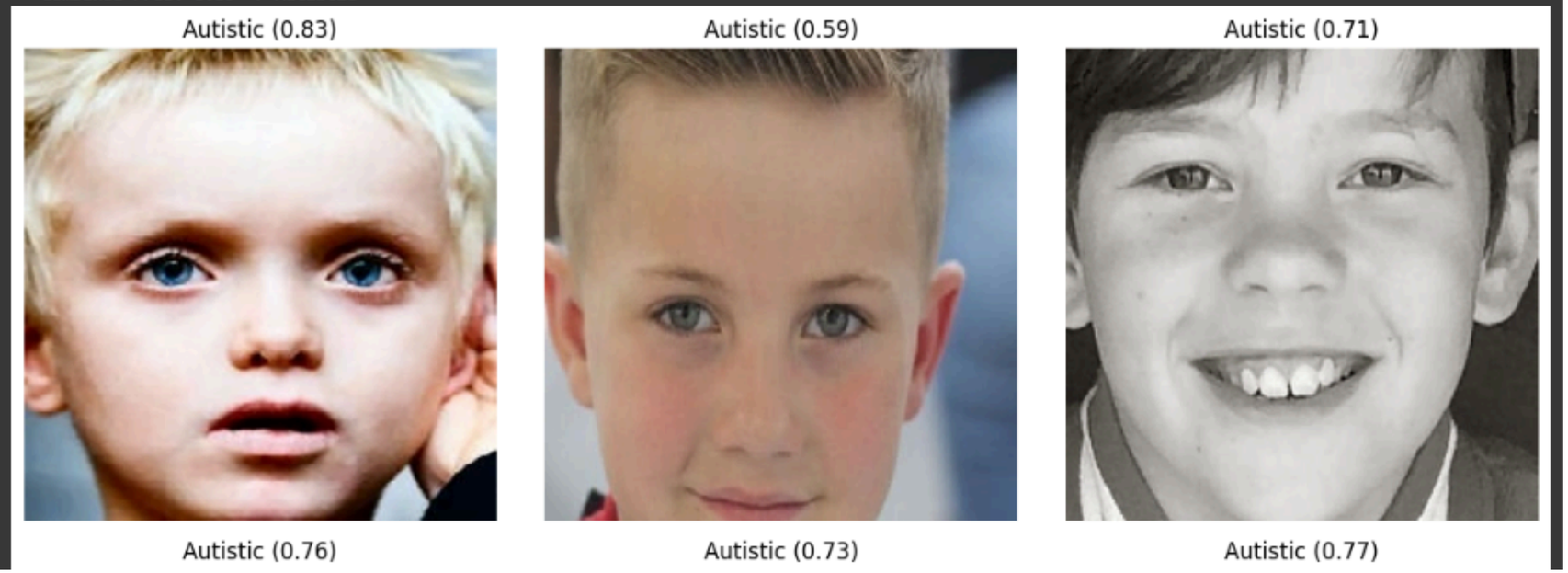
# Multimodal Autism Prediction: Image and Text Fusion Approach

```python
import numpy as np
import os
import random
import pandas as pd
from tensorflow.keras.preprocessing.image import load_img, img_to_array

# Define test image directory (Modify as needed)
test_images_dir = "/content/drive/MyDrive/autism_images_1212212/autism-image-data/AutismDataset/test"

# Get list of all test images
test_image_paths = [os.path.join(test_images_dir, img) for img in os.listdir(test_images_dir) if img.endswith(".jpg")]

# Check if images exist
if len(test_image_paths) == 0:
    raise ValueError("No test images found! Check the directory path.")

# Shuffle images to get a balanced mix
random.shuffle(test_image_paths)

# Select 9 random test images
num_samples = min(9, len(test_image_paths))
sample_indices = np.random.choice(len(test_image_paths), num_samples, replace=False)

# Visualization setup
plt.figure(figsize=(12, 12))

for i, idx in enumerate(sample_indices):
    img_path = test_image_paths[idx]

    # Load and preprocess image
    img = load_img(img_path, target_size=(224, 224))  # Resize to match model input
    img_array = img_to_array(img) / 255.0  # Normalize pixel values
    img_array = np.expand_dims(img_array, axis=0)  # Expand dimensions for model input

    # Get prediction from image model
    img_pred = image_model.predict(img_array)
    img_prob = np.max(img_pred)  # Take max probability

    # Select a random row from Excel test data
    random_excel_idx = np.random.randint(0, len(excel_data_test))
    rf_prob = rf_model.predict_proba([excel_data_test.iloc[random_excel_idx]])[:, 1][0]

    # Adjust weights for better fusion
    weight_image = 0.5
    weight_rf = 0.5

    # Weighted probability fusion
    combined_prob = (weight_image * img_prob) + (weight_rf * rf_prob)
```

```python
    img_array = np.expand_dims(img_array, axis=0)  # Expand dimensions for model input

    # Get prediction from image model
    img_pred = image_model.predict(img_array)
    img_prob = np.max(img_pred)  # Take max probability

    # Select a random row from Excel test data
    random_excel_idx = np.random.randint(0, len(excel_data_test))
    rf_prob = rf_model.predict_proba([excel_data_test.iloc[random_excel_idx]])[:, 1][0]

    # Adjust weights for better fusion
    weight_image = 0.5
    weight_rf = 0.5

    # Weighted probability fusion
    combined_prob = (weight_image * img_prob) + (weight_rf * rf_prob)

    # Classification threshold tuning
    threshold = 0.55  # Adjust to balance predictions
    predicted_label = "Autistic" if combined_prob > threshold else "Non-Autistic"

    # Debugging: Print predictions
    print(f"Image: {os.path.basename(img_path)}")
    print(f"   Image Model Prob: {img_prob:.2f}")
    print(f"   RF Model Prob: {rf_prob:.2f}")
    print(f"   Combined Prob: {combined_prob:.2f} -> {predicted_label}")

    # Display image with prediction
    plt.subplot(3, 3, i + 1)
    plt.imshow(img)
    plt.axis("off")
    plt.title(f"{predicted_label} ({combined_prob:.2f})")

plt.tight_layout()
plt.show()
```

