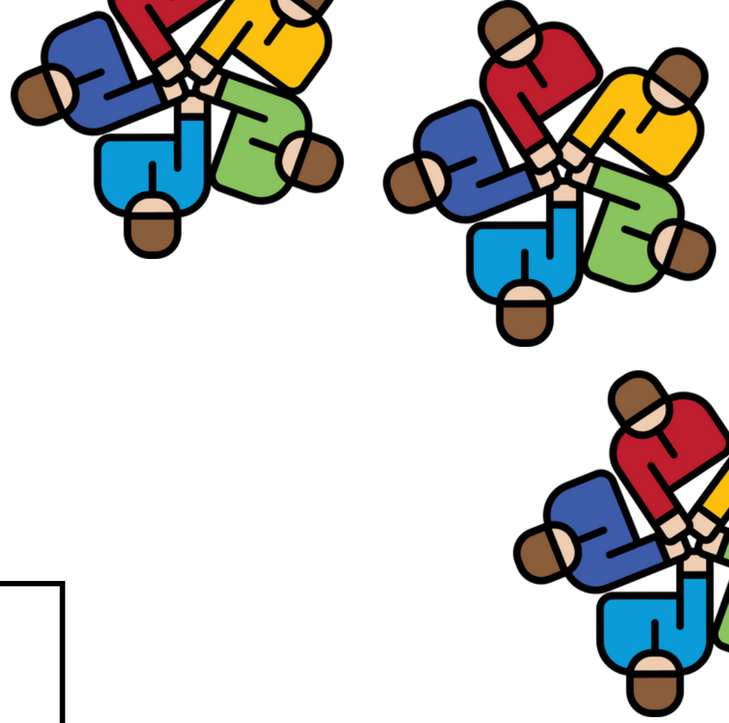
Enhancing the Efficiency of Diagnosing and Managing Mental Disorders using Machine Learning

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

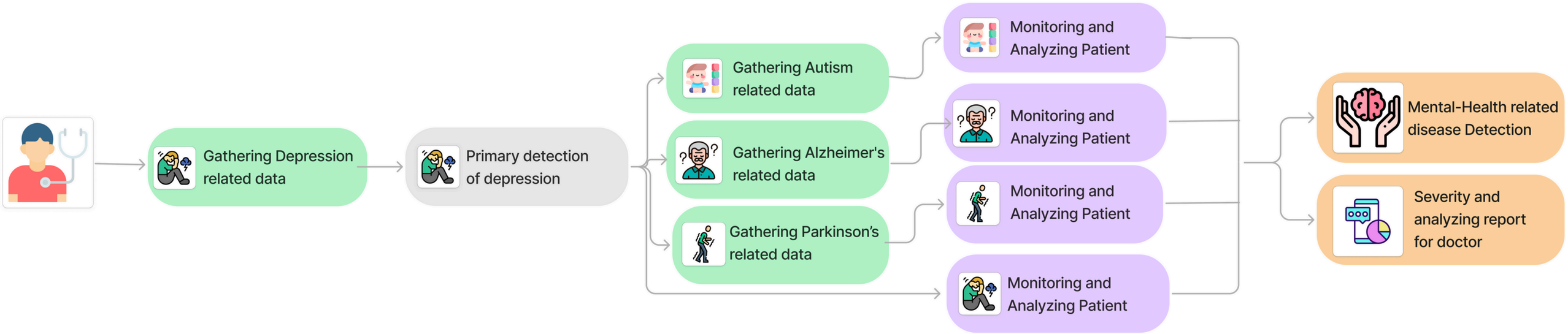
Diagnosing and Analyzing Alzheimer’s Risk through Machine Learning Techniques

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.

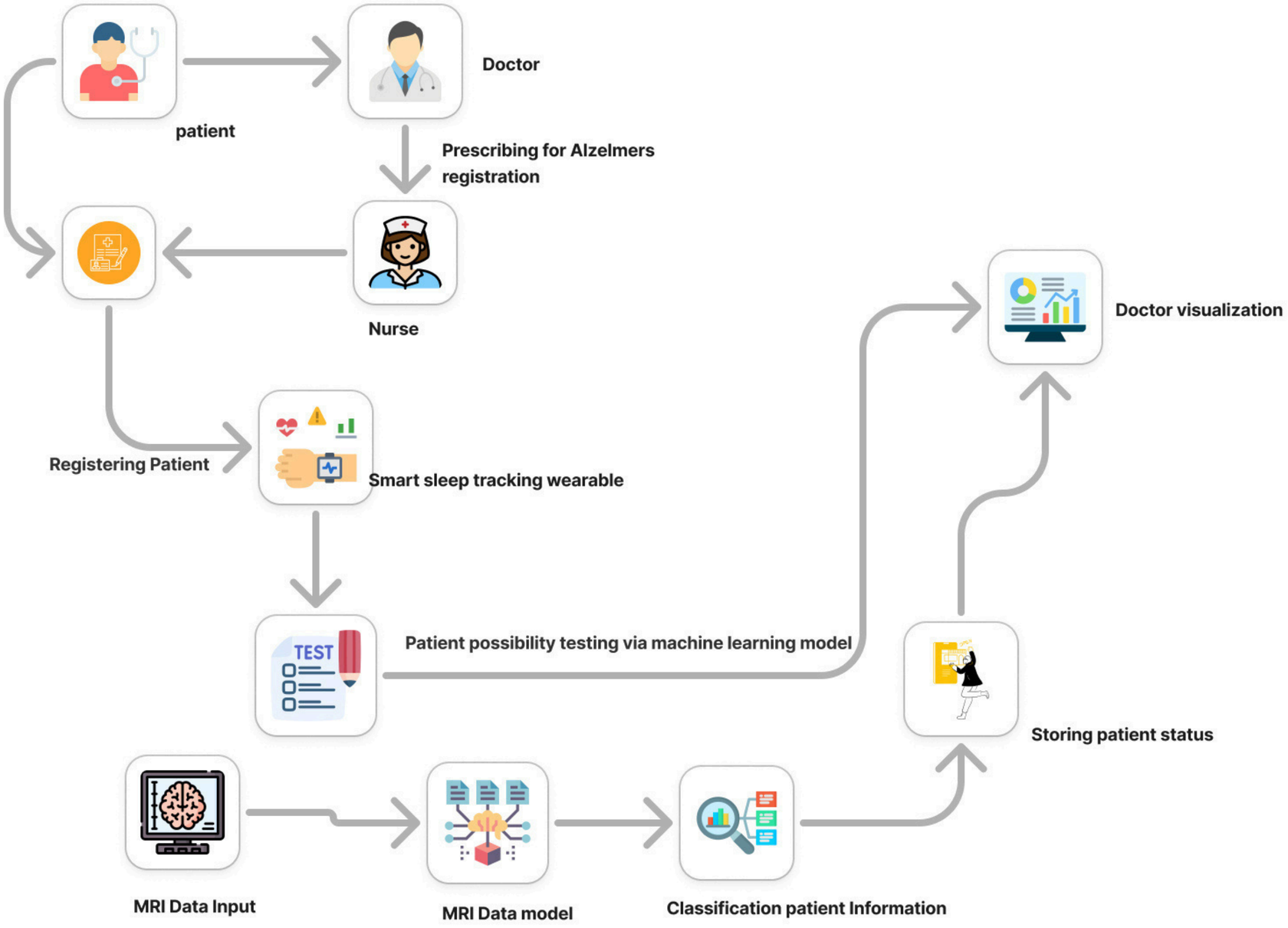
Research Question

- 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

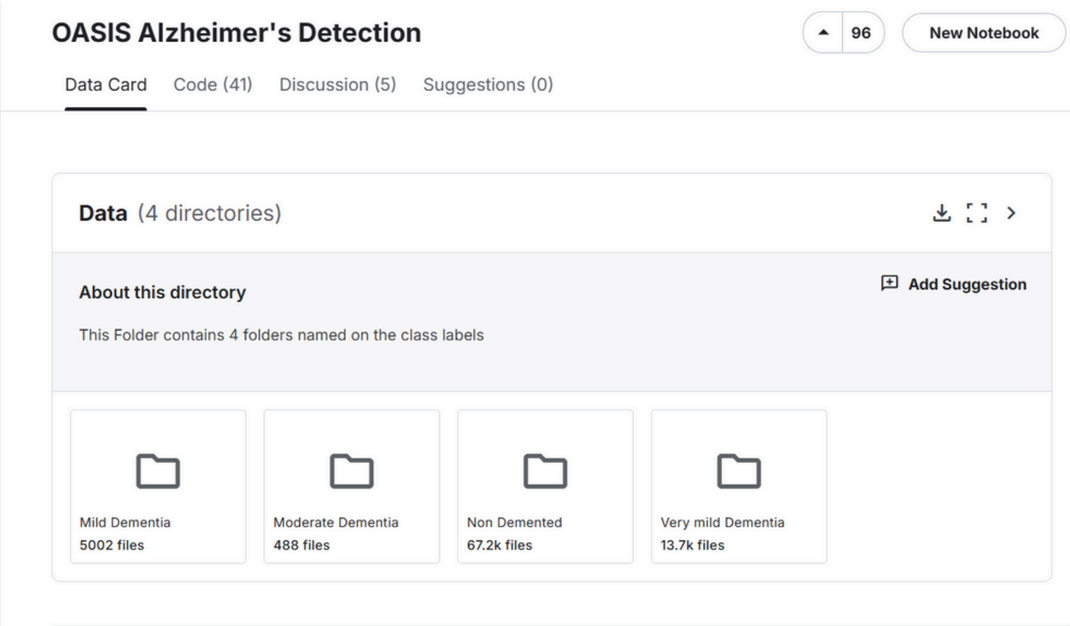


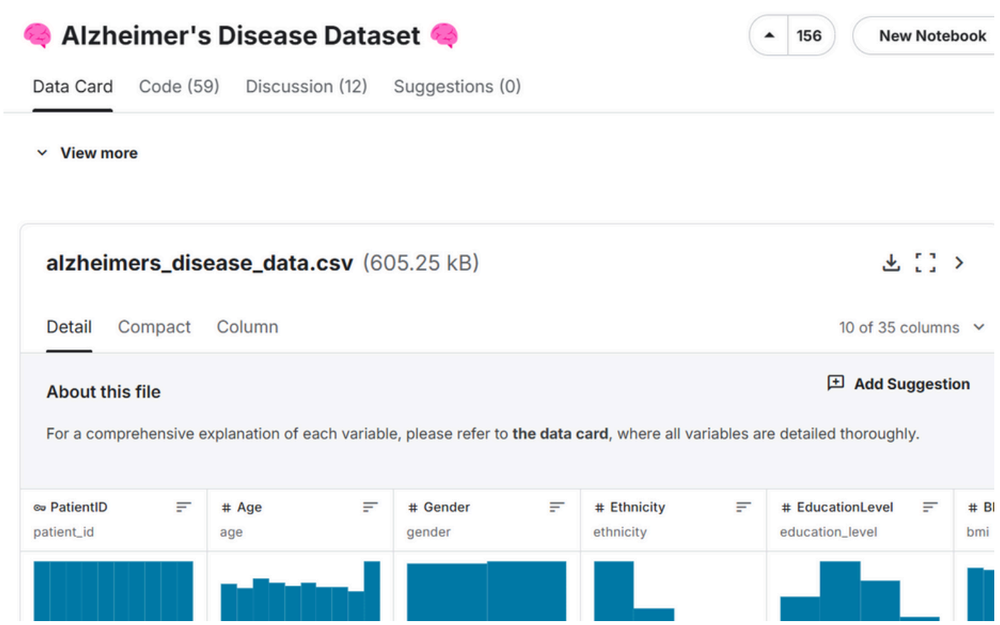
Completion & What's left?

90%

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

Data gathering



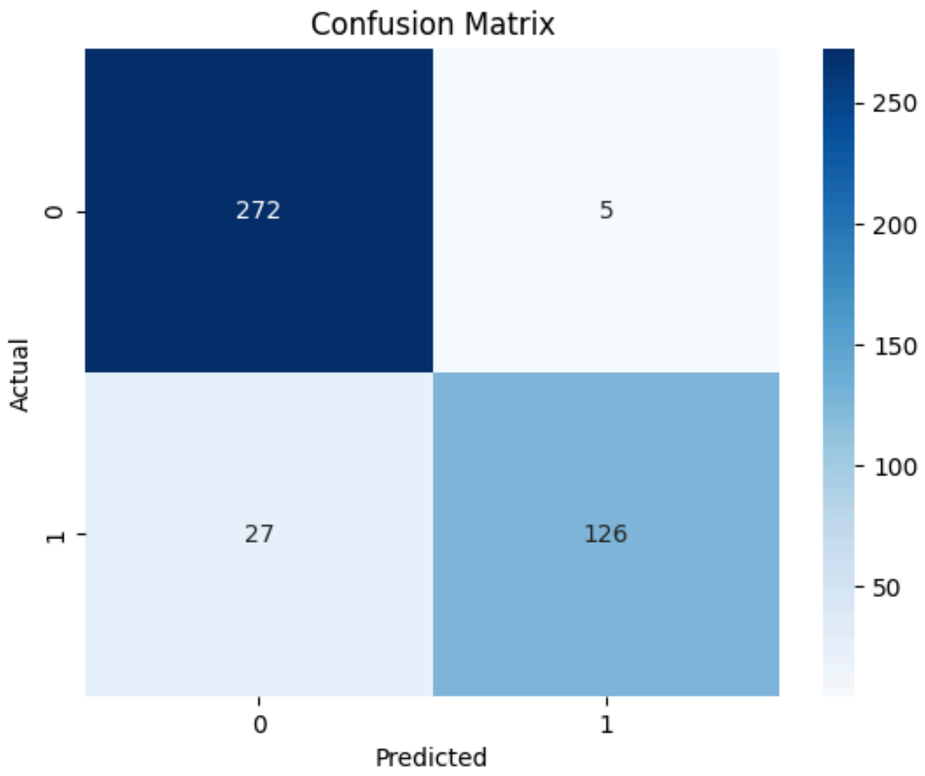


Data Preprocessing

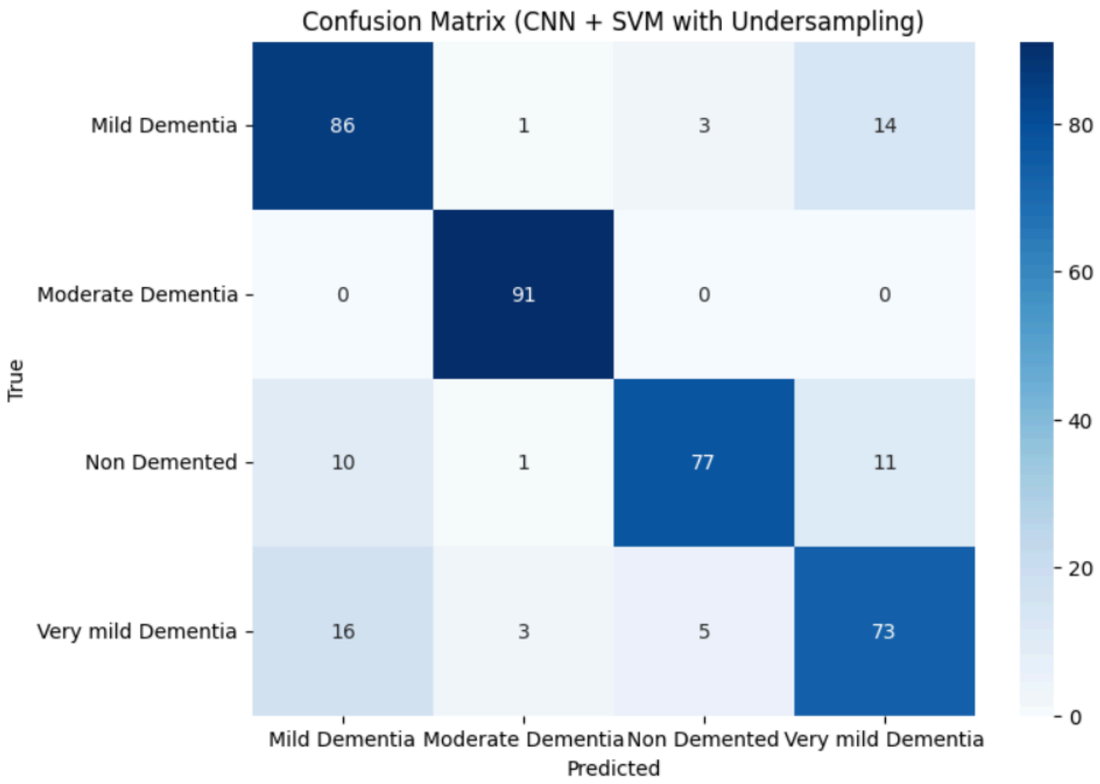
```python
# Check for missing values
print("\nMissing Values:")
print(data.isnull().sum())

# 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
```

Modal validation

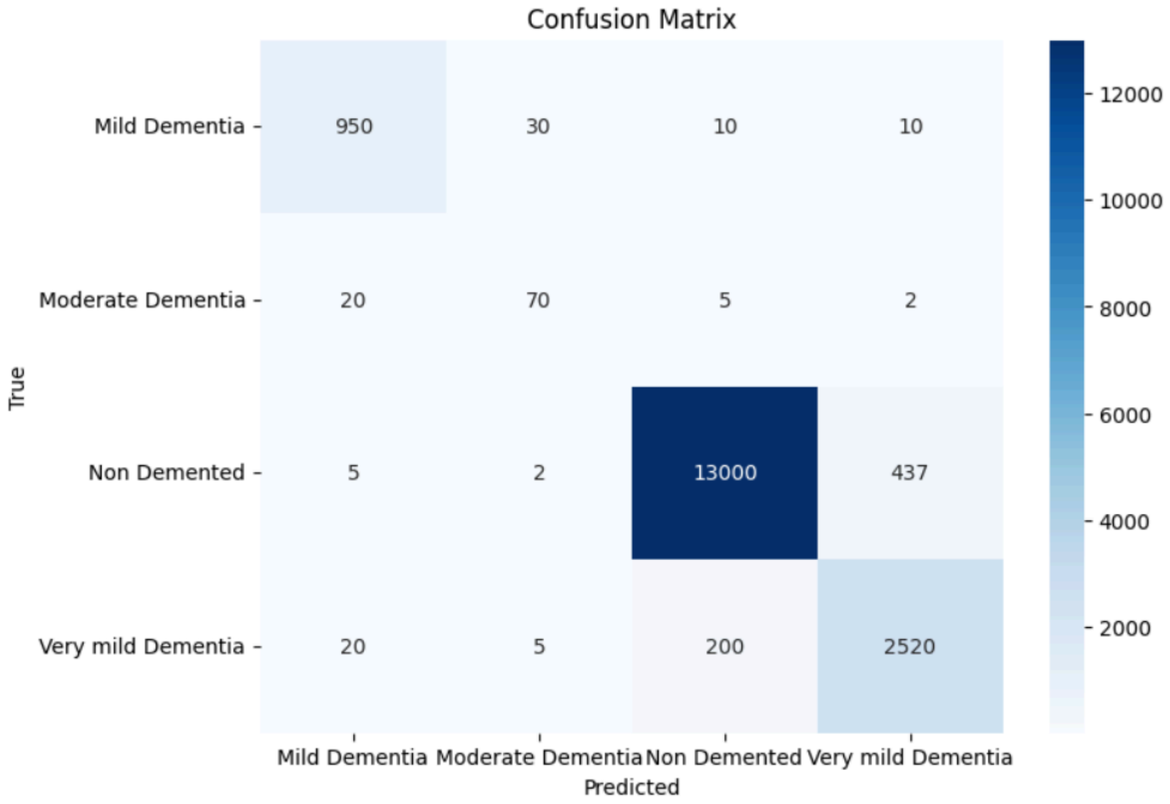


Data CNN + SVM model summary




```
...
    Classification Report:

                       precision    recall  f1-score   support

         Mild Dementia       0.77      0.83      0.80       104
     Moderate Dementia       0.95      1.00      0.97        91
          Non Demented       0.91      0.78      0.84        99
    Very mild Dementia       0.74      0.75      0.75        97

              accuracy                           0.84       391
             macro avg       0.84      0.84      0.84       391
          weighted avg       0.84      0.84      0.84       391


...  ['feature_scaler.pkl']
```