# Autism Detection Using Multimodal Fusion: Image and Text-Based Predictions

```
Image: Autistic.51.jpg
   Image Model Prob: 0.72
   RF Model Prob: 0.82
   Combined Prob: 0.77 -> Autistic
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
Image: Autistic.18.jpg
   Image Model Prob: 0.84
   RF Model Prob: 0.76
   Combined Prob: 0.80 -> Autistic
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step
Image: Non_Autistic.131.jpg
   Image Model Prob: 0.86
   RF Model Prob: 0.63
   Combined Prob: 0.75 -> Autistic
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
Image: Non_Autistic.32.jpg
   Image Model Prob: 0.87
   RF Model Prob: 0.90
   Combined Prob: 0.89 -> Autistic
```



# Conclusion & Future Work

- Conclusion

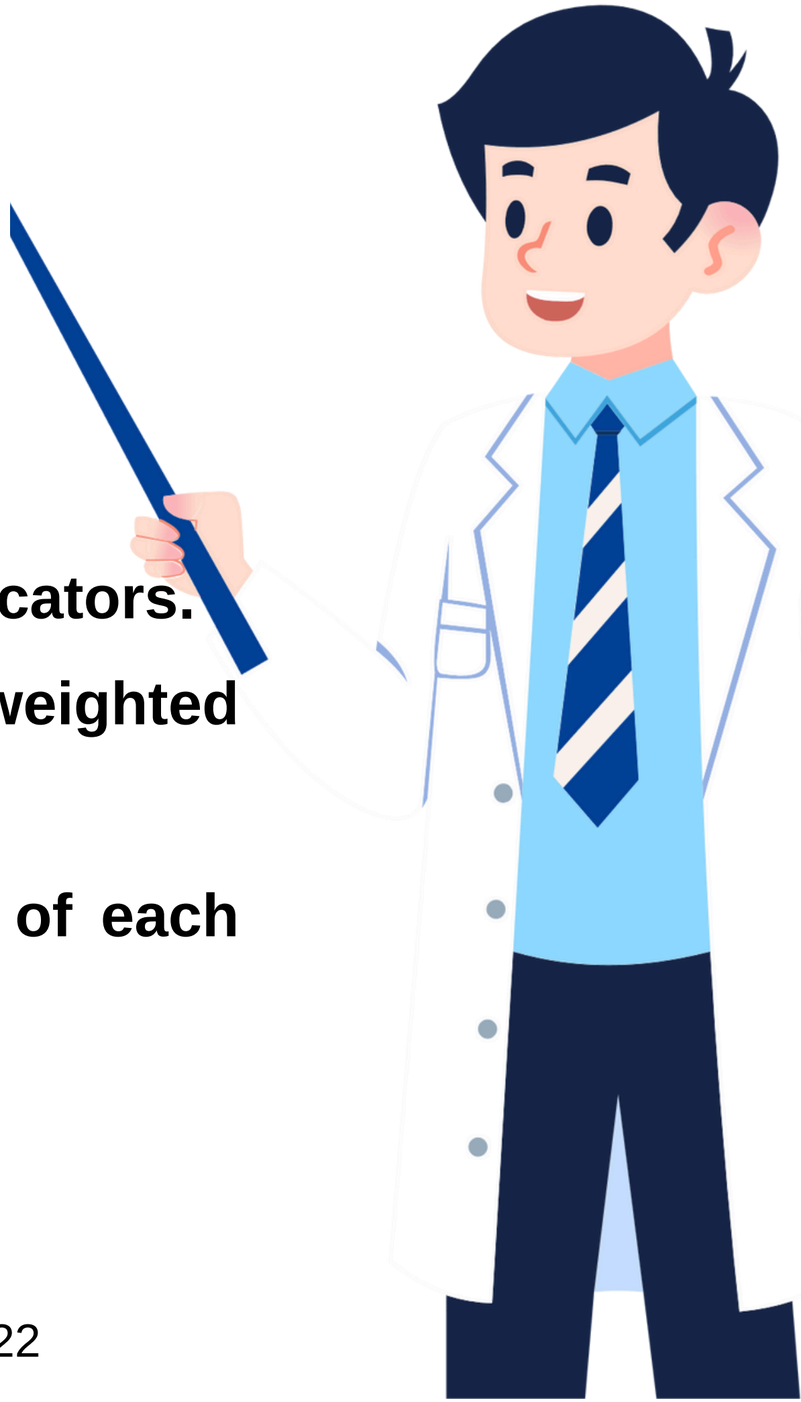
  - Successfully developed an ASD detection model using facial images & text data.
  - Fusion of both modalities improves accuracy compared to single-method approaches.
  - Can assist in early screening and reduce clinical workload.

- Future Work

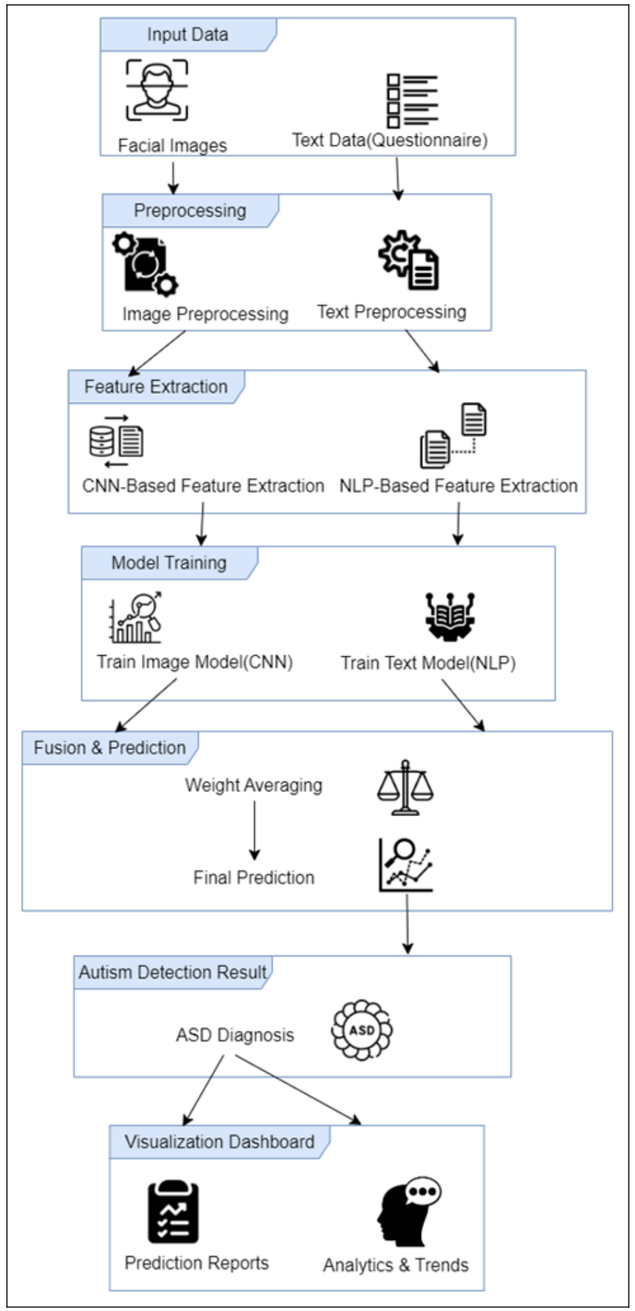
  - Expand Dataset – Collect more diverse facial images for better generalization.
  - Enhance Model Features – Integrate more behavioral and facial analysis techniques.
  - Real-Time Implementation – Develop a mobile or web-based screening tool.
  - Clinical Validation – Collaborate with healthcare professionals for real-world testing.