EfficientNetB0 Modal summary



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

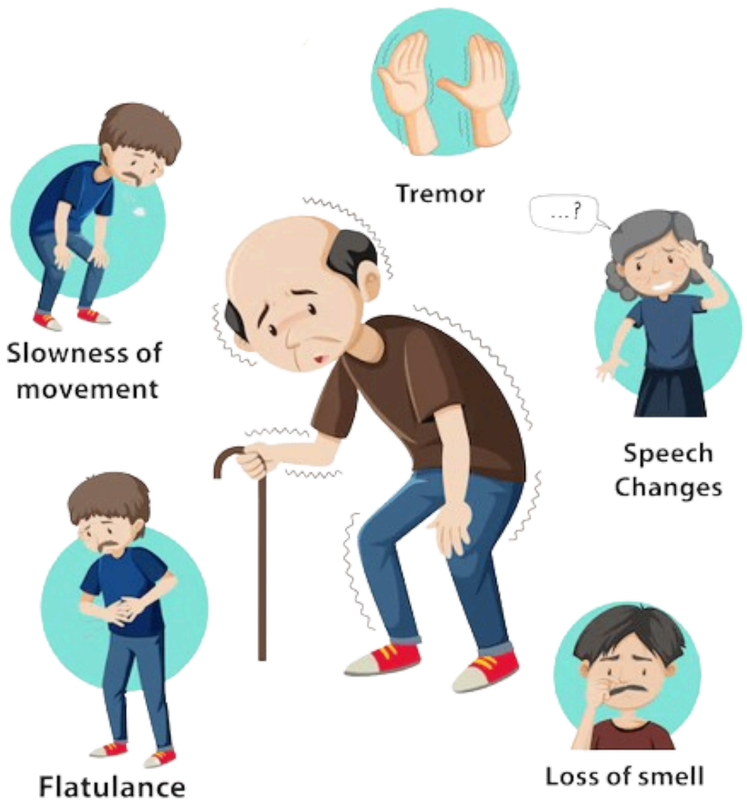
Bachelor of Science (Hons) in Information Technology
Specializing in Data Science

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

This dataset contains clinical records of Parkinson's patients, including:

- Demographic details (Age, Gender, etc.)
- Clinical test results (UPDRS score, Tremor severity, MoCA scores, etc.)
- Lifestyle & medical history factors

-2,105 patient records and 35 features

-Reads the dataset from a CSV file and displays the first 5 rows

-Checking missing values

```
df.shape
```

```
(2105, 35)
```

```
[ ] #load the dataset
    df = pd.read_csv('parkinsons_disease_data.csv')
    df.head()
```

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	PhysicalActivity	DietQuality	...	FunctionalA
0	3058	85	0	3	1	19.619878	0	5.108241	1.380660	3.893969	...	
1	3059	75	0	0	2	16.247339	1	6.027648	8.409804	8.513428	...	
2	3060	70	1	0	0	15.368239	0	2.242135	0.213275	6.498805	...	
3	3061	52	0	0	0	15.454557	0	5.997788	1.375045	6.715033	...	
4	3062	87	0	0	1	18.616042	0	9.775243	1.188607	4.657572	...	

5 rows × 35 columns

```
df.isnull().sum()
```

	0
PatientID	0
Age	0
Gender	0
Ethnicity	0
EducationLevel	0
BMI	0
Smoking	0
AlcoholConsumption	0

Data Preprocessing

```
[ ] #Label Encoding the Target Variable

    from sklearn.preprocessing import LabelEncoder


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

```
[ ] df['Diagnosis'].head()
```

	Diagnosis
0	0
1	1
2	1
3	1
4	0

dtype: int64

Encode categorical labels into numerical format

-Used LabelEncoder()

```
#Drop Unrelated Columns
df.drop(['PatientID','DoctorInCharge'],axis=1,inplace=True)
df.head()
```

	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	PhysicalActivity	DietQuality	SleepQuality	...	MoCA	FunctionalAssessment	Tr
0	85	0	3	1	19.619878	0	5.108241	1.380660	3.893969	9.283194	...	29.181289	1.572427	
1	75	0	0	2	16.247339	1	6.027648	8.409804	8.513428	5.602470	...	12.332639	4.787551	
2	70	1	0	0	15.368239	0	2.242135	0.213275	6.498805	9.929824	...	29.927783	2.130686	
3	52	0	0	0	15.454557	0	5.997788	1.375045	6.715033	4.196189	...	21.304268	3.391288	
4	87	0	0	1	18.616042	0	9.775243	1.188607	4.657572	9.363925	...	8.336364	3.200969	

5 rows × 33 columns

Removing unnecessary columns that do not contribute to the model's prediction.

Data Preprocessing

Feature Selection

-Using the Chi-Square (χ^2) statistical test

```python
#Feature selection using Chi-Square
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']
```

Spliit the dataset

-Splits the dataset into training and testing sets before model training

```python
#Train-Test Split (80%-20%)
from sklearn.model_selection import train_test_split

X = df[['Age' , 'AlcoholConsumption' ,'CholesterolTriglycerides', 'UPDRS' ,'MoCA',
 'FunctionalAssessment', 'Tremor' ,'Rigidity', 'Bradykinesia',
 'PosturalInstability']]
y = df['Diagnosis']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
y_train
```

	Diagnosis
874	0
650	0
1596	1
1578	1
1560	0
...	...
1638	1
1095	0
1130	1
1294	0
860	1

Model Training

```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]
    }
}
```

Defines hyperparameter tuning grids for three classifiers:

- Decision Tree Classifier
- Random Forest Classifier
- XGBoost Classifier

Model Training

```python
[ ]
grid_search.fit(X_train, y_train)

# Get the best model and evaluate
best_model = grid_search.best_estimator_
best_models[model_name] = best_model

# Make predictions
train_pred = best_model.predict(X_train)
test_pred = best_model.predict(X_test)

# Calculate metrics
train_accuracy = accuracy_score(y_train, train_pred)
test_accuracy = accuracy_score(y_test, test_pred)

results[model_name] = {
    "Best Params": grid_search.best_params_,
    "Train Accuracy": train_accuracy,
    "Test Accuracy": test_accuracy,
    "Classification Report": classification_report(y_test, test_pred, output_dict=True)
}

print(f"{model_name} - Train Accuracy: {train_accuracy:.4f}, Test Accuracy: {test_accuracy:.4f}")
print(f"Best Parameters: {grid_search.best_params_}")
```

```
Optimizing Decision Tree...
Fitting 3 folds for each of 72 candidates, totalling 216 fits
Decision Tree - Train Accuracy: 0.9181, Test Accuracy: 0.8670
Best Parameters: {'criterion': 'entropy', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 10}
Optimizing Random Forest...
Fitting 3 folds for each of 81 candidates, totalling 243 fits
Random Forest - Train Accuracy: 0.9792, Test Accuracy: 0.9145
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 300}
Optimizing XGBoost...
Fitting 3 folds for each of 243 candidates, totalling 729 fits
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [09:12:36] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
XGBoost - Train Accuracy: 0.9572, Test Accuracy: 0.9145
Best Parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100, 'subsample': 1.0}
```

Performs model training and hyperparameter tuning using Grid Search CV for three models:

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

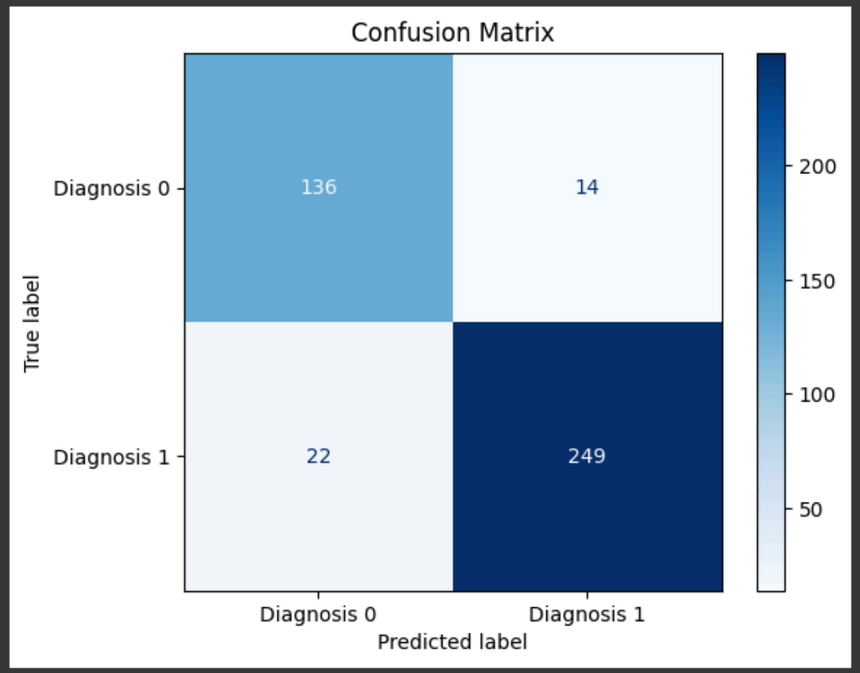
```python
[ ] # Find the best model based on Test Accuracy
best_model_name = max(results, key=lambda x: results[x]["Test Accuracy"])
best_model = best_models[best_model_name]

print(f"\nBest Model: {best_model_name} with Test Accuracy: {results[best_model_name]['Test Accuracy']:.4f}")
print(f"Best Hyperparameters for {best_model_name}: {results[best_model_name]['Best Params']}")
```

```
Best Model: Random Forest with Test Accuracy: 0.9145
Best Hyperparameters for Random Forest: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 300}
```

Random Forest outperformed the others, achieving a test accuracy of **91.45%**

Performance Evaluation

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

# Assuming best_model is your trained model
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()
```



Confusion Matrix

True Positives (TP) = 249

True Negatives (TN) = 136

False Positives (FP) = 14

False Negatives (FN) = 22

Key Performance Metrics

Precision - 94.68%

Recall - 91.89%

F1-score - Around 93%

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

Overview of the Dataset

There are 2 types of category files:

Helathy vs. Parkinson's

-Ensures the dataset is correctly structured before feature extraction.

-Helps detect missing or unbalanced data early.

```python
import os
import pandas as pd

# Define the dataset path
dataset_path = "/content/drive/MyDrive/speech_dataset/Audio"

# Define paths for Healthy and Parkinson's audio folders
healthy_audio_path = os.path.join(dataset_path, "HC_AH - Healthy/HC_AH")
parkinson_audio_path = os.path.join(dataset_path, "PD_AH - With Parkinson/PD_AH")

# Get list of files in each folder
healthy_files = os.listdir(healthy_audio_path)
parkinson_files = os.listdir(parkinson_audio_path)

# Create a DataFrame for overview
df_overview = pd.DataFrame({
    "Category": ["Healthy", "Parkinson's"],
    "Total Files": [len(healthy_files), len(parkinson_files)],
    "Sample Files": [healthy_files[:3], parkinson_files[:3]]  # Display first 3 sample files
})

# Print dataset summary
print("\n=== Dataset Overview ===")
print(df_overview)

from IPython.display import display
display(df_overview)
```

```
=== Dataset Overview ===
      Category  Total Files                                       Sample Files
0      Healthy           41  [AH_114S_A89F3548-0B61-4770-B800-2E26AB3908B6....
1  Parkinson's           40  [AH_545622718-C052AD58-5E6B-4ADC-855C-F76B66BA...
```

	Category	Total Files	Sample Files
0	Healthy	41	[AH_114S_A89F3548-0B61-4770-B800-2E26AB3908B6....
1	Parkinson's	40	[AH_545622718-C052AD58-5E6B-4ADC-855C-F76B66BA...

```python
# Display in table format (for Colab)
from IPython.display import display
display(df_files)
```

```
=== List of All Audio Files ===
                                                Filename     Category
0    AH_114S_A89F3548-0B61-4770-B800-2E26AB3908B6.wav      Healthy
1    AH_064F_7AB034C9-72E4-438B-A9B3-AD7FDA1596C5.wav      Healthy
2    AH_264Z_593C20CD-0A54-4177-B031-26EE147080A3.wav      Healthy
3    AH_195B_39DA6A45-F4CC-492A-80D4-FB79049ACC22.wav      Healthy
4    AH_325J_7F5F27AA-5A93-43CF-AB17-FC53940BF4B0.wav      Healthy
..                                                ...          ...
76  AH_545812846-0C14B32A-6C50-4B62-BC89-0A815C2DE...  Parkinson's
77  AH_545841227-5C77713A-66F1-49D0-BC8A-702C152E6...  Parkinson's
78  AH_545789682-7554E0C7-4E25-49C3-9E6C-04D525455...  Parkinson's
79  AH_545806325-8A17002B-CFD3-4DCF-8854-04F0F2BFF...  Parkinson's
80  AH_545834603-857E007F-1CCF-4249-8160-3A0F3F5AB...  Parkinson's

[81 rows x 2 columns]
```

	Filename	Category
0	AH_114S_A89F3548-0B61-4770-B800-2E26AB3908B6.wav	Healthy
1	AH_064F_7AB034C9-72E4-438B-A9B3-AD7FDA1596C5.wav	Healthy
2	AH_264Z_593C20CD-0A54-4177-B031-26EE147080A3.wav	Healthy
3	AH_195B_39DA6A45-F4CC-492A-80D4-FB79049ACC22.wav	Healthy
4	AH_325J_7F5F27AA-5A93-43CF-AB17-FC53940BF4B0.wav	Healthy
...	...	...
76	AH_545812846-0C14B32A-6C50-4B62-BC89-0A815C2DE...	Parkinson's
77	AH_545841227-5C77713A-66F1-49D0-BC8A-702C152E6...	Parkinson's
78	AH_545789682-7554E0C7-4E25-49C3-9E6C-04D525455...	Parkinson's
79	AH_545806325-8A17002B-CFD3-4DCF-8854-04F0F2BFF...	Parkinson's
80	AH_545834603-857E007F-1CCF-4249-8160-3A0F3F5AB...	Parkinson's

Data Preprocessing

```python
def match_audio_with_metadata(metadata, hc_audio_dir, pd_audio_dir):
    audio_data = []
    for _, row in metadata.iterrows():
        sample_id = row['Sample ID']
        label = row['Label']
        age = row['Age']
        sex = row['Sex']


        if label == 'HC':
            file_path = os.path.join(hc_audio_dir, f"{sample_id}.wav")
        else:
            file_path = os.path.join(pd_audio_dir, f"{sample_id}.wav")

        if os.path.exists(file_path):
            audio_data.append({
                'Sample ID': sample_id,
                'File Path': file_path,
                'Label': label,
                'Age': age,
                'Sex': sex
            })
        else:
            print(f"Audio file missing: {sample_id}.wav")

    return pd.DataFrame(audio_data)
```

```python
[ ] # Match audio files with metadata
    audio_metadata = match_audio_with_metadata(metadata, hc_audio_dir, pd_audio_dir)
```

Audio File Matching & Metadata Integration

- Matched audio files with metadata (Sample ID, Age, Sex).
- Identified and handled missing files.
- Structured the dataset into a DataFrame for feature extraction.

Data Preprocessing

Enhancing Model Generalization Through Audio Data Augmentation

Augmentation Techniques Used

-Noise Addition -Time Stretching -Pitch Shifting

```python
# 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

# Save augmented audio files
def augment_dataset(audio_metadata, output_dir):
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    augmented_metadata = []
    for _, row in audio_metadata.iterrows():
        file_path = row['File Path']
        label = row['Label']
        augmented_audios, sr = augment_audio(file_path)

        # Save augmented audios
        for i, augmented_audio in enumerate(augmented_audios):
            augmented_file = os.path.join(output_dir, f"{os.path.basename(file_path).split('.')[0]}_aug{i}.wav")
            sf.write(augmented_file, augmented_audio, sr)  # Save using soundfile
            augmented_metadata.append({
                'Sample ID': f"{row['Sample ID']}_aug{i}",
                'File Path': augmented_file,
                'Label': label
            })

    return pd.DataFrame(augmented_metadata)
```

Feature Extraction

Used YAMNet pre-trained audio feature extraction model

```python
import tensorflow_hub as hub
import tensorflow as tf

# Load YAMNet
yamnet_model = hub.load('https://tfhub.dev/google/yamnet/1')

def extract_yamnet_features(file_path):
    audio, sr = librosa.load(file_path, sr=16000)
    waveform = tf.convert_to_tensor(audio, dtype=tf.float32)
    scores, embeddings, spectrogram = yamnet_model(waveform)
    return embeddings.numpy().mean(axis=0)


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)
```

Data Preprocessing

Dataset Splitting & Label Encoding for Speech Classification

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform labels to numerical values
y_train = label_encoder.fit_transform(y_train).astype('float32')
y_test = label_encoder.transform(y_test).astype('float32')
```

- Split dataset into 80% training and 20% testing.
- Encoded categorical labels (HC, PD) into numerical values (0,1).

Model Training

```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],))
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

-Used a Fully Connected Neural Network (Dense Layers).

Built a deep learning model using transfer learning (YAMNet embeddings).

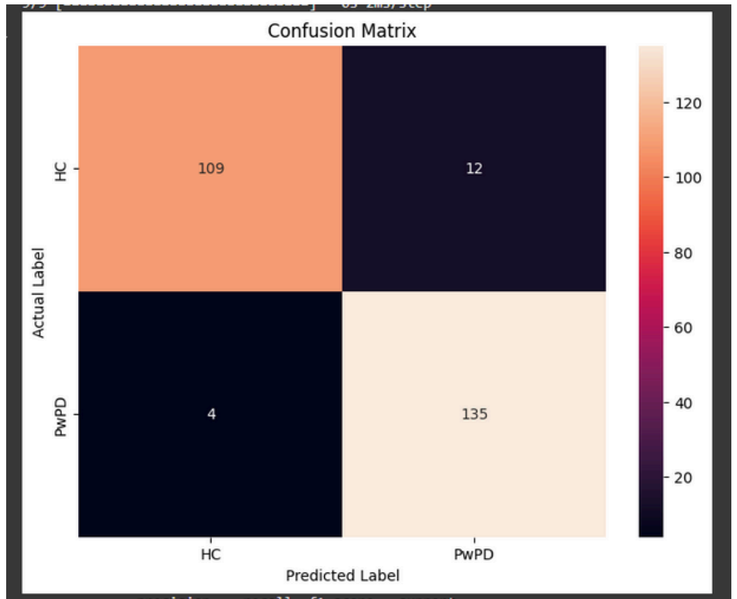
```python
[ ] history = model.fit(
        X_train, y_train,
        epochs=20,
        batch_size=16,
        validation_data=(X_test, y_test)
    )
```

```
65/65 [==============================] - 0s 4ms/step - loss: 0.1574 - accuracy: 0.9431 - val_loss: 0.1999 - val_accuracy: 0.9308
Epoch 16/20
65/65 [==============================] - 0s 4ms/step - loss: 0.1300 - accuracy: 0.9517 - val_loss: 0.2014 - val_accuracy: 0.9308
Epoch 17/20
65/65 [==============================] - 0s 4ms/step - loss: 0.1421 - accuracy: 0.9479 - val_loss: 0.1881 - val_accuracy: 0.9308
Epoch 18/20
65/65 [==============================] - 0s 4ms/step - loss: 0.1399 - accuracy: 0.9402 - val_loss: 0.1995 - val_accuracy: 0.9346
Epoch 19/20
65/65 [==============================] - 0s 4ms/step - loss: 0.1231 - accuracy: 0.9517 - val_loss: 0.1667 - val_accuracy: 0.9423
Epoch 20/20
65/65 [==============================] - 0s 4ms/step - loss: 0.1199 - accuracy: 0.9546 - val_loss: 0.1907 - val_accuracy: 0.9385
```

- Training Accuracy (Final Epoch 20): **95.46%**
- Validation Accuracy (Final Epoch 20): **93.85%**

Performance Evaluation

```python
[ ] from sklearn.metrics import confusion_matrix, classification_report
    import seaborn as sns
    import matplotlib.pyplot as plt

    # Evaluate model
    y_pred = (model.predict(X_test) > 0.5).astype(int).ravel()

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)

    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", xticklabels=['HC', 'PwPD'], yticklabels=['HC', 'PwPD'])
    plt.title("Confusion Matrix")
    plt.xlabel("Predicted Label")
    plt.ylabel("Actual Label")
    plt.show()

    # Classification Report
    print(classification_report(y_test, y_pred, target_names=['HC', 'PwPD']))
```




```
              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
```

- True Negatives (TN = 109)
- True Positives (TP = 135)
- False Positives (FP = 12)
- False Negatives (FN = 4)

- HC Precision: 96%, Recall: 90%
- PwPD Precision: 92%, Recall: 97%

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

Overview of the Dataset

```
[6]  # Get list of images
     healthy_images = [f for f in os.listdir(healthy_path) if f.lower().endswith(('.png', '.jpg', '.jpeg'))]
     parkinson_images = [f for f in os.listdir(parkinson_path) if f.lower().endswith(('.png', '.jpg', '.jpeg'))]
```

```
[7]  # Create a DataFrame for overview
     df_overview = pd.DataFrame({
         "Category": ["Healthy", "Parkinson"],
         "Total Images": [len(healthy_images), len(parkinson_images)],
         "Sample Filenames": [healthy_images[:3], parkinson_images[:3]]  # Show 3 samples
     })
```

```
# Print dataset summary
print("\n=== Dataset Overview ===")
print(df_overview)
display(df_overview)
```

```
=== Dataset Overview ===
    Category  Total Images                                   Sample Filenames
0    Healthy          1632  [Healthy1482.png, Healthy1490.png, Healthy1597...
1  Parkinson          1632  [Parkinson1617.png, Parkinson1481.png, Parkins...
```

	Category	Total Images	Sample Filenames
0	Healthy	1632	[Healthy1482.png, Healthy1490.png, Healthy1597...
1	Parkinson	1632	[Parkinson1617.png, Parkinson1481.png, Parkins...

There are 2 catogries.

- Healthy
- Parkinson

3264 Total Images

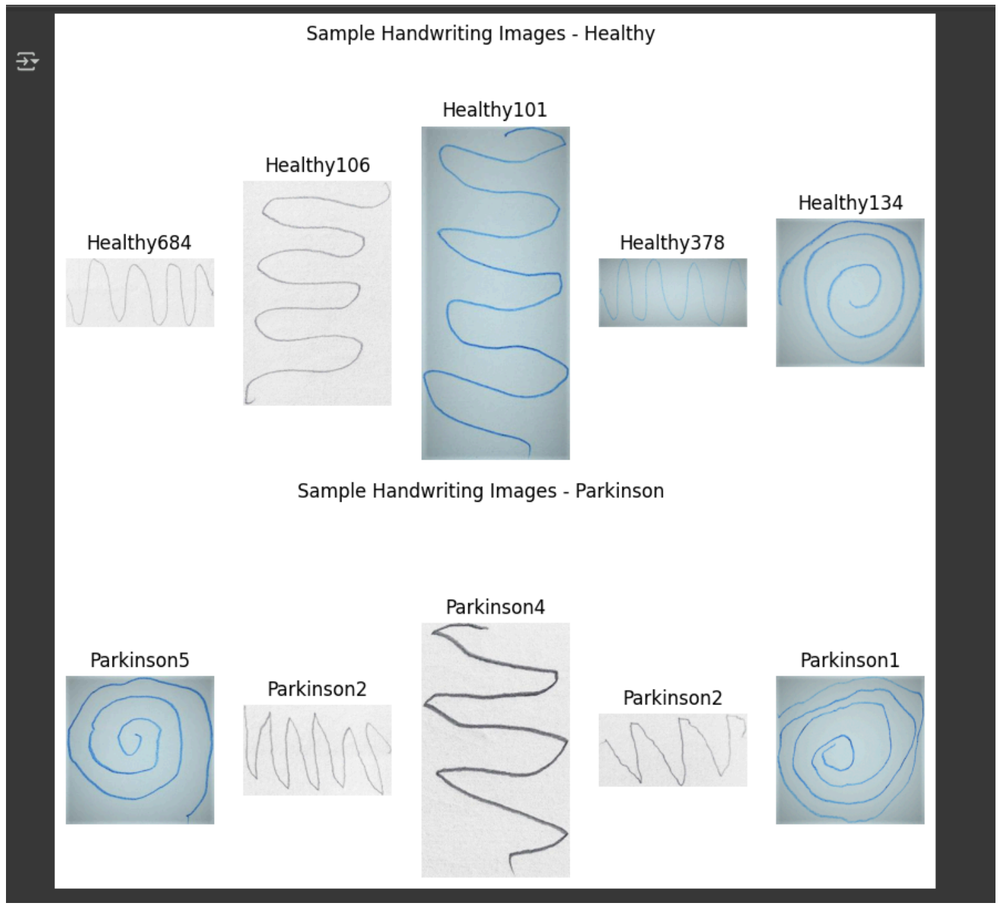
- Healthy - 1632
- Parkinson - 1632

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

Overview of the Dataset

Displays 5 random sample images from each category

```
[9]  # Show random sample images from each category
     def show_sample_images(folder_path, image_list, title):
         plt.figure(figsize=(10, 5))
         random_images = random.sample(image_list, min(5, len(image_list)))  # Pick 5
         for i, img_name in enumerate(random_images):
             img_path = os.path.join(folder_path, img_name)
             img = cv2.imread(img_path)
             img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
             plt.subplot(1, 5, i+1)
             plt.imshow(img)
             plt.axis("off")
             plt.title(img_name[:10])  # Show first 10 characters of filename
         plt.suptitle(title)
         plt.show()

     # Display 5 random sample images from each category
     show_sample_images(healthy_path, healthy_images, "Sample Handwriting Images - Hea
     show_sample_images(parkinson_path, parkinson_images, "Sample Handwriting Images -
```



Data Preprocessing

Dtaset loading & Splitting

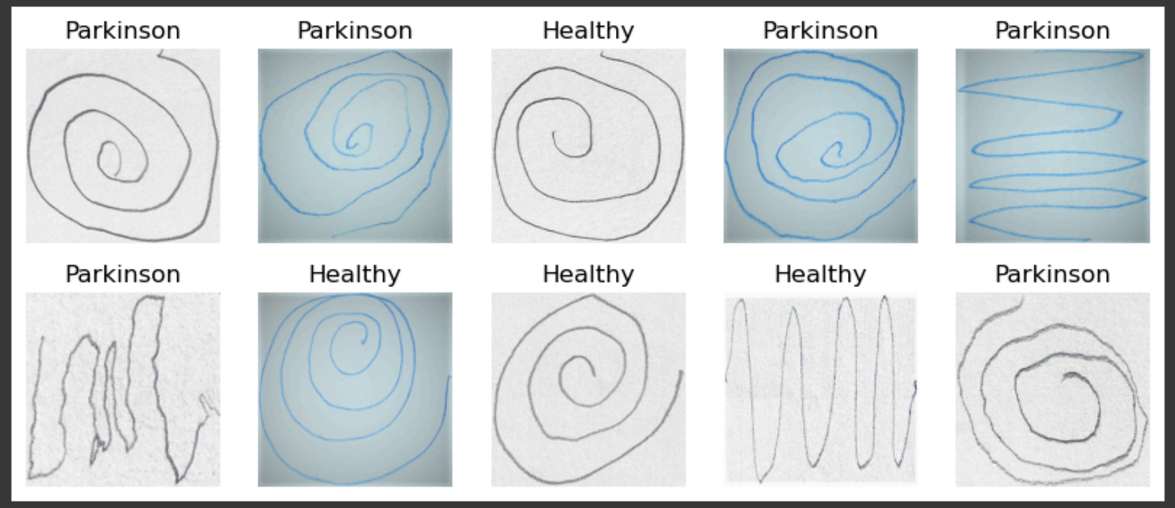
```
BATCH_SIZE = 32
IMG_SIZE = (224, 224)
```

```
[ ] 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
    )
```

Image Normalization

Normalizes pixel values in the training & validation dataset from [0,255] to [0,1].

```
plt.figure(figsize=(10, 4))
for image,label in train_data.take(1) :
  for i in range(10):
    ax = plt.subplot(2,5, i+ 1)
    plt.imshow(image[i].numpy().astype("uint8"))
    plt.title(class_names[label[i]])
    plt.axis("off")

plt.show()
```




```
[ ] 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))
```

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'])
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #

Load Pretrained MobileNetV2 for Feature Extraction

-Used MobileNetV2 as a feature extractor with frozen layers.

Added Custom Fully Connected Layers for Handwriting Classification

-Applied GlobalAveragePooling2D to convert feature maps into a single feature vector.

Compiled and Trained the Model

-Optimizer: Adam

Performance Evaluation

Training Accuracy & Validation Accuracy

```python
    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_checkpoint],
                    batch_size = 32
                    )


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

```
Epoch 1/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 622ms/step - accuracy: 0.7383 - loss: 0.5226
Epoch 1: val_loss improved from inf to 0.24858, saving model to best_model.keras
92/92 ━━━━━━━━━━━━━━━━━━━━ 76s 763ms/step - accuracy: 0.7390 - loss: 0.5217 - val_accuracy: 0.8988 - val_loss: 0.2486
Epoch 2/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 600ms/step - accuracy: 0.8755 - loss: 0.2946
Epoch 2: val_loss improved from 0.24858 to 0.17169, saving model to best_model.keras
92/92 ━━━━━━━━━━━━━━━━━━━━ 62s 672ms/step - accuracy: 0.8757 - loss: 0.2943 - val_accuracy: 0.9479 - val_loss: 0.1717
Epoch 3/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 592ms/step - accuracy: 0.9026 - loss: 0.2295
Epoch 3: val_loss improved from 0.17169 to 0.15361, saving model to best_model.keras
92/92 ━━━━━━━━━━━━━━━━━━━━ 62s 667ms/step - accuracy: 0.9026 - loss: 0.2294 - val_accuracy: 0.9448 - val_loss: 0.1536
Epoch 4/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 603ms/step - accuracy: 0.9230 - loss: 0.1991
Epoch 4: val_loss improved from 0.15361 to 0.11844, saving model to best_model.keras
92/92 ━━━━━━━━━━━━━━━━━━━━ 62s 673ms/step - accuracy: 0.9231 - loss: 0.1990 - val_accuracy: 0.9632 - val_loss: 0.1184
Epoch 5/5
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 577ms/step - accuracy: 0.9377 - loss: 0.1572
Epoch 5: val_loss improved from 0.11844 to 0.10694, saving model to best_model.keras
92/92 ━━━━━━━━━━━━━━━━━━━━ 60s 649ms/step - accuracy: 0.9378 - loss: 0.1571 - val_accuracy: 0.9755 - val_loss: 0.1069
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is consider
```

Used MobileNetV2 with transfer learning, and the training process was enhanced with Early Stopping and Model Checkpointing.

Training Accuracy

93.8%

Validation Accuracy

97.6%

Performance Evaluation

Training Accuracy & Validation Accuracy

```
[ ] 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
```

Model Evaluation on Validation Data

Final Validation Accuracy: 97.6%

Final Validation Loss: 0.1069

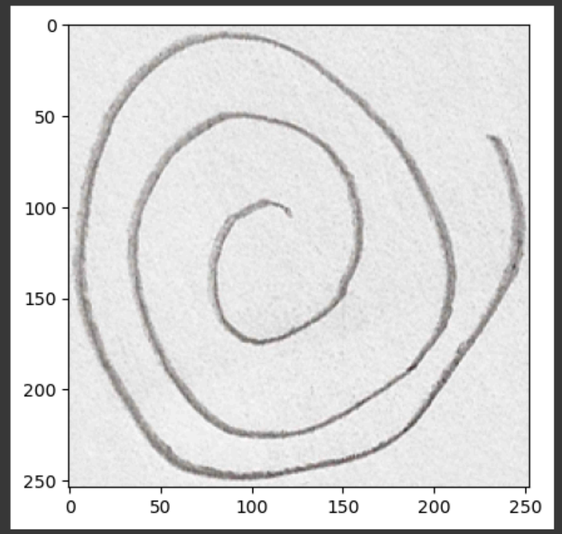
Performance Evaluation

Training Accuracy & Validation Accuracy

Prediction Process

```
[ ] image = cv2.imread('../../DataSets/hehe/Healthy/Healthy80.png')
```

```
[ ] plt.imshow(image)
    plt.show()
```




```
resized_img = tf.image.resize(image, (IMG_SIZE))
scaled_img = resized_img / 255
```

```
[ ] np.expand_dims(scaled_img, axis=0).shape
```

```
(1, 224, 224, 3)
```

```
[ ] y_hat = model.predict(np.expand_dims(scaled_img, axis=0))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
```

```
[ ] y_hat
```

```
array([[0.08288653]], dtype=float32)
```

```
[ ] class_names
```

```
['Healthy', 'Parkinson']
```

```
[ ] if y_hat > 0.5:
        print(f'Predicted class is healthy')
    else:
        print(f'Predicted class is parkinson')
```

```
Predicted class is parkinson
```

Multimodal Late Fusion for Parkinson's Prediction

```python
def get_clinical_prediction(input_data):
    # Get predictions from the clinical model
    clinical_pred = clinical_model.predict_proba([input_data])[0]  # [Low, Moderate, High]
    return clinical_pred

def get_speech_prediction(audio_file):
    # Extract MFCC features and get predictions from the speech model
    y, sr = librosa.load(audio_file, sr=22050)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=32)
    mfcc = mfcc.T  # Transpose the MFCC matrix

    # Flatten the MFCCs to a 1D array with 1024 features
    mfcc = mfcc.flatten()[:1024]  # Trim or pad to 1024

    # Reshape to match the expected input shape (None, 1024)
    mfcc = mfcc.reshape(1, -1)  # -1 infers the dimension

    # Check the input shape expected by the speech model
    print("Input shape expected by the model:", speech_model.input_shape)
    print("Shape of MFCCs:", mfcc.shape)
    speech_pred = speech_model.predict(mfcc)[0]  # [Low, Moderate, High]
    return speech_pred

def get_handwriting_prediction(img_path):
    # Preprocess handwriting image and get predictions from the handwriting model
    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)
    handwriting_pred = handwriting_model.predict(img_array)[0]  # [Low, Moderate, High]
    return handwriting_pred
```

Defines functions for getting predictions from each individual model (clinical, speech, handwriting) before applying late fusion.

get_clinical_prediction(input_data)

get_speech_prediction(audio_file)

get_handwriting_prediction(img_path)

Multimodal Late Fusion for Parkinson’s Prediction

```python
def apply_late_fusion(clinical_pred, speech_pred, handwriting_pred):
    """Combine predictions using Weighted Averaging (adjusted weights)."""

    weights = [0.5, 0.3, 0.2]

    final_score = [
        (clinical_pred[0] * weights[0]) + (speech_pred[0] * weights[1]) + (handwriting_pred[0] * weights[2]),
        (clinical_pred[1] * weights[0]) + (speech_pred[1] * weights[1]) + (handwriting_pred[1] * weights[2]),
        (clinical_pred[2] * weights[0]) + (speech_pred[2] * weights[1]) + (handwriting_pred[2] * weights[2])
    ]

    risk_levels = ["Low Risk", "Moderate Risk", "High Risk"]
    final_prediction = risk_levels[np.argmax(final_score)]

    return final_prediction
```

Define Weights

Assigns higher weight (0.5) to clinical predictions

Assigns moderate weight (0.3) to speech model

Assigns lower weight (0.2) to handwriting model

Compute Weighted Scores

Computes the final probability score for each risk level (Low, Moderate, High).

Multimodal Late Fusion for Parkinson’s Prediction

Takes patient input from three sources (clinical, speech, handwriting).

Each model predicts Parkinson’s risk levels (Low, Moderate, High).

Late fusion integrates all three outputs using weighted averaging.

Final prediction is selected based on the highest probability.

```
[ ] # Example inputs (replace with real values)
    clinical_sample = [53, 16.902, 270.33, 134.18, 3.09, 8.40, 0, 0, 0, 0]  # Example clinical features
    speech_file = "AH_064F_7AB034C9-72E4-438B-A9B3-AD7FDA1596C5.wav"  # Replace with actual speech file path
    handwriting_img = "Healthy1.png"  # Replace with actual handwriting image path

    # Get individual predictions
    clinical_pred, speech_pred, handwriting_pred = get_predictions(clinical_sample, speech_file, handwriting_img)

    # Verify Shapes Again
    print("Final Clinical Prediction Shape:", clinical_pred.shape)
    print("Final Speech Prediction Shape:", speech_pred.shape)
    print("Final Handwriting Prediction Shape:", handwriting_pred.shape)

    # Ensure all models return exactly 3 probabilities
    if clinical_pred.shape[0] != 3 or speech_pred.shape[0] != 3 or handwriting_pred.shape[0] != 3:
        raise ValueError("One or more models are not returning 3 probability values. Check model outputs.")