# Specific & Sub Objectives

- Detect ASD traits using facial image analysis.
- Analyze questionnaire responses to identify ASD indicators.
- Combine both methods for better accuracy using a weighted approach.
- Giving the visualization report for current situation of each patient
- Verify the prediction result use in clinical data



# Methodology



# Completion & What's left?

55%

| Completed | Remaining on PP2 |
|---|---|
| Data Gathering | Model Enhancement |
| Text & Image Preprocessing | Frontend Development |
| Text & Image Model Building | Backend Development |
| Frontend Development | Combining Model Use in Mutimodel Technique(Fusion) |
| Best Model Selection | Data Visualization Report |

# References

[1] Uddin, M. Z., Shahriar, M. A., Mahamood, M. N., Alnajjar, F., Pramanik, M. I., & Ahad, M. A. R. (2024). Deep learning with image-based autism spectrum disorder analysis: A systematic review. Engineering Applications of Artificial Intelligence, 127, 107185.

[2] Rasul, R. A., Saha, P., Bala, D., Karim, S. R. U., Abdullah, M. I., & Saha, B. (2024). An evaluation of machine learning approaches for early diagnosis of autism spectrum disorder. Healthcare Analytics, 5, 100293.

[3] *Rashid, A., & Shaker, S. (2023). Autism spectrum disorder detection using face features based on deep neural network. Wasit Journal of Computer and Mathematics Sciences, 2(1), 74-83.*

# Diagnosing and Analyzing Depression Through Text and Facial Images


## IT21178368 | Perera K.K.S

Bachelor of Science (Hons) in Information Technology
Specializing in Data Science

# Introduction & Background

## Component 4

### What is Depression?

- Depression is a common mental disorder.
- Globally, an estimated 5% of adults suffer from depression.
- It involves a depressed mood or loss of pleasure or interest in activities **for long periods of time.**
- Depression can lead to suicide.

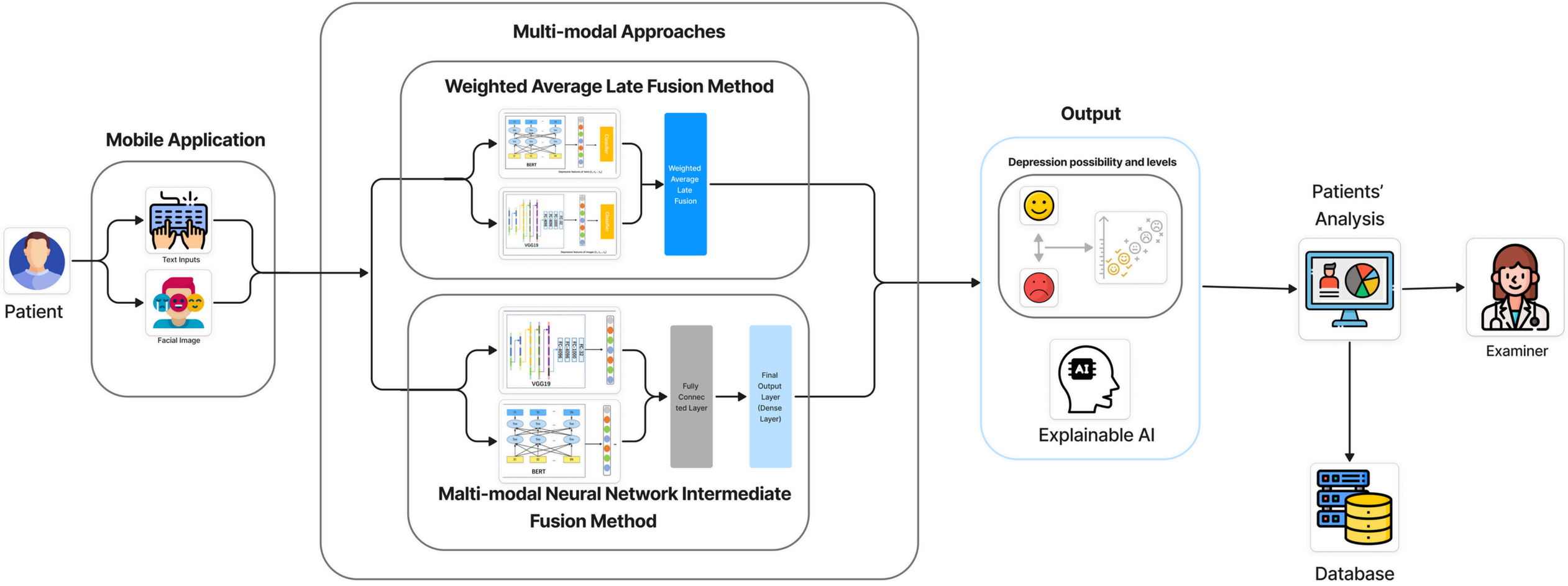
# Research Questions

- How general practitioners can easily diagnose mental disorder like Depression?
- How can practitioners monitor and examine depression patients' behaviors effectively?

# Specifics and Sub Objectives

- Initially determining whether a patient is at risk of having depression.
- Monitoring patients' behavior daily by collecting text inputs and facial images together.
- Give analysis/summary about patients’ current status to doctor.

# Methodology



# Completion & What's left?

55%

| Completed | Remaining on PP2 |
|---|---|
| Data Gathering | Frontend & Backend Completion |
| Text & Image data preprocessing | Model Enhancement & Depression Severity assessment |
| Text & Image processing model building & evaluation | Multimodal approach completion |
| Model Comparison | Use Explainable AI |
| Multimodal fusion method development | Depression Report Visualization |

# Evidence of Completion - Datasets

```python
# Load data
text_dataset_path = '/content/drive/MyDrive/Colab Notebooks/Research/depression_dataset.csv'
df = pd.read_csv(text_dataset_path)
```

```python
df.head()
```

| | text | is_depression |
|---|---|---|
| 0 | we understand that most people who reply immed... | 1 |
| 1 | welcome to r depression s check in post a plac... | 1 |
| 2 | anyone else instead of sleeping more when depr... | 1 |
| 3 | i ve kind of stuffed around a lot in my life d... | 1 |
| 4 | sleep is my greatest and most comforting escap... | 1 |

Next steps: Generate code with df | View recommended plots | New interactive sheet

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7731 entries, 0 to 7730
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   text           7731 non-null   object
 1   is_depression  7731 non-null   int64
dtypes: int64(1), object(1)
memory usage: 120.9+ KB
```

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Paths to train and test data
train_directory = './train'
test_directory = './test'

# Image size and batch size
IMG_SIZE = (224, 224)  # Resize to 224*224 for VGG19
BATCH_SIZE = 128

# Data Generators
train_datagen = ImageDataGenerator(rescale=1.0/255)
test_datagen = ImageDataGenerator(rescale=1.0/255)

# Load train and test data
train_data = train_datagen.flow_from_directory(
    train_directory,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

test_data = test_datagen.flow_from_directory(
    test_directory,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)
```

[6] ✓ 1.3s

```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
```

# Preprocessing

```python
# Parameters for text processing
max_words = 10000
max_len = 1000

# Preprocess text
text_column = 'text'
label_column = 'is_depression'

texts = df[text_column].str.lower().values
labels = df[label_column].values  # Labels already in binary format (0 and 1)

# Tokenize and pad sequences
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
padded_sequences = pad_sequences(sequences, maxlen=max_len, padding='post')

print(f"Text samples: {texts[:2]}")
print(f"Labels: {labels[:2]}")
print(f"Padded sequences shape: {padded_sequences.shape}")
```

```
Text samples: ['we understand that most people who reply immediately to an op with a
 'welcome to r depression s check in post a place to take a moment and share what is
Labels: [1 1]
Padded sequences shape: (7731, 1000)
```

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Paths to train and test data
train_directory = './train'
test_directory = './test'