    # Apply Late Fusion
    final_risk_level = apply_late_fusion(clinical_pred, speech_pred, handwriting_pred)

    print("✅ Final Parkinson’s Risk Level:", final_risk_level)
```

```
⚠️ Warning: Clinical model only returned 2 probabilities. Fixing output.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted wi
  warnings.warn(
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
Fixed Clinical Prediction Shape: (3,)
Fixed Speech Prediction Shape: (3,)
Fixed Handwriting Prediction Shape: (3,)
Final Clinical Prediction Shape: (3,)
Final Speech Prediction Shape: (3,)
Final Handwriting Prediction Shape: (3,)
✅ Final Parkinson’s Risk Level: Low Risk
```

Completion & What's left?

90%

Completed	Remaining on Finals
Data Gathering	Front-end Development
Clinical Data Model Building	Data Visualization Report
Speech Data Model Building	Novelty Part
Handwriting Data Model Building	
Back-end Deployment	

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.*

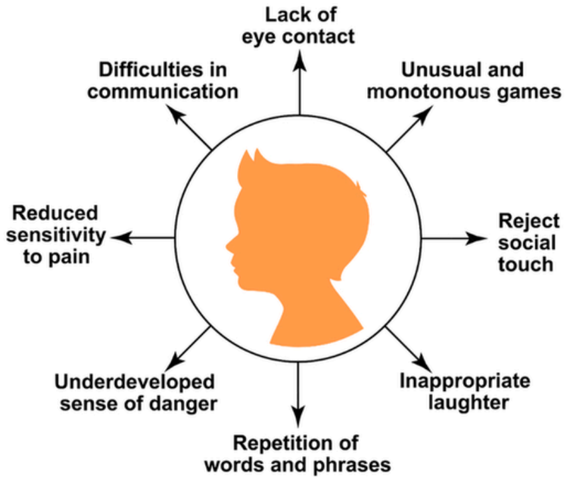
[4] *Shaban, M. (2023). Deep learning for Parkinson's disease diagnosis: a short survey. Computers, 12(3), 58.*

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



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
- Lower Costs – Reduces long-term medical and education expenses.
- Better Life Quality – Early help improves social and learning skills.

Research Question?

Challenges in Existing ASD Detection Methods

- Delayed Diagnosis
- Questionnaire answers may not always be accurate

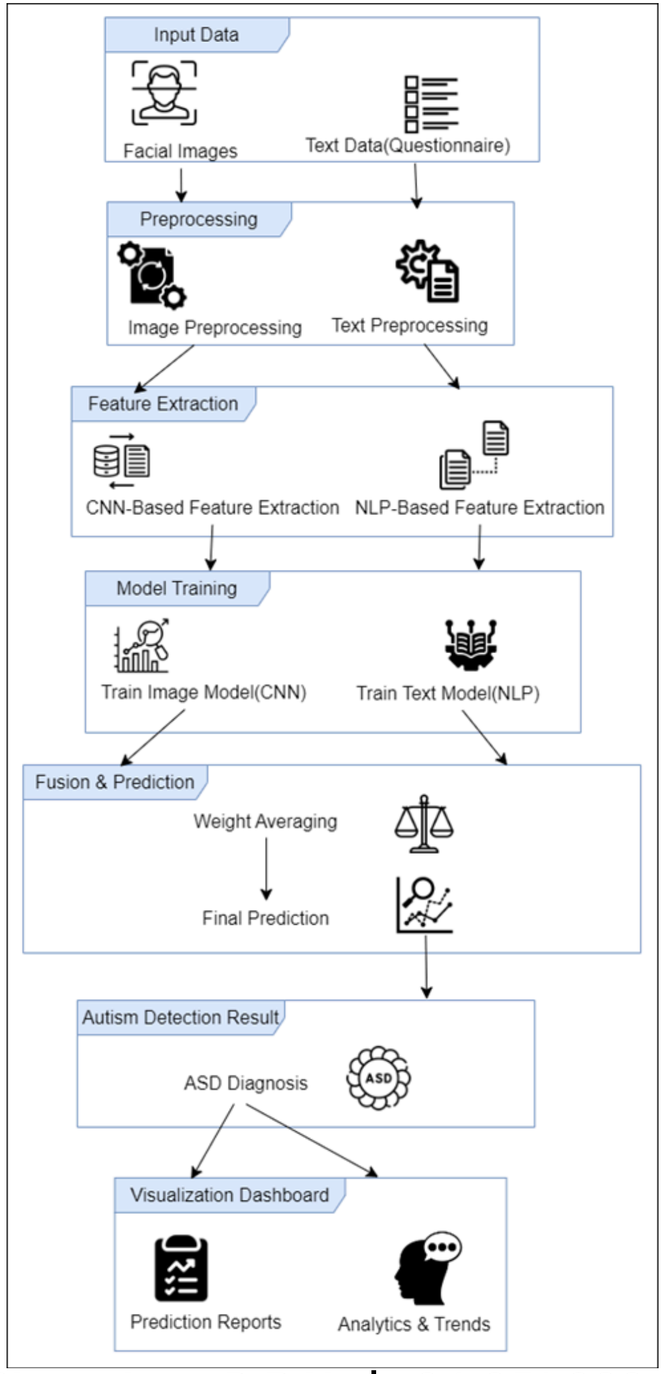
Why Combine Facial Images & Questionnaire Data?

- Improves Accuracy
- Captures More Features & Faster Screening

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



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?

Completion & What's left?

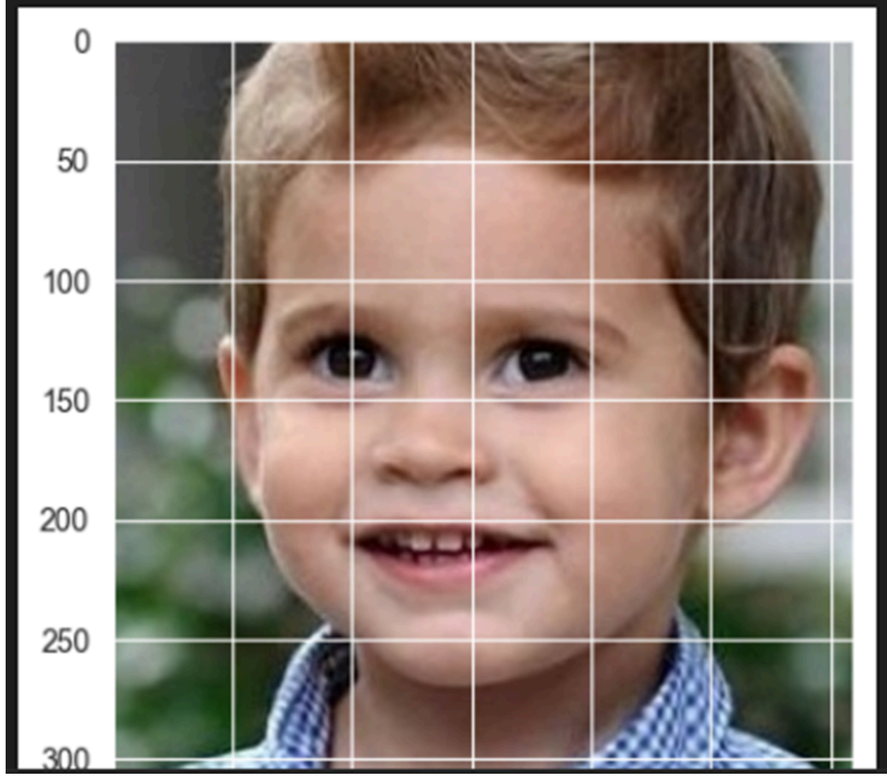
90%

Completed in PP2	Remaining on Finals
Data Gathering	Performance Optimization
Image and Text Data preprocessing	Enhancing Multimodal Fusion Techniques
Frontend & Backend Development	
Combining Model Use Fusion Technique (Weighted Average Approach)	
Data Visualization	
Model Fine-Tuning & Hyperparameter Tuning	

Evidence of Completion

Overview of the Autism Image 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 For Image Data

1. Preprocessing Techniques:

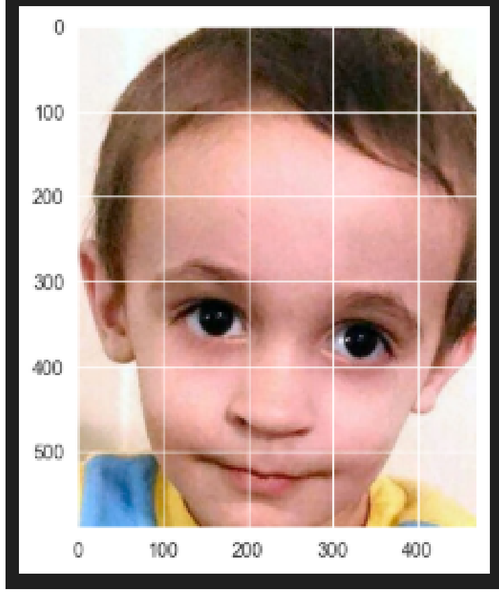
- ***Loading and Converting Images*** - Opening an image file so I can work with it on the computer.
- ***Grayscale Conversion*** - Turning a color image into shades of gray.
- ***Resizing Images*** - Changing the size of an image (making it smaller or bigger).
- ***Normalization*** - Changing the brightness of an image

2. Image Enhancement & Augmentation:

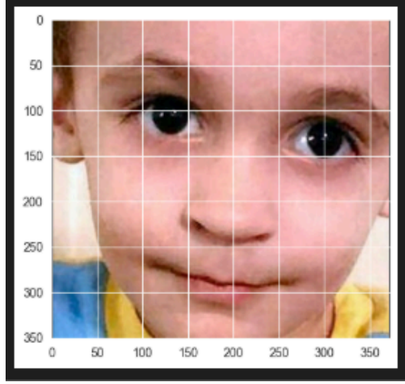
- ***Face Detection and cropping*** - Finding faces in a photo and cutting out the part with the face.
- ***Image Alignment & Rotation Correction*** - Fixing an image that's tilted or not aligned properly.
- ***Brightness & Contrast Adjustment*** - Changing how bright or dark the image is
- ***Noise Reduction*** - Removing blurry or random marks in the image to make it clearer.
- ***Data Augmentation*** - Making new versions of an image by rotating, flipping, or zooming

Data Preprocessing For Image Data

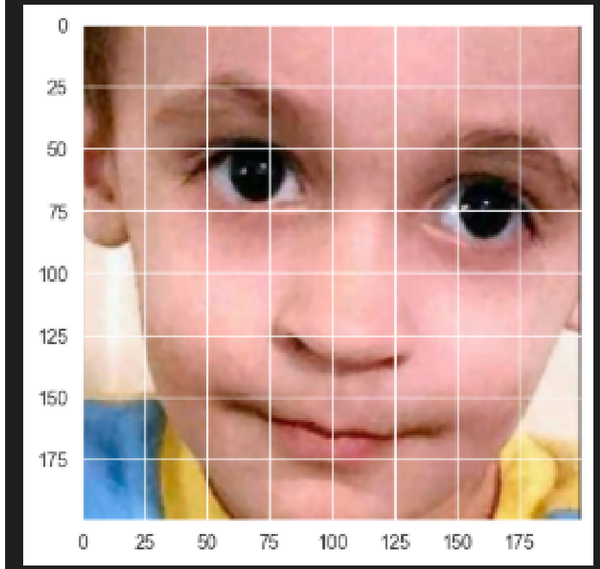
```python
sample2=[]
def sampledata():
    for i in training_data[0:2]:
        img=cv2.imread(i)
        sample=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        imgplot = plt.imshow(sample)
        plt.show()
        sample2.append(sample)
    return sample2
sample_data=sampledata()
```




```python
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 ens
    # 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)
```

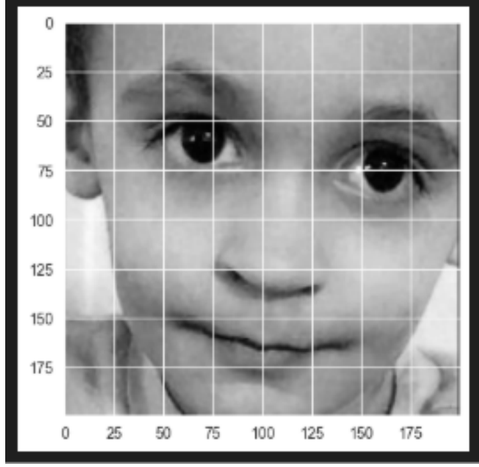



```python
resizedimg=[]
def ReSize(detimg):
    img=cv2.resize(detimg,(200,200)
    imgplot = plt.imshow(img)
    plt.show()
    resizedimg.append(img)
    return resizedimg

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

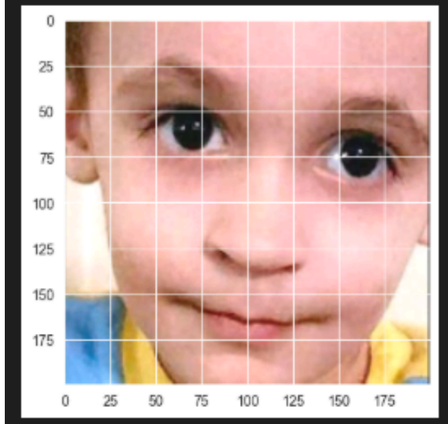


Data Preprocessing For Image Data

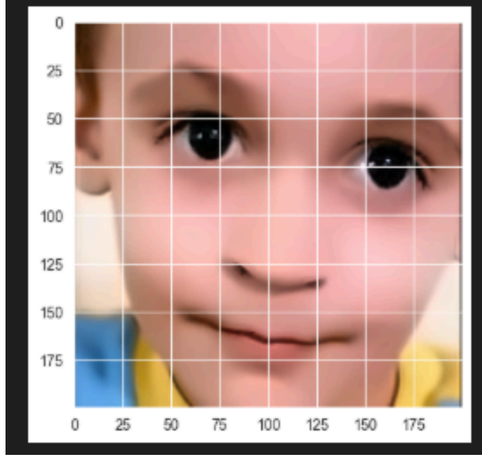
```python
grey=[]
def gray(resimg):
    img=cv2.cvtColor(resimg, cv2.COLOR_BGR2GRAY)
    I=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    imgplot = plt.imshow(I)
    plt.show()
    grey.append(I)
    return grey

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



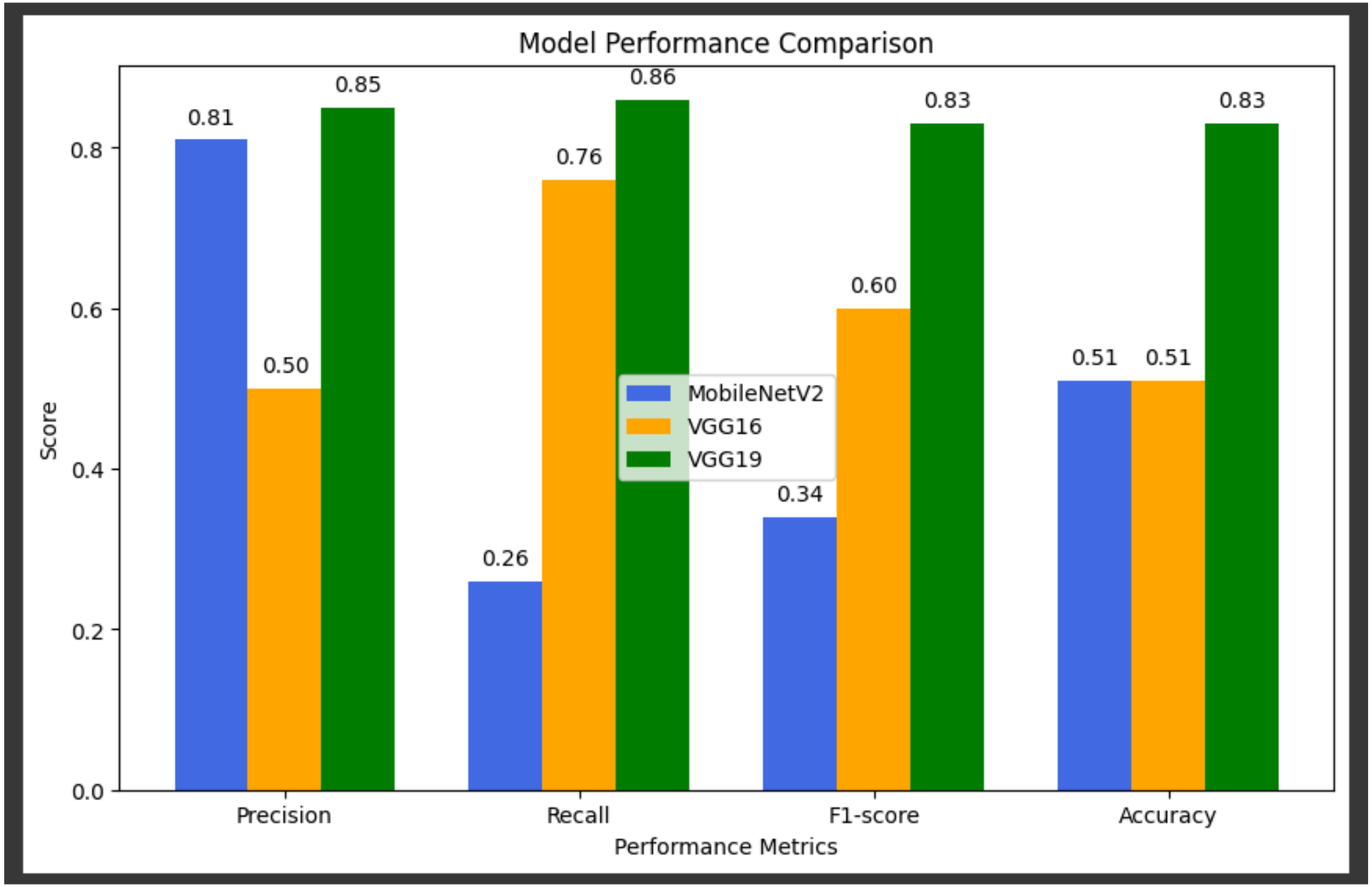

```python
bc=[]
def Brightnessandcontrast(resimg):
    Bright = 20
    Contrast = 1
    img=cv2.cvtColor(resimg,cv2.COLOR_BGR2RGB)
    adjusted = cv2.convertScaleAbs(img, alpha=Contrast, beta=Bright)
    rgbadjusted=cv2.cvtColor(adjusted,cv2.COLOR_BGR2RGB)
    imgplot = plt.imshow(rgbadjusted)
    plt.show()
    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,10,1
    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)
```

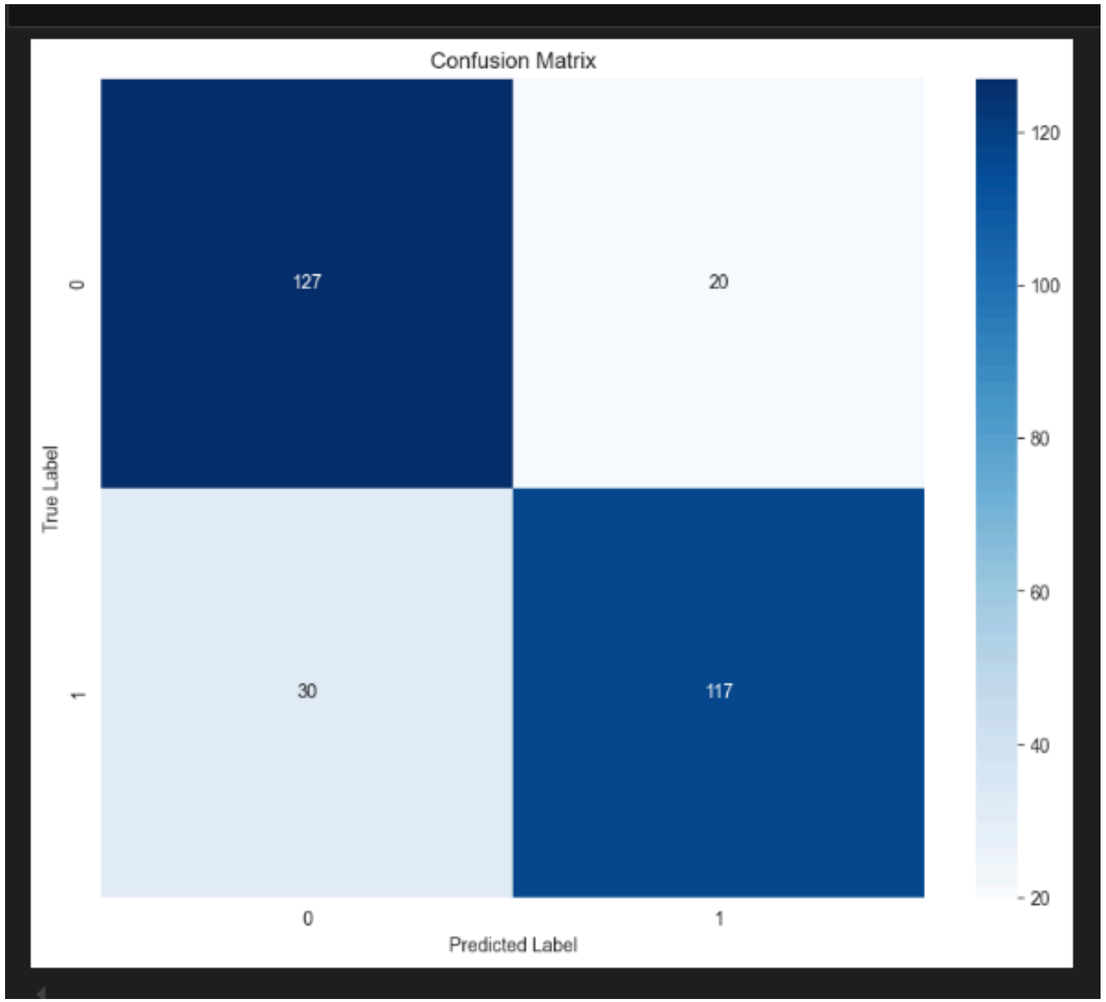


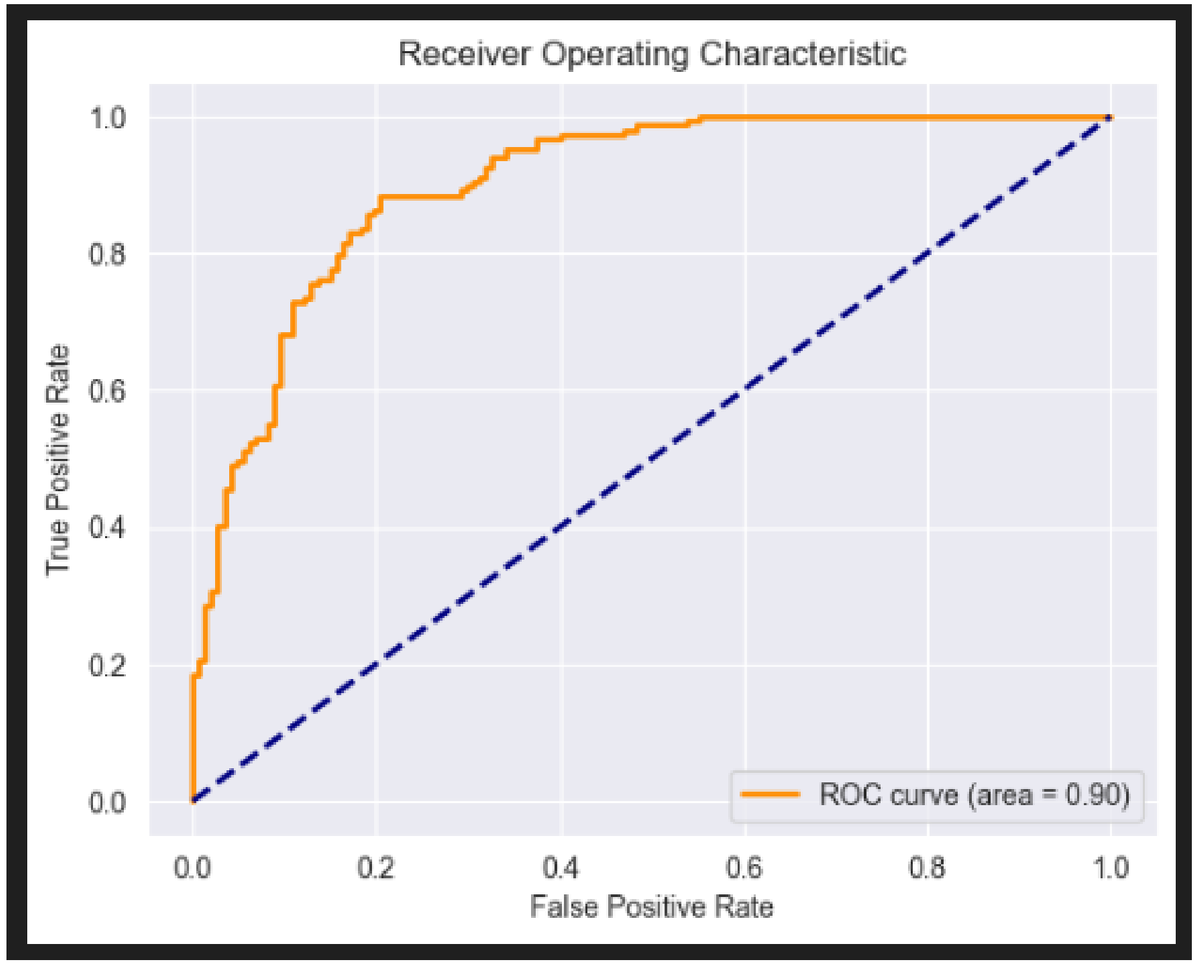
Model Comparison to find the best model



Performance Evaluation

Based on the VGG19 model



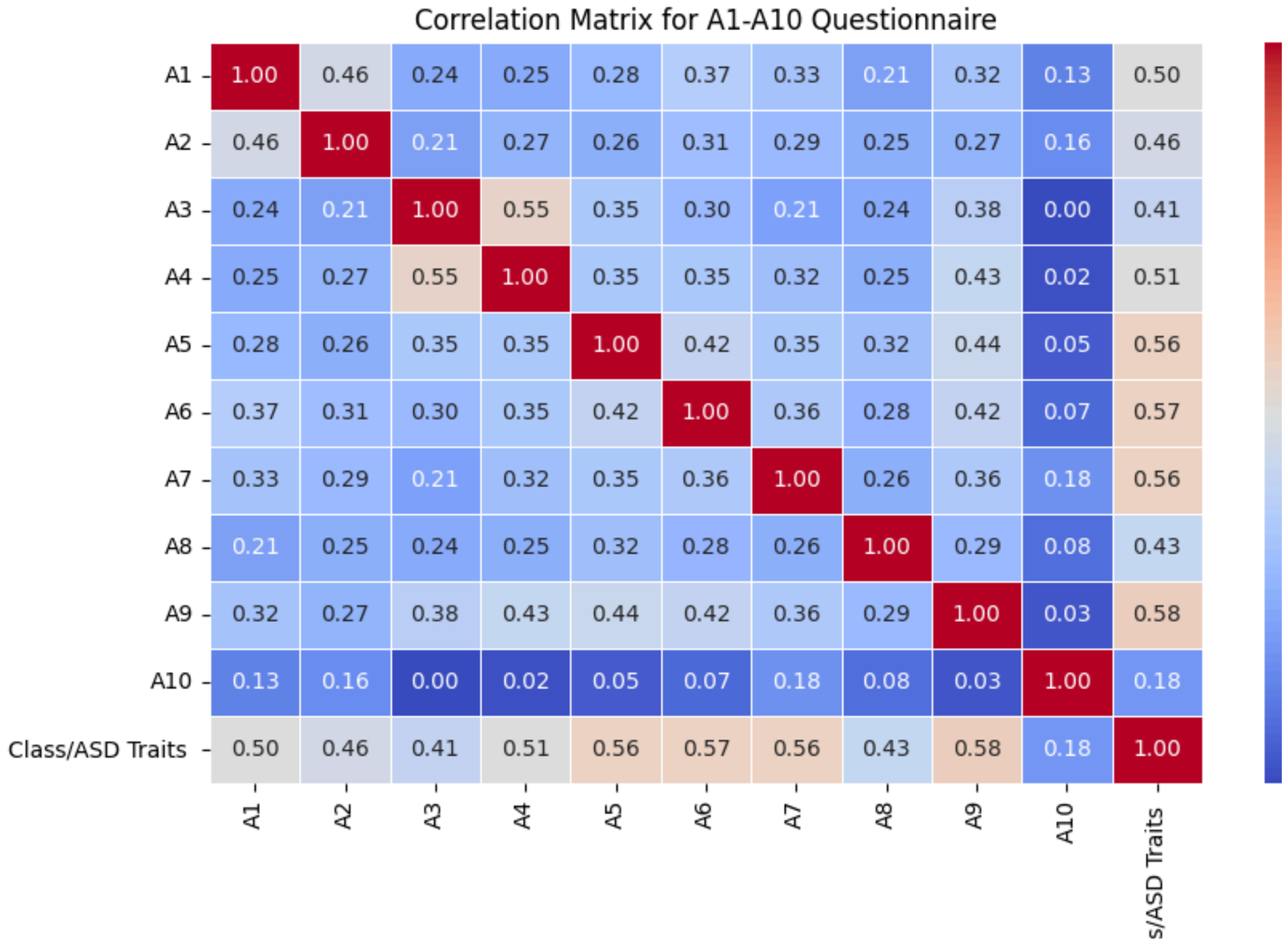


Text-Based Model

Autism screening data for toddlers

	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

Best Features Selection For Autism Toddler Dataset



- **A1, A5, A6, A7, and A9** show moderate to strong correlations (0.50+ range) with ASD traits.
- A10 has a weak correlation (0.18), suggesting it contributes less to ASD detection.

Data Preprocessing For Text Base Data

Stage 1: Data Cleaning & Transformation

- Handling Missing Values & Data Inspection
- Encoding Categorical Features
- Feature Engineering: Q-Chat-10 Score Calculation

Stage 2: Feature Selection & Data Balancing

- Feature Selection Using P-values
- Handling Class Imbalance with SMOTE

Additional Techniques for Data Preprocessing

- Outlier Detection & Removal
- Dimensionality Reduction - Feature Extraction(Transform data into a lower-dimensional space)

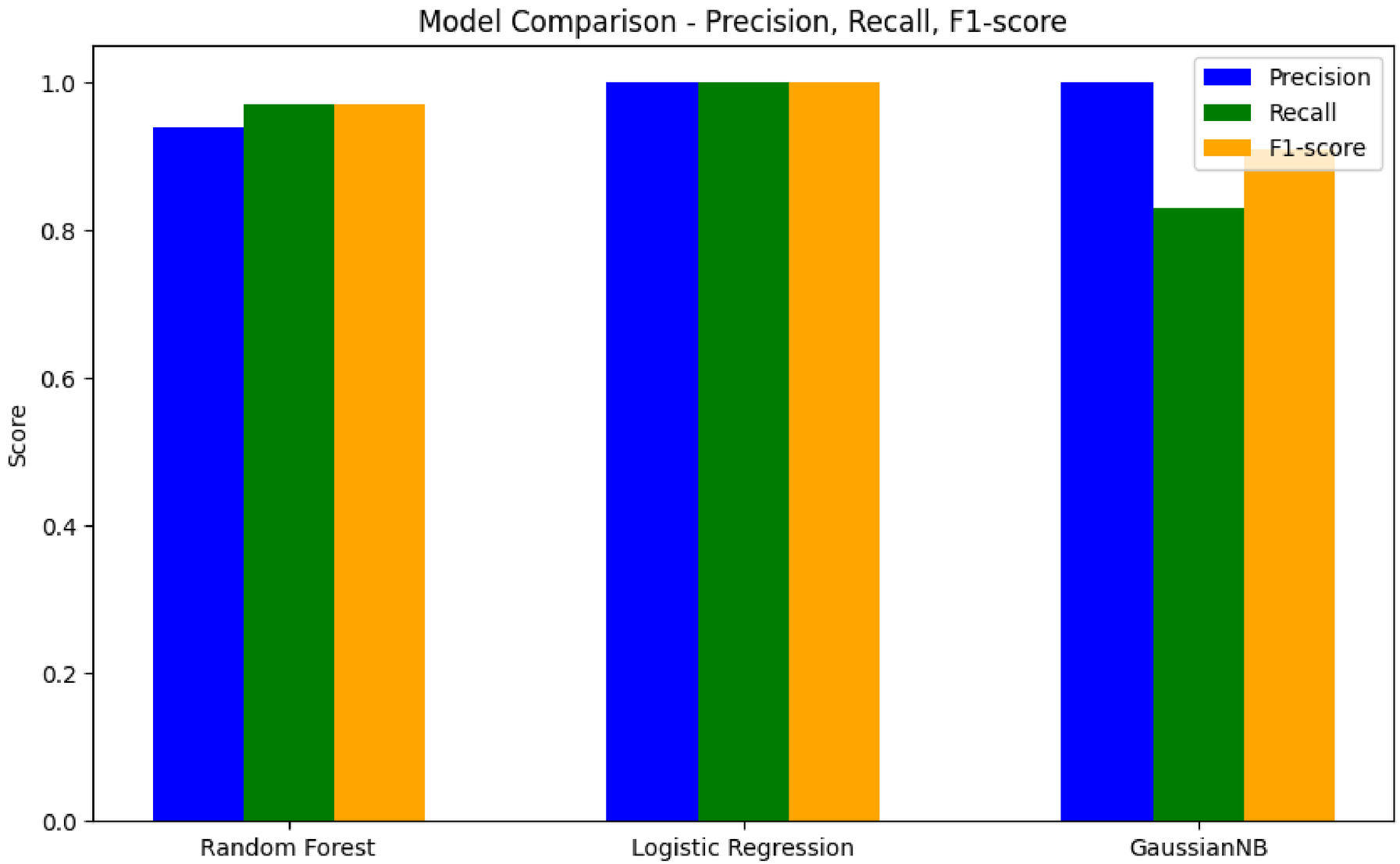
Model Training

- GaussianNB
- LogisticRegression
- RandomForest
- Kneighbors

```
[ ] # 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)
    }
```

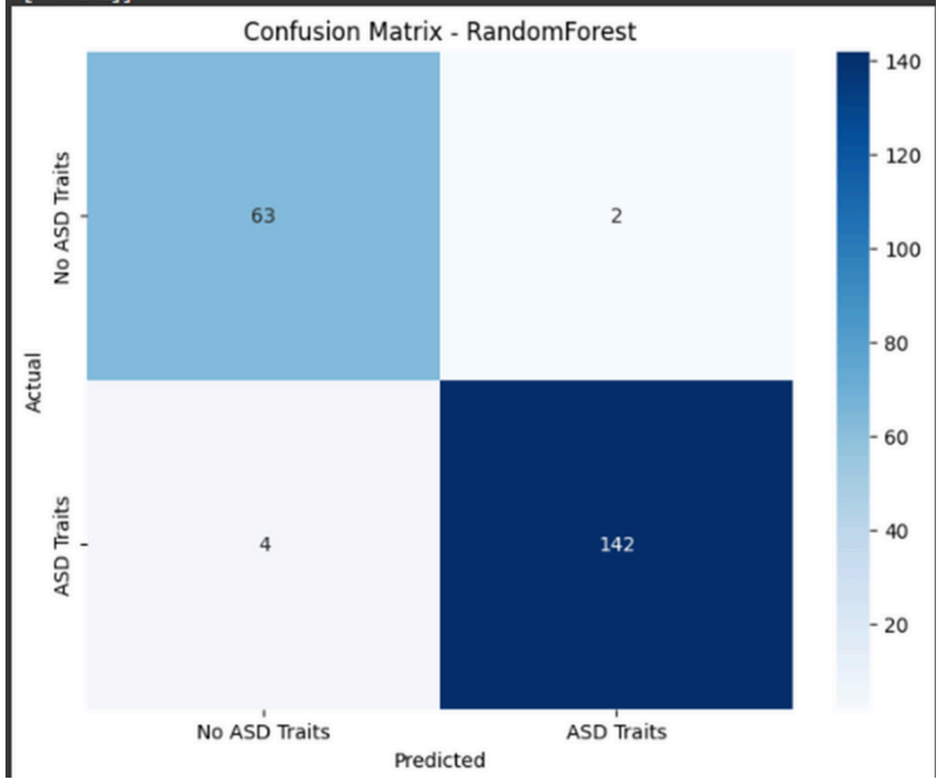
```
[ ] # Function to train, evaluate, and visualize results
    def train_evaluate_model(model, name, X_train, y_train, X_test, y_test):
        print(f"Training and Evaluating {name}...")
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        y_prob = model.predict_proba(X_test)[:,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