# Image size and batch size
IMG_SIZE = (224, 224)  # Resize to 224*224 for VGG19
BATCH_SIZE = 128

# Data Generators
train_datagen = ImageDataGenerator(rescale=1.0/255)
test_datagen = ImageDataGenerator(rescale=1.0/255)

# Load train and test data
train_data = train_datagen.flow_from_directory(
    train_directory,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

test_data = test_datagen.flow_from_directory(
    test_directory,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)
```

[6] ✓ 1.3s

# Model Creation

```python
# Split data for feeding the model
X_train, X_test, y_train, y_test = train_test_split(
    padded_sequences, labels, test_size=0.2, random_state=42
)

print(f"Training data shape: {X_train.shape}")
print(f"Testing data shape: {X_test.shape}")
```

```
Training data shape: (6184, 1000)
Testing data shape: (1547, 1000)
```

```python
# Define LSTM Model
input_shape = X_train.shape[1]

lstm_model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=max_words, output_dim=128, input_length=input_shape),
    tf.keras.layers.LSTM(64, return_sequences=True),
    tf.keras.layers.LSTM(32),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile the LSTM model
lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy', 'AUC'])
```

```python
from tensorflow.keras.applications import VGG16, VGG19
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam

# Load VGG16 or VGG19
base_model = VGG16(input_shape=(48, 48, 3), include_top=False, weights='imagenet')

# Freeze the base layers
base_model.trainable = False

# Add custom layers for emotion classification
x = Flatten()(base_model.output)

# Add additional dense layers with BatchNorm and Dropout
x = Dense(256, activation='relu')(x)
x = BatchNormalization()(x)  # Normalize activations for faster convergence
x = Dropout(0.5)(x)           # Prevent overfitting

x = Dense(128, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.3)(x)

# Output layer for emotion classification (7 categories)
output = Dense(7, activation='softmax')(x)

# Define the final model
model = Model(inputs=base_model.input, outputs=output)

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

# Model Training

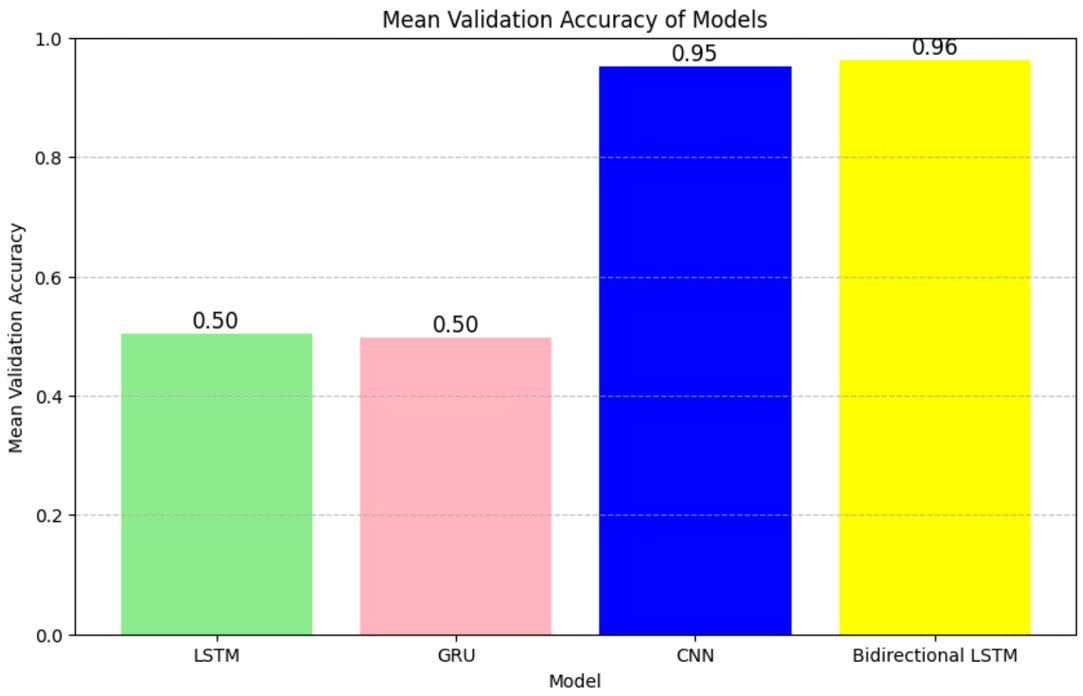
```python
# Train the LSTM model
history_lstm = lstm_model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=10,
    batch_size=32,
    verbose=1
)
```