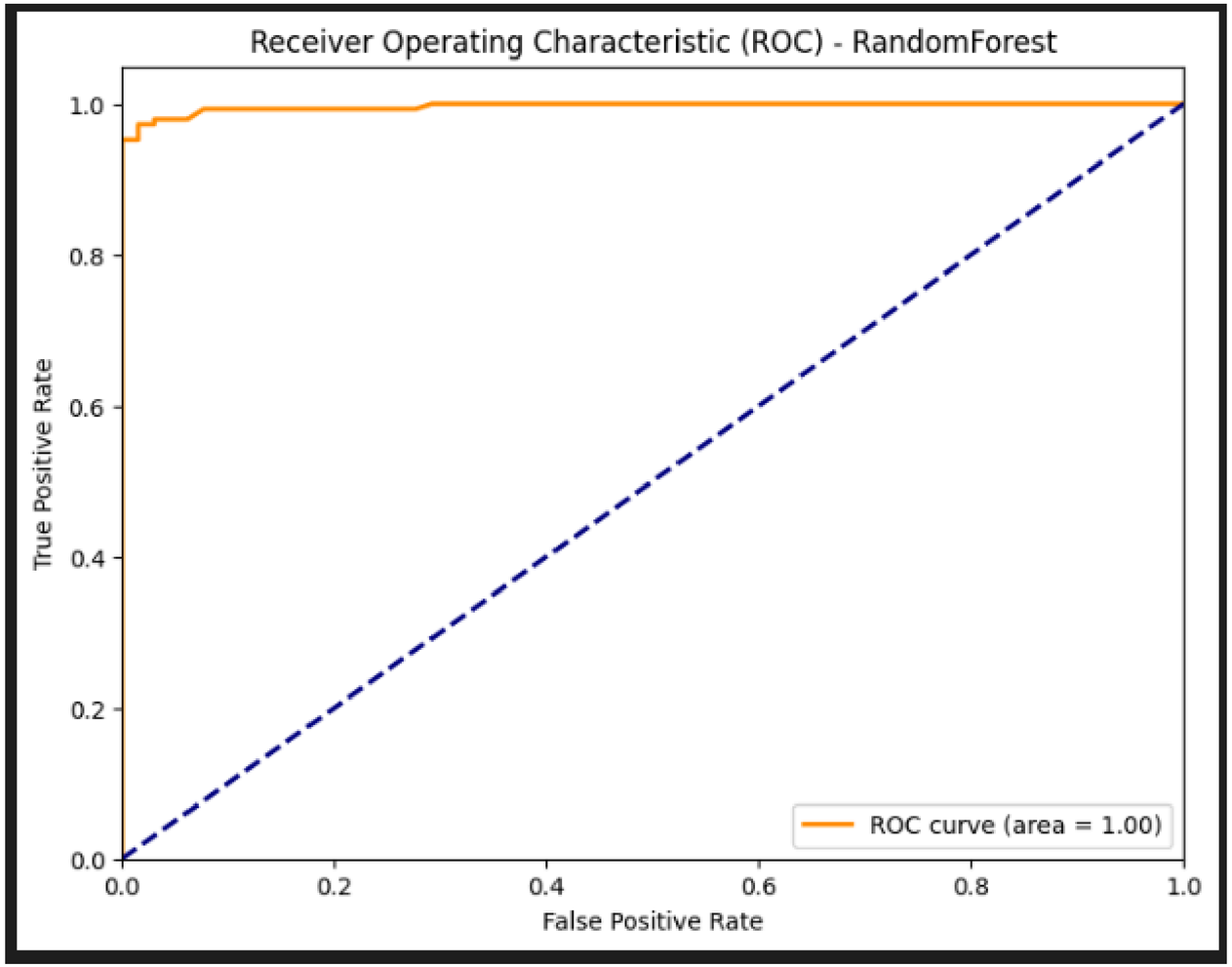
Model Comparison - Text Base Data



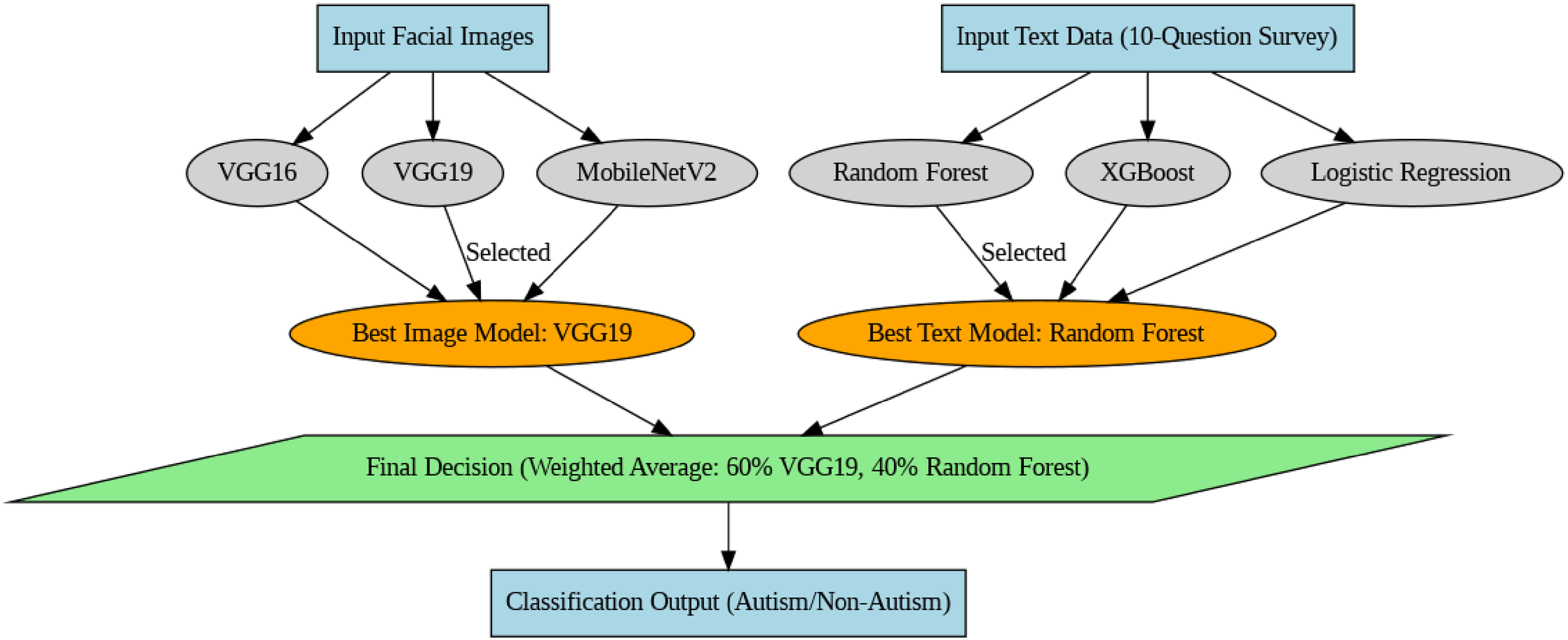
Performance Evaluation

```
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]]
```





Fusion Strategy-Weighted Average Approch

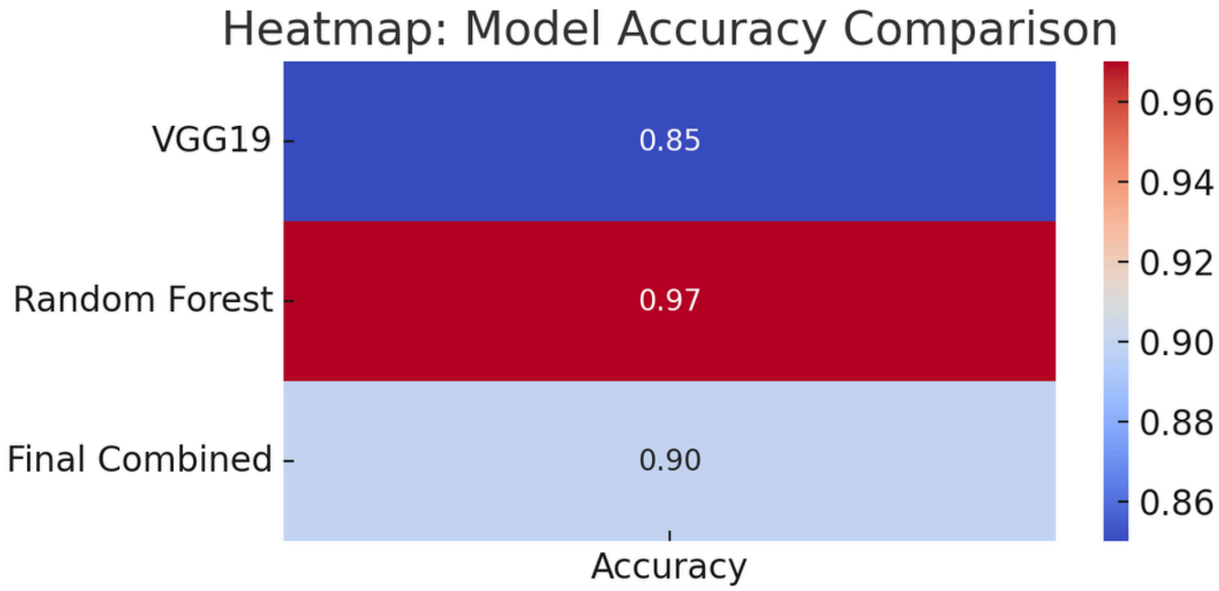


Weighted Average Approach for text and image base data

```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 av
            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
    # 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
```

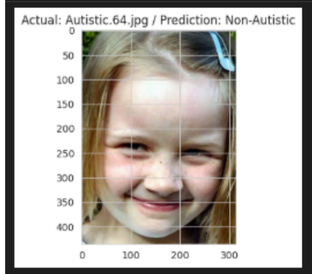
Model Evaluation and Classification Report for Autism Detection

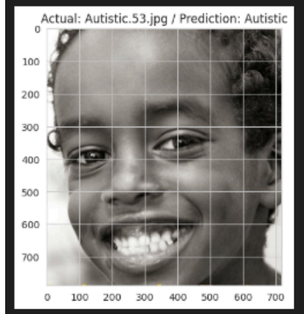



```
Image classification model loaded successfully.
Random Forest model loaded successfully.
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 885ms/step

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]
```

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

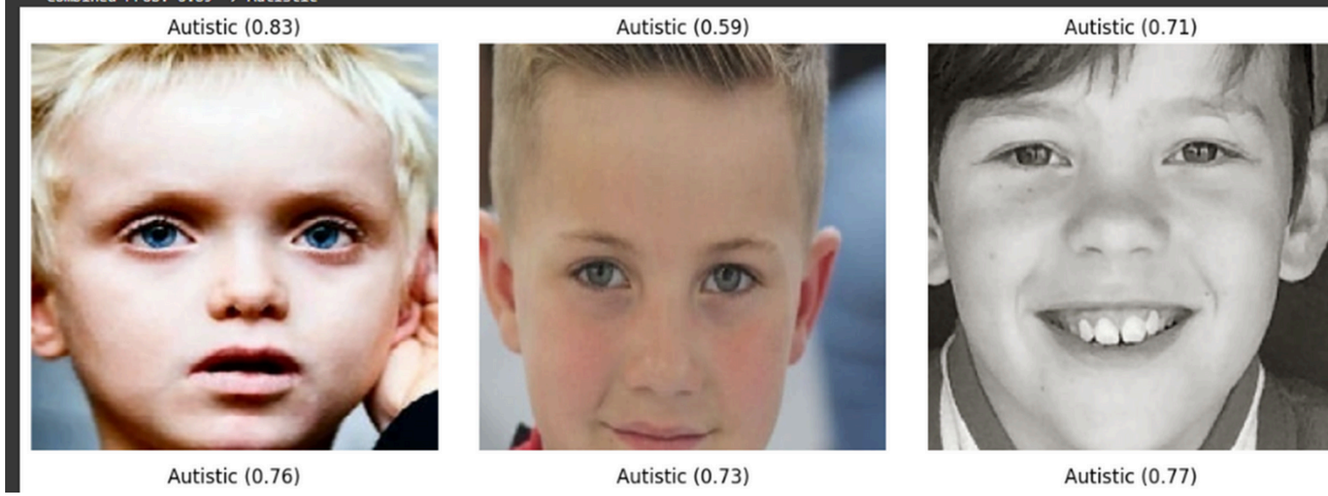
```python
# Let's check our predcitions against some test images
plt.figure(figsize=(4,4))
for val, i in enumerate(test_imgs[:10]):
    img = mpimg.imread(i)
    imgplot = plt.imshow(img)
    plt.title('Actual: ' + os.path.basename(i) + ' / Prediction: ' +  f"{'Autistic' if predictions[val] == 1 else 'Non-Autistic'}")
    plt.show()
```





Use only facial Images classification

```
   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
```



Use Fusion Technique

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.

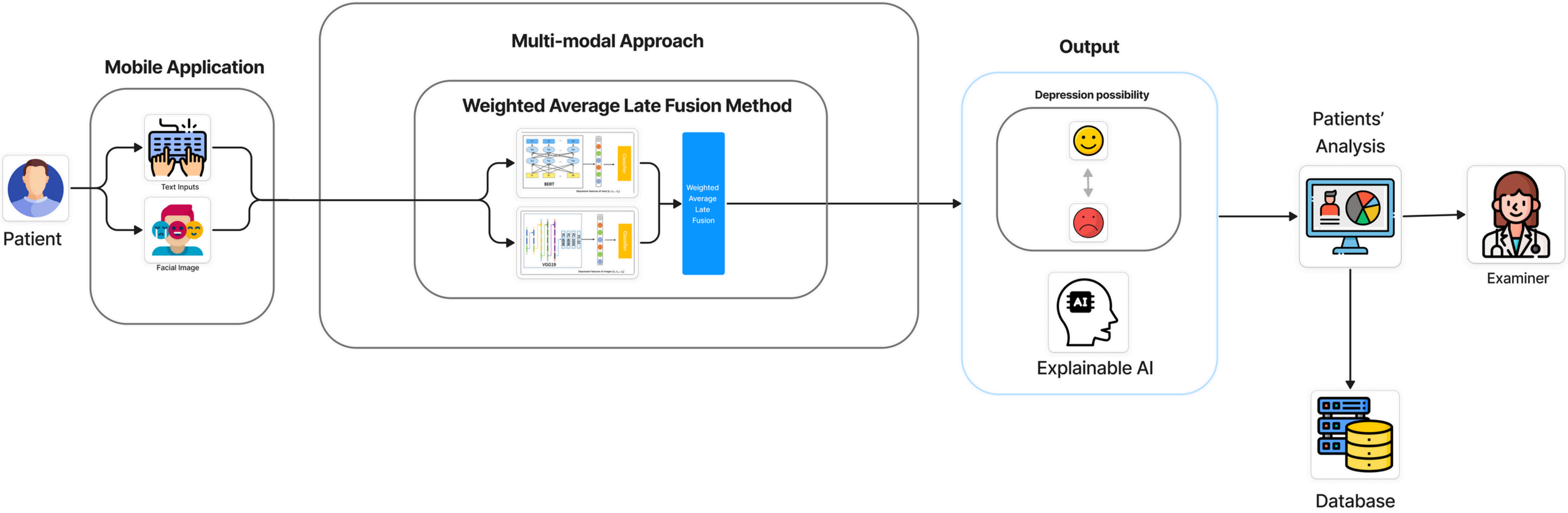
Research Questions

- 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 analyzing text inputs and facial images together.
- Give analysis/summary about patients’ current status to doctor.

Methodology



Completion & What's left?

90%

Completed	Remaining on Finals
Data Gathering	Model Enhancement
Text & Image data preprocessing	System Interface Enhancements
Text & Image processing model building, enhancing & evaluation	System Testing
Model Comparison	
Used Explainable AI	
Multimodal fusion method development	
Frontend & Backend Development	

Evidence of Completion - Datasets

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

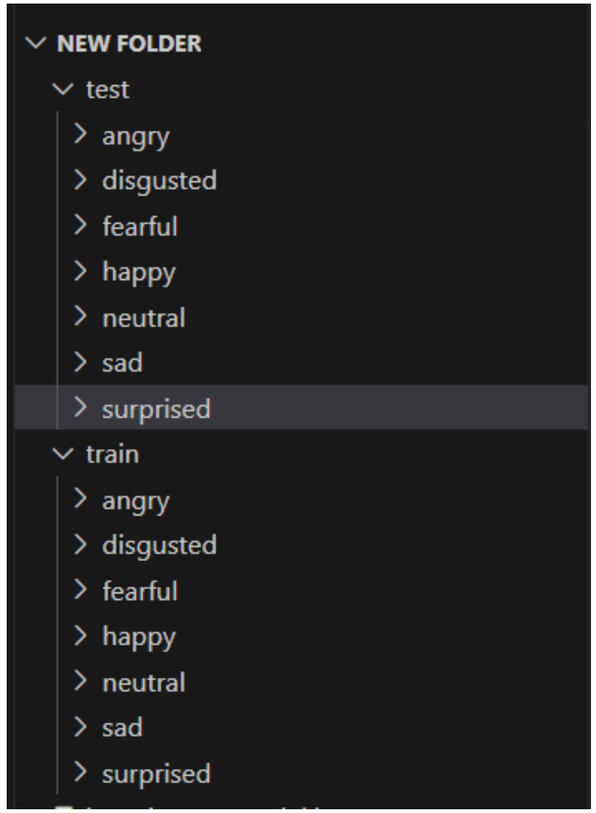
```
df.head()
```

	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

```
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
```



Text Preprocessing

Steps used :

- Tokenization
- Stopword Removal
- Punctuation & Special Character Removal
- TF-IDF Vectorization

Why important?

- Normalize text
- Reduce noise
- Prepare data for model training

Text Embedding

Approaches used :

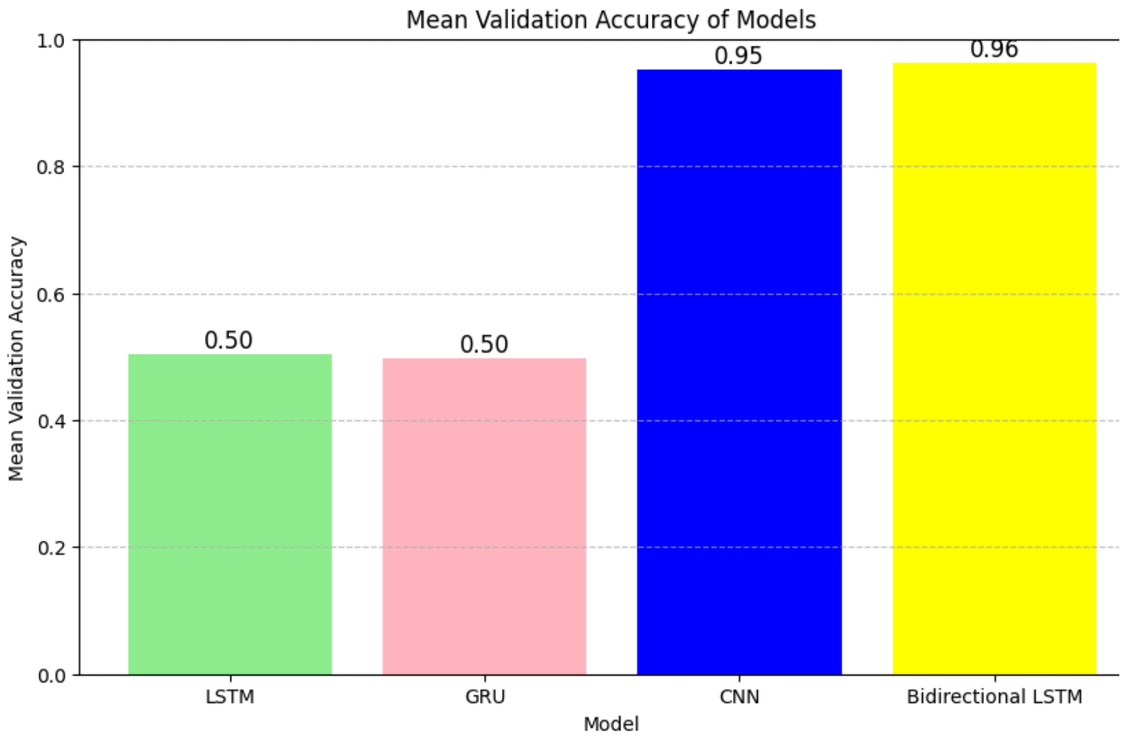
- Tokenizer and pad_sequences() for deep learning models.
- TF-IDF Vectorizer for traditional machine learning models.
- GloVe Embedding - unsupervised learning algorithm for obtaining vector representations for words.

Why important?