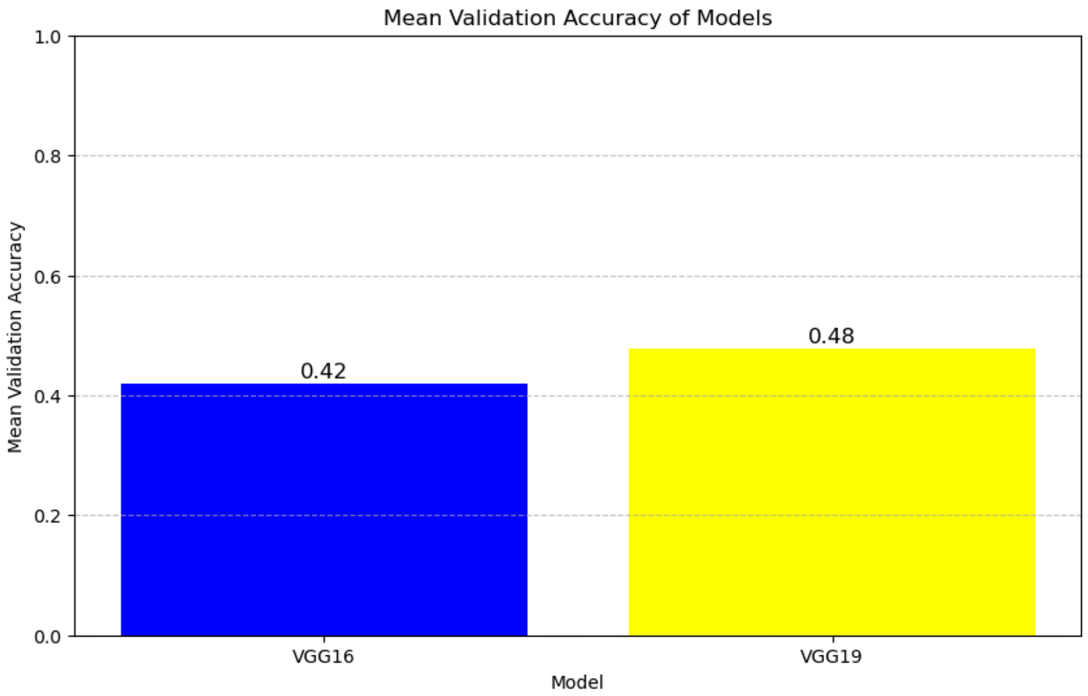
```python
# Compile, train, and evaluate the Bidirectional LSTM model
bi_lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy', 'AUC'])
history_bi_lstm = bi_lstm_model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=10,
    batch_size=32,
    verbose=1
)
```

```python
# Compile the models
model_vgg19.compile(optimizer=Adam(learning_rate=0.001),
                    loss='categorical_crossentropy',
                    metrics=['accuracy'])

# Train the model
history_vgg19 = model_vgg19.fit(train_data, validation_data=test_data, epochs=20)
```