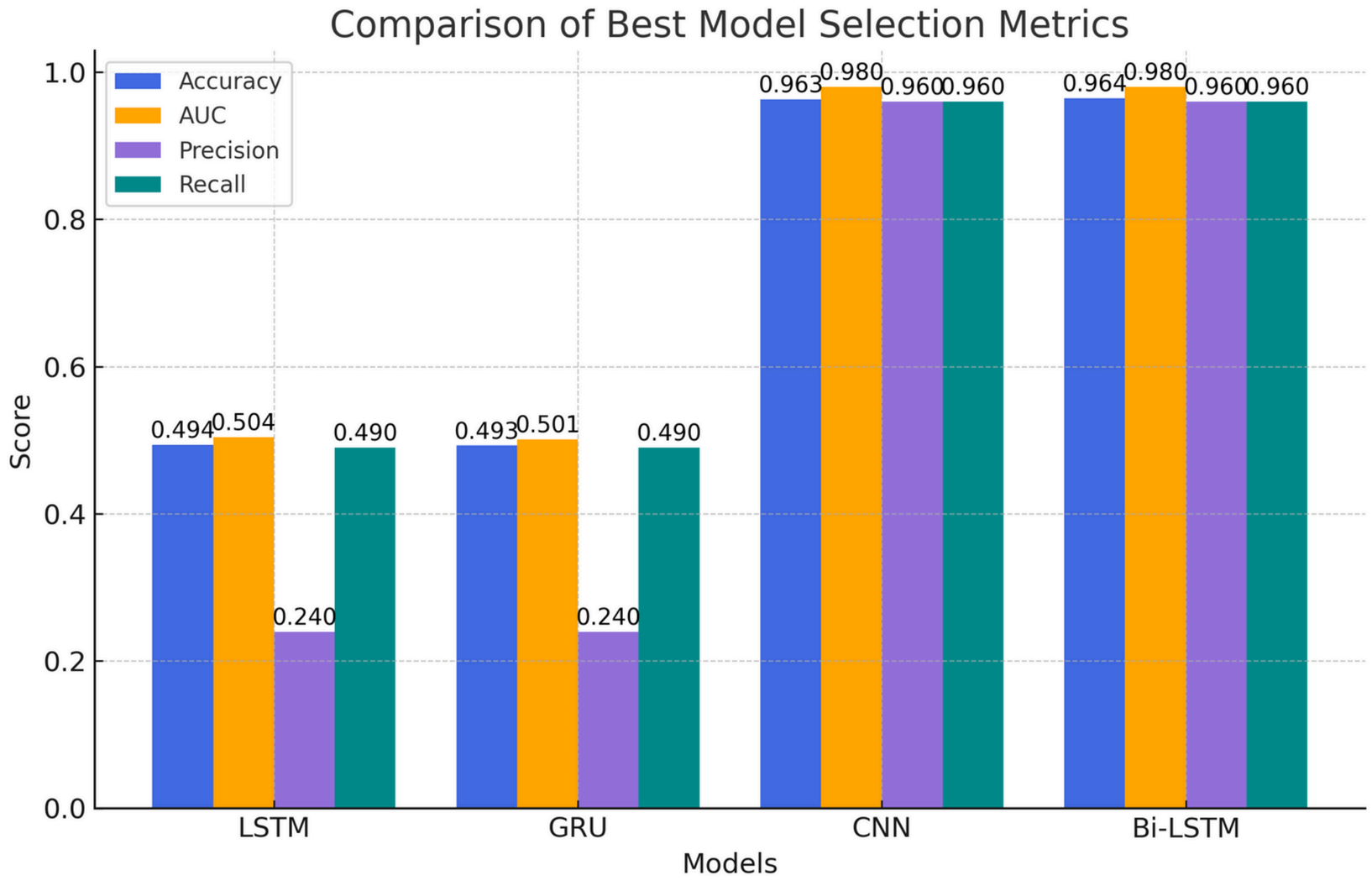
- Converts text into numerical form for model training.

Text Models Comparison

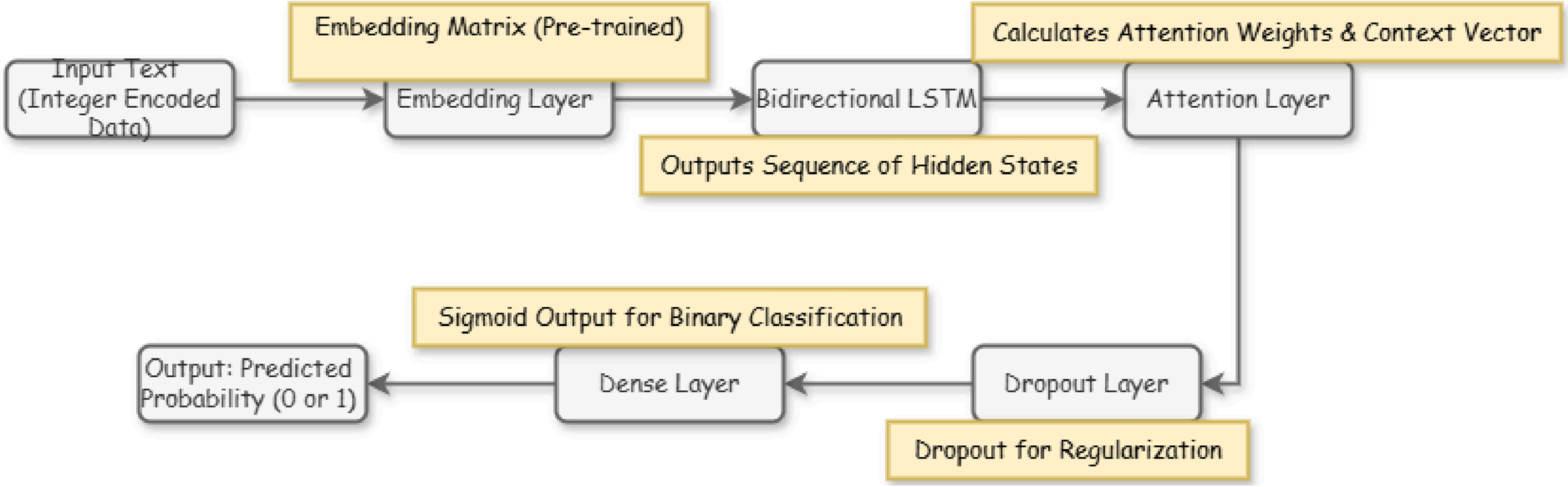
Model	Accuracy
Bidirectional LSTM	0.96
LSTM	0.50
CNN	0.95
GRU	0.50



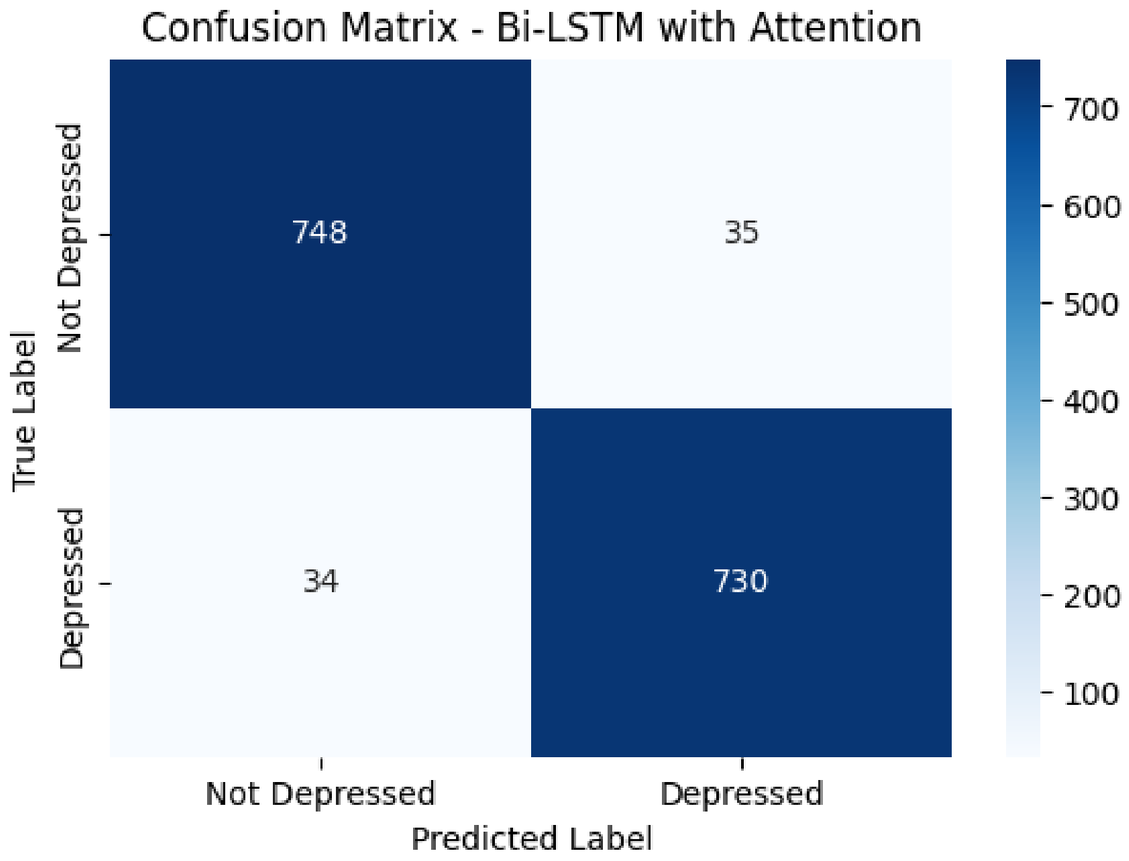
Text Model Selection

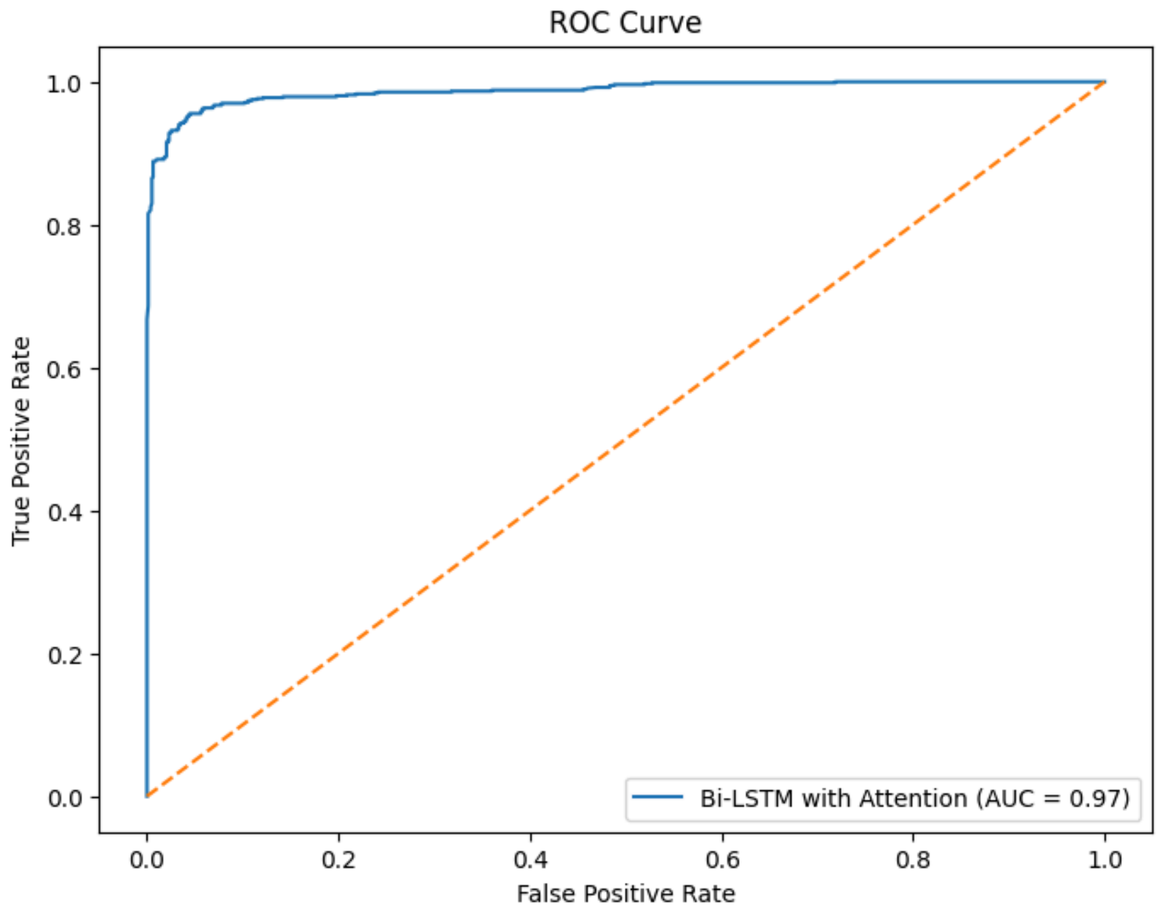


Text Model Architecture



Text Model Performance Evaluation





Text Model Performance Evaluation

```
Bi-LSTM with Attention:
Accuracy: 0.9625, AUC: 0.9764
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       783
           1       0.98      0.94      0.96       764

    accuracy                           0.96      1547
   macro avg       0.96      0.96      0.96      1547
weighted avg       0.96      0.96      0.96      1547

Confusion Matrix:
 [[770  13]
 [ 45 719]]
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
Text: I feel so down and hopeless every day. Nothing makes sense anymore.
Predicted Probability: 0.9961
Predicted Class: Depressed
--------------------------------------------------
Text: Life is awesome! I am feeling super happy and excited about my future.
Predicted Probability: 0.0008
Predicted Class: Not Depressed
--------------------------------------------------
Text: I have been feeling so lost and empty inside.
Predicted Probability: 0.0007
Predicted Class: Not Depressed
--------------------------------------------------
Text: I am okay, just going through some normal stress.
Predicted Probability: 0.0035
Predicted Class: Not Depressed
--------------------------------------------------
Text: My anxiety keeps getting worse. I don't know what to do.
Predicted Probability: 0.9880
Predicted Class: Depressed
--------------------------------------------------
```

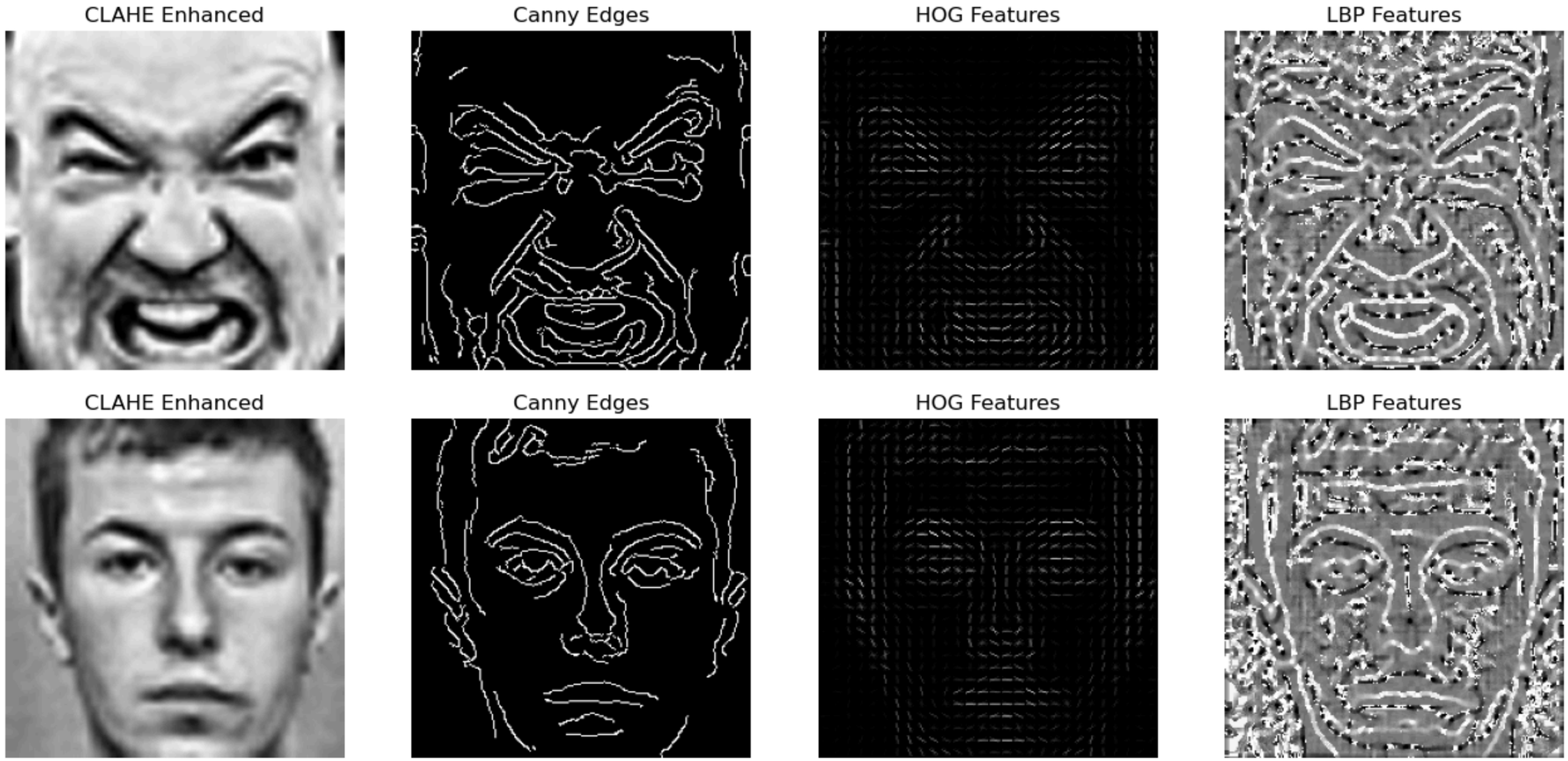
Image Preprocessing

Approaches Used	Why?
Resizing	To resize the image to a fixed size (224x224), and ensure consistency across all images for model input
Contrast Enhancement using CLAHE	To enhance visibility of features, especially in low-light or poor-contrast images.
Noise Reduction using Gaussian Blur	To reduce unwanted variations in pixel intensity while retaining important edges.
Grayscale Conversion	To convert the image to grayscale, reducing the number of channels (from RGB to a single intensity value), which simplifies the image for processing.

Feature Extraction

Approaches Used	Why?
Edge Detection using Canny Algorithm	Highlights edges and contours, making it easier for models to learn structural patterns
Histogram of Oriented Gradients (HOG)	Captures shape and texture information critical for facial expressions
Local Binary Pattern (LBP):	Captures fine-grained texture patterns

Feature Extraction Techniques Samples

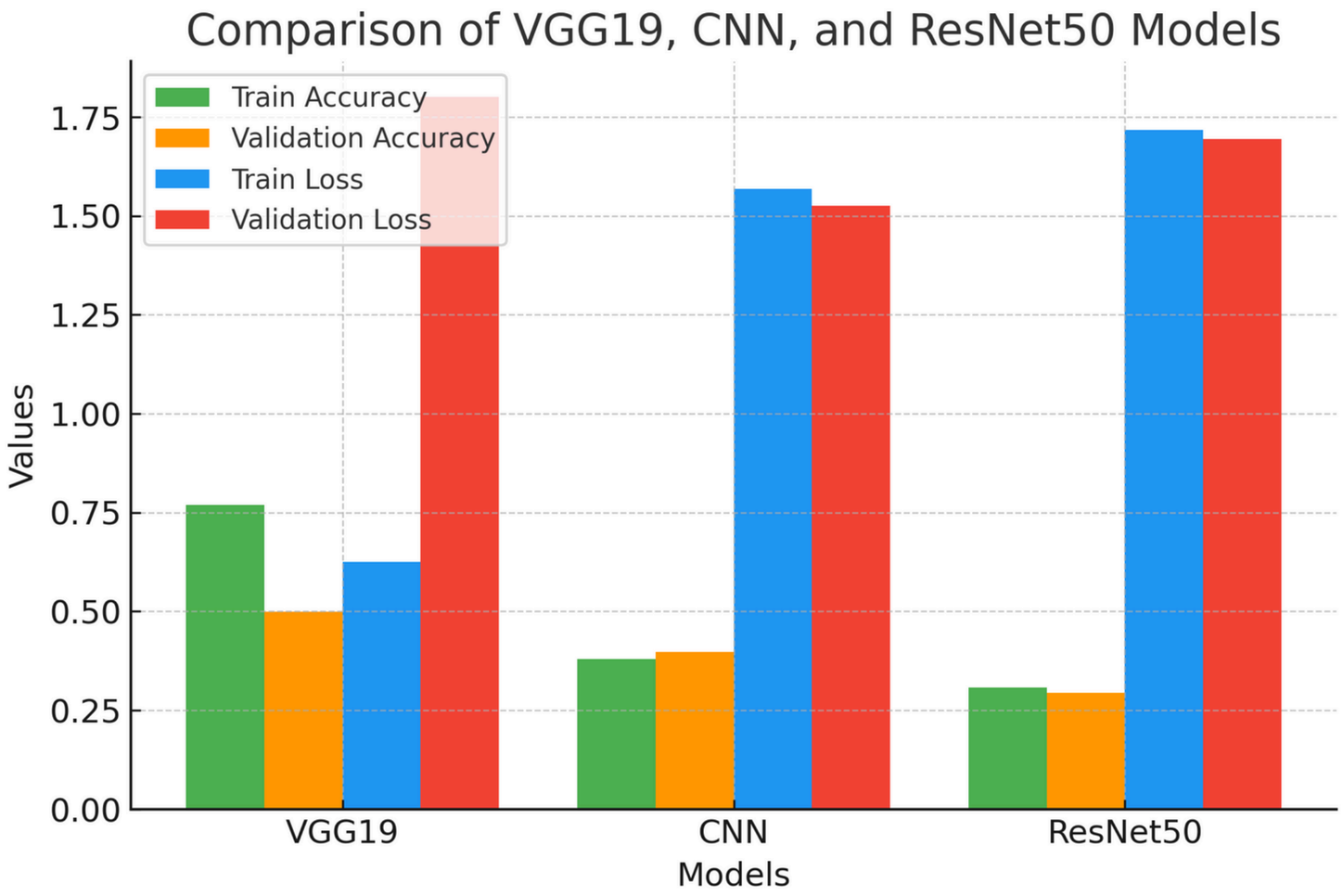


Data Augmentation

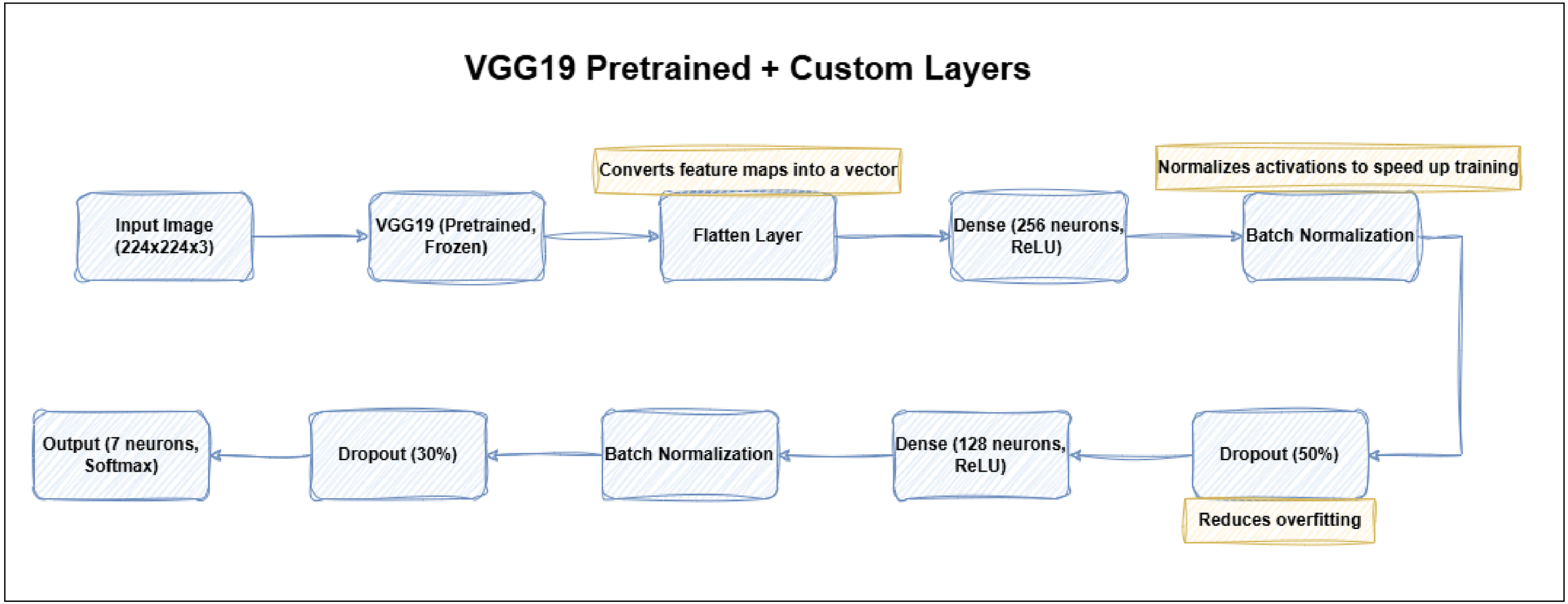
Used transformations which were rescale, rotation, zoom, and flipping to increase dataset variability and prevent overfitting.

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

# Define paths
BASE_DIR = '/content/drive/MyDrive/Colab Notebooks/Research/emotions/balanced_dataset'
TRAIN_DIR = BASE_DIR + '/train'
VALID_DIR = BASE_DIR + '/validation'
TEST_DIR = BASE_DIR + '/test'

# Define Image Augmentation for Training Data
train_datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode="nearest"
)

# Validation & Test Data Generator (No Augmentation)
val_test_datagen = ImageDataGenerator(rescale=1.0/255.0)

# Load images in batches
BATCH_SIZE = 128
```

Model Comparison



Model Creation



Model Architecture

Feature Extraction | **Fully Connected Layers**

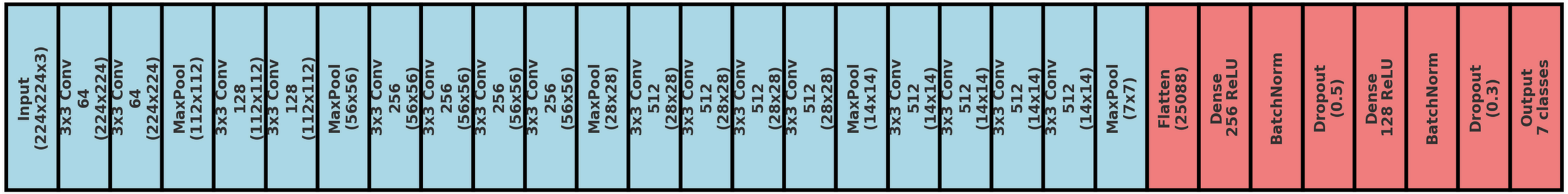
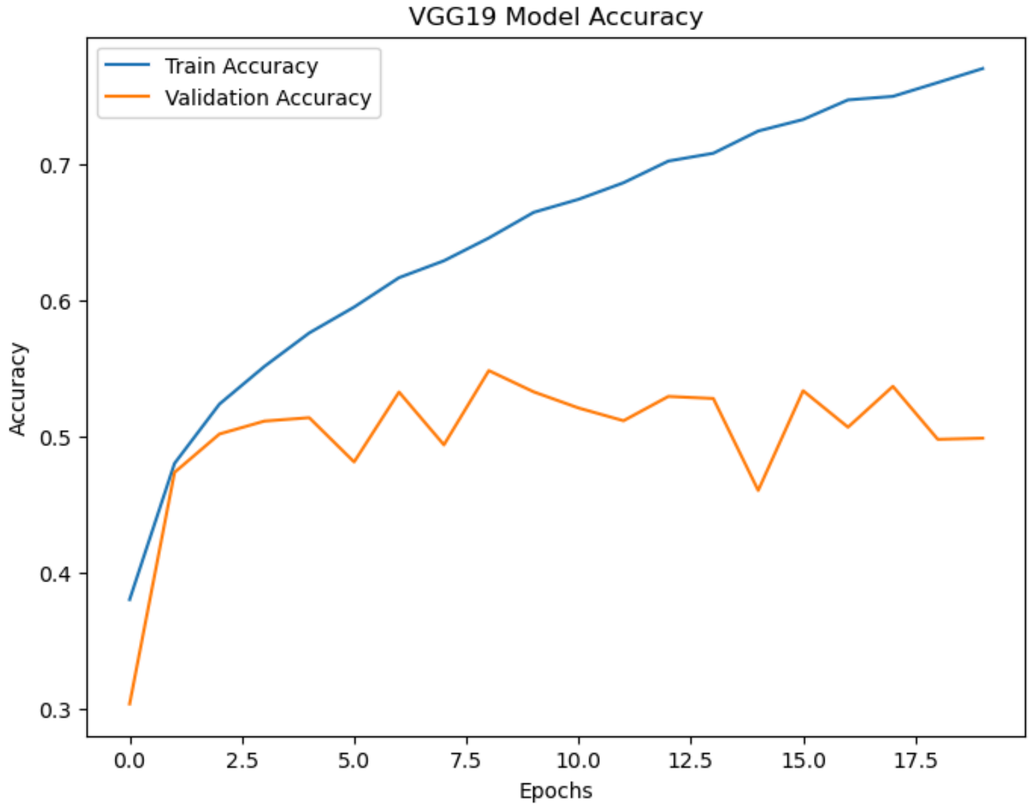


Image Model Performance & Evaluation



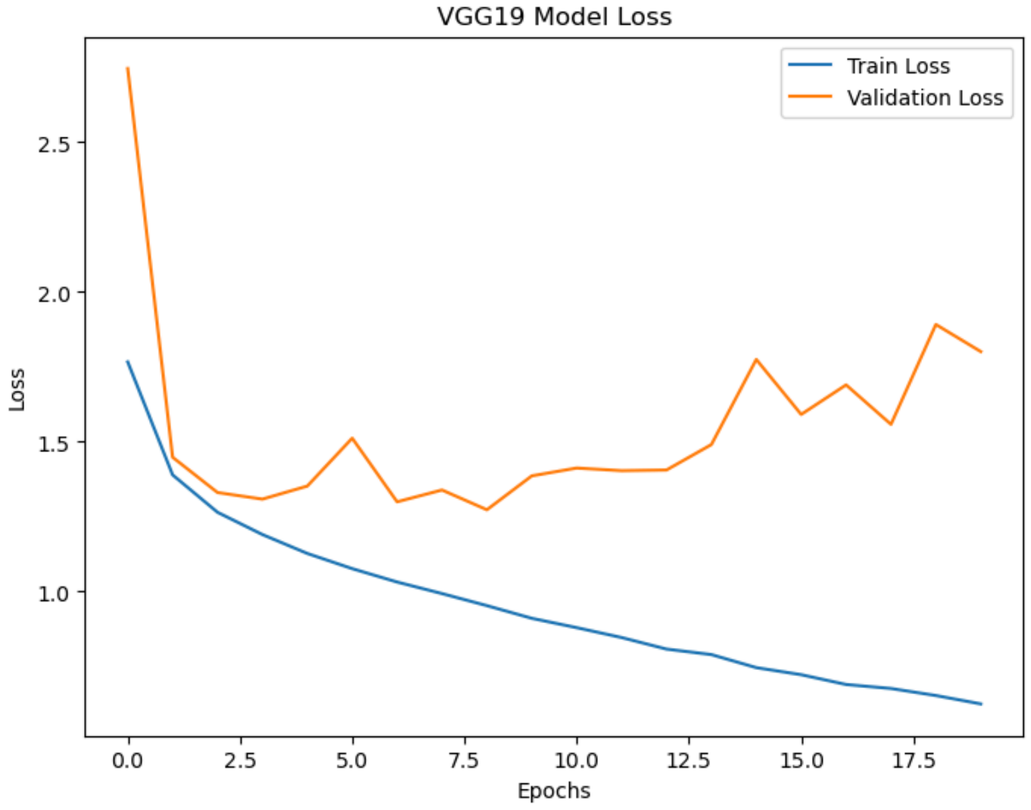


Image Model Performance & Evaluation

```python
# Step 10: Predict a Single Image
from tensorflow.keras.preprocessing.image import load_img, img_to_array

def predict_image(image_path):
    img = load_i (import) img_to_array: Any 224, 224))
    img_array = img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model_vgg19.predict(img_array)
    predicted_class = np.argmax(prediction)
    class_labels = list(test_data.class_indices.keys())

    print(f"Predicted Class: {class_labels[predicted_class]} with {np.max(prediction) * 100:.2f}% confidence")

# Example usage
predict_image("D:\\VS Code\\New folder\\train\\neutral\\im1.png")
```

```
1/1 [==============================] - 0s 165ms/step
Predicted Class: neutral with 98.28% confidence
```

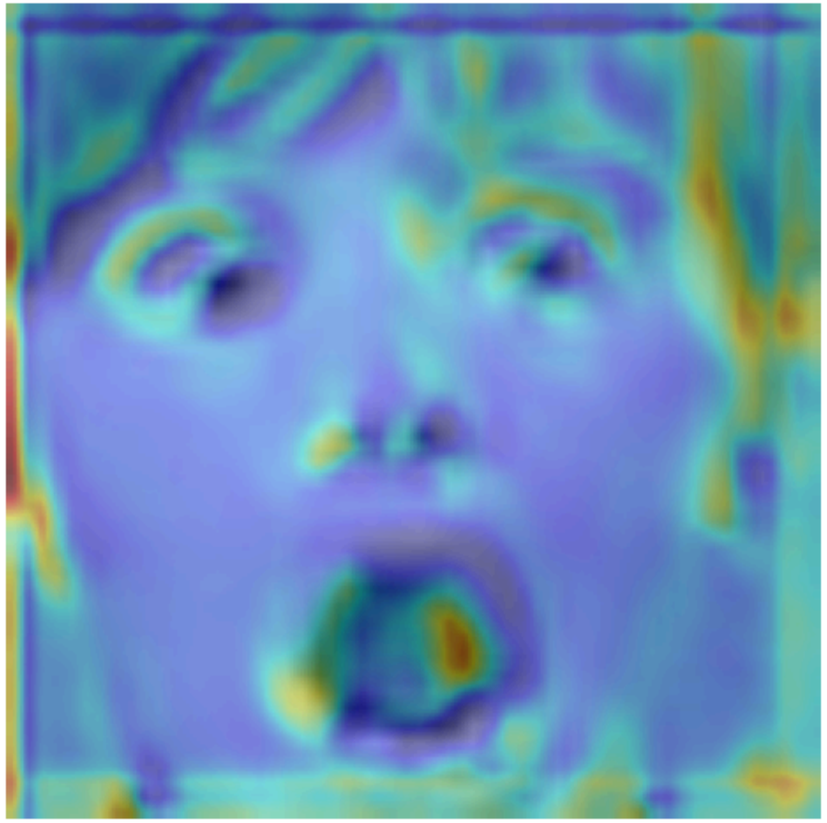
Explainable AI for Model Outputs

```python
# Load and preprocess image
image_path = "D:\\VS Code\\New folder\\test\\surprised\\im11.png"
input_image = preprocess_image(image_path)

# Get predictions
predictions = model.predict(input_image)
predicted_class = np.argmax(predictions)
confidence = np.max(predictions) * 100

# Print prediction result
print(f"Predicted Class: {class_labels[predicted_class]} with {confidence:.2f}% confidence")

# Generate Grad-CAM heatmap
last_conv_layer_name = "block3_conv4"  # Replace with the correct layer name in your model
heatmap = generate_gradcam_heatmap(model, input_image, last_conv_layer_name)

# Overlay heatmap on original image
overlay_heatmap(image_path, heatmap)

```

✓ 2.3s

```
1/1 [==============================] - 1s 514ms/step
Predicted Class: surprised with 99.22% confidence
```

Explainable AI - Grad-CAM



Explainable AI for Model Outputs

```python
# Step 10: Predict a Single Image
from tensorflow.keras.preprocessing.image import load_img, img_to_array

def predict_image(image_path):
    img = load_img(image_path, target_size=(224, 224))
    img_array = img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model_vgg19.predict(img_array)
    predicted_class = np.argmax(prediction)
    class_labels = list(test_data.class_indices.keys())

    print(f"Predicted Class: {class_labels[predicted_class]} with {np.max(pre

# Example usage
predict_image("D:\\VS Code\\New folder\\test\\fearful\\im0.png")
```

```
1/1 [==============================] - 0s 172ms/step
Predicted Class: fearful with 52.66% confidence
```

Explainable AI - Grad-CAM

Mutlimodal Fusion - Weighted Average Approach

Multimodal Fusion:

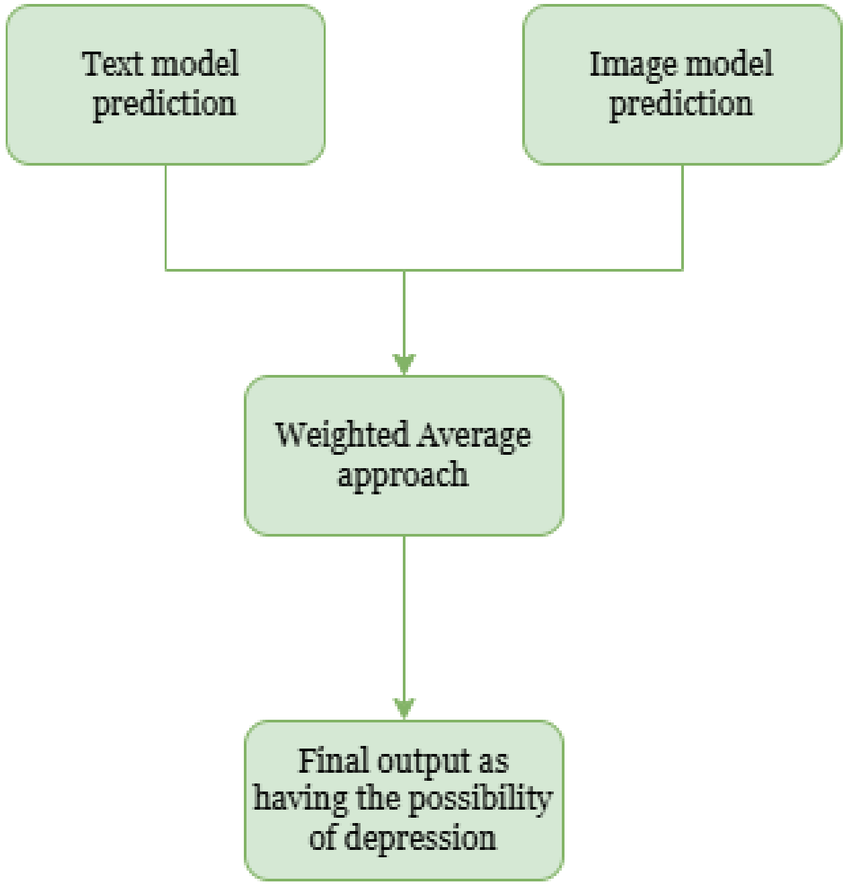
- To combine the text and image prediction probabilities.
- The weights for the text and image probabilities are adjustable (for now 0.5 for both).
- The formula used is:

$$\textit{Final Score} = (w\ \textit{text} \times \textit{Text Prediction}) + (w\ \textit{image} \times \textit{Image Prediction})$$

Final Output:

- The final prediction is made based on the fused probabilities (using weighted average).
- If the final probability exceeds 0.5, the result is classified as "Depressed," otherwise "Not Depressed."

Mutlimodal Fusion - Weighted Average Approach



Mutlimodal Fusion - Weighted Average Approach

```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")
```

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

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compi
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compi
WARNING:absl:Error in loading the saved optimizer state. As a result, your model is starting with a
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
Text Input: I feel very satisfying.
Prob of text:  0.010250252
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
Probabilities of emotions:  [0.02920195 0.00413914 0.05542067 0.83891743 0.04600881 0.014124
 0.01218805]
Image Path: /content/drive/MyDrive/Colab Notebooks/Research/emotions/test/happy/im0.png
Prob of image:  0.09874662
Final prob using weighted avg:  0.054498434
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