✓ 2011m 26.3s

# Model Comparison



# Mutlimodal Fusion - Weighted Average Approach

```python
[ ] from tensorflow.keras.models import load_model
    from tensorflow.keras.preprocessing.sequence import pad_sequences
    from tensorflow.keras.preprocessing import image
    import numpy as np

    # Load Text and Image Models
    text_model = load_model("/content/drive/MyDrive/Colab Notebooks/Research/best_text_model.h5")  # LSTM text model
    image_model = load_model("/content/drive/MyDrive/Colab Notebooks/Research/temp_image_model.h5")  # VGG16 image model

    # Max sequence length used during text model training
    max_length = 4000

    # Define emotion class indices
    class_indices = {
        'angry': 0,
        'disgusted': 1,
        'fearful': 2,
        'happy': 3,
        'neutral': 4,
        'sad': 5,
        'surprised': 6
    }

    # Predict Depression from Text
    def predict_text(text):
        # Preprocess and tokenize the text
        sequence = tokenizer.texts_to_sequences([text])
        padded_sequence = pad_sequences(sequence, maxlen=max_length, padding='post')

        # Get model predictions
        probability = text_model.predict(padded_sequence)[0][0]
        return probability  # Probability of depression
```

```python
# Predict Depression from Text
def predict_text(text):
    # Preprocess and tokenize the text
    sequence = tokenizer.texts_to_sequences([text])
    padded_sequence = pad_sequences(sequence, maxlen=max_length, padding='post')

    # Get model predictions
    probability = text_model.predict(padded_sequence)[0][0]
    return probability  # Probability of depression

# Example Text Prediction
P_text = predict_text("I feel very sad and hopeless.")
print("Prob of text: ", P_text)

# Predict Depression from Image
def predict_image(img_path):
    # Load and preprocess the image
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    # Get emotion probabilities
    emotion_probs = image_model.predict(img_array)[0]  # Shape: (num_classes,)
    print("Probabilities of emotions: ", emotion_probs)

    # Depression-related emotions
    depressed_emotions = ['angry', 'fearful', 'sad']
    depression_score = emotion_probs[5] + emotion_probs[0] + emotion_probs[2]
    return depression_score

# Example Image Prediction
P_image = predict_image("/content/drive/MyDrive/Colab Notebooks/Research/emotions/test/angry/im957.png")
```

```python
# Example Image Prediction
P_image = predict_image("/content/drive/MyDrive/Colab Notebooks/Research/emotions/test/angry/im957.png")
print("Prob of image: ", P_image)

# Multimodal Fusion: Weighted Average
def weighted_average(P_text, P_image, w_text=0.5, w_image=0.5):
    return w_text * P_text + w_image * P_image

# Final Prediction
P_final = weighted_average(P_text, P_image)
print("Final prob using weighted avg: ", P_final)

if P_final > 0.5:
    print("Depressed")
else:
    print("Not Depressed")
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_met
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_met
WARNING:absl:Error in loading the saved optimizer state. As a result, your model is starting with a fresh
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 660ms/step
Prob of text:  0.10733011
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 836ms/step
Probabilities of emotions:  [0.03356706 0.00650709 0.5682599  0.01025432 0.28322008 0.08598385
 0.01220761]
Prob of image:  0.6878108
Final prob using weighted avg:  0.3975704461336136
Not Depressed
```

# References


[1] *K. Shah, U. Patel and Y. Kumar, "Machine Learning-Based Approaches for Early Prediction of Depression," 2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), Bangalore, India, 2024, pp. 1-7, doi: 10.1109/IITCEE59897.2024.10467234. keywords: {Computational modeling;Medical treatment;Mental health;Forestry;Predictive models;Depression;Risk management;depression detection;machine learning;random forest;ensemble method},*

[2] *A. Agrawal, S. Dey and G. C. Jana, "Depressive and Non-depressive Tweets Classification using a Sequential Deep Learning Model," 2023 International Conference on Intelligent Systems, Advanced Computing and Communication (ISACC), Silchar, India, 2023, pp. 1-6, doi: 10.1109/ISACC56298.2023.10083981. keywords: {Deep learning;Social networking (online);Computational modeling;Mental health;Media;Depression;Intelligent systems;Depression Detection;Sequential Deep Learning Model;ID-Convolution;LSTM;Social Media},*

# References

[3] *M. k. Jha, R. Ranjan, G. K. Dixit and K. Kumar, "An Efficient Machine Learning Classification with Feature Selection Techniques for Depression Detection from Social Media," 2023 International Conference on Communication, Security and Artificial Intelligence (ICCSAI), Greater Noida, India, 2023, pp. 481-486, doi: 10.1109/ICCSAI59793.2023.10421064. keywords: {Sensitivity;Social networking (online);Machine learning;Mental health;Depression;Feature extraction;Web sites;Depression Detection;Random Forest;Machine Learning;Gredient Boosting;Recursive Feature Elimination;ADASYN;Linear Regression},*

# Business Plan of MENTCARE AI

**April 2024**
Project Initiation

**August 2024**
Finalize Architecture

**September 2024**
Model Development and Implementation

**October 2024**
Begin development of platform

**Q1 2025**
Complete development of platform and have a working prototype

**Q2 2025**
Release Mentcure to the market

**Q2 2025**
Begin marketing campaigns and outreach programs for user acquisition

**Q3 2025**
Expand the functionality of the platform and focus on new features

SLIIT FACULTY OF COMPUTING

**Group ID: 24-25J-322**

# Commercialization

## Target Market

General Practitioners of the MOH centers

Institutes who learn about mental disorders

General population who wish to take self mental care

## Market Size

~12.5% of the world population

## Market Access Points

MOH Center

Mentcare Web

# Q & A Session

# Thank you for your attention