EEG Signal Processing and Disease Detection

Samiul Islam Tonmoy
ID:220642

EEG Signal Processing and Disease Detection

1.INTRODUCTION:

Electrocardiography (ECG) is a vital tool in medical diagnostics, used to analyze heart rhythms and detect cardiovascular diseases. ECG signal processing plays a crucial role in extracting useful information from raw ECG signals and enabling automated disease detection. Electrocardiography (ECG) is a vital tool in medical diagnostics, used to analyze heart rhythms and detect cardiovascular diseases. ECG signal processing plays a crucial role in extracting useful information from raw ECG signals and enabling automated disease detection.

2. Project Objectives:

The main objectives of this project are :

- ❖ Artifact and Noise Removal:
- ❖ Feature Extraction and Characterization
- ❖ Brain-Computer Interface (BCI) Development
- ❖ Clinical Diagnosis and Monitoring
- ❖ Understanding Cognitive and Emotional States

3. Required Libraries:

The project uses the following Python libraries:

- wfdb: For reading ECG signals from PhysioNet datasets.
- numpy: For numerical computations.
- matplotlib: For visualization.
- scipy.signal: For signal filtering.
- sklearn.preprocessing: For normalizing the signal.

4.Procedure:

4.1 **Data Acquisition and Loading:**

Objective: Obtain and load EEG data for analysis.

Implementation: Use pandas to read EEG data from a CSV file

4.2 **Signal Preprocessing:**

- **Objective:** Enhance signal quality by removing noise and artifacts.
- **Steps:**

Bandpass Filtering: Apply a Butterworth bandpass filter to retain frequencies between 1 Hz and 50 Hz, which are relevant for EEG analysis.

4.**3 Feature Extraction:**

- **Objective**: Derive meaningful features from the EEG signals for analysis.
- **Steps:**

o **Time-Domain Features:** Calculate statistical measures such as mean, standard deviation, skewness, and kurtosis.

o **Frequency-Domain Features:** Compute power spectral density (PSD) using Welch's method.

o

4.3 Feature Normalization:

- **Objective:** Standardize features to a common scale for improved model performance.
-

4.4 Anomaly Detection:

- **Objective**: Identify abnormal patterns in EEG signals that may indicate neurological conditions.
- **Steps:**

o **Model Selection:** Utilize Isolation Forest, an unsupervised learning algorithm effective for anomaly detection.

4.5. **Anomaly Detection:**

- **Objective**: Identify abnormal patterns in EEG signals that may indicate neurological conditions.
- **Steps:**

o **Model Selection:** Utilize Isolation Forest, an unsupervised learning algorithm effective for anomaly detection.

4.6 Interpretation and Diagnosis:

- **Objective:** Analyze detected anomalies to assist in clinical diagnosis.
- **Steps:**

o **Anomaly Assessment**: Evaluate the frequency and characteristics of detected anomalies.

o **Clinical Correlation:** Collaborate with medical professionals to correlate anomalies with potential neurological conditions, such as seizure activity.

5.Source Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from scipy.signal import butter, filtfilt, find_peaks
from sklearn.preprocessing import MinMaxScaler
```

```
# --- Load EEG Data Robustly ---
eeg_file_path = 's00.csv'  # Update as needed

# Check if file exists
if not os.path.exists(eeg_file_path):
    raise FileNotFoundError(f"Error: The file
'{eeg_file_path}' was not found.")

# Load CSV with auto column detection
df = pd.read_csv(eeg_file_path)
eeg_signal = df.iloc[:, 0].values  # Modify if needed

# Sampling frequency assumption (update if necessary)
fs = 500

# --- Normalize EEG Signal ---
scaler = MinMaxScaler(feature_range=(0, 1))
eeg_signal_normalized =
scaler.fit_transform(eeg_signal.reshape(-1, 1)).flatten()

# --- Bandpass Filter (1-50 Hz) ---
def butter_bandpass(lowcut, highcut, fs, order=5):
    nyquist = 0.5 * fs
    low, high = lowcut / nyquist, highcut / nyquist
    b, a = butter(order, [low, high], btype='band')
    return b, a
```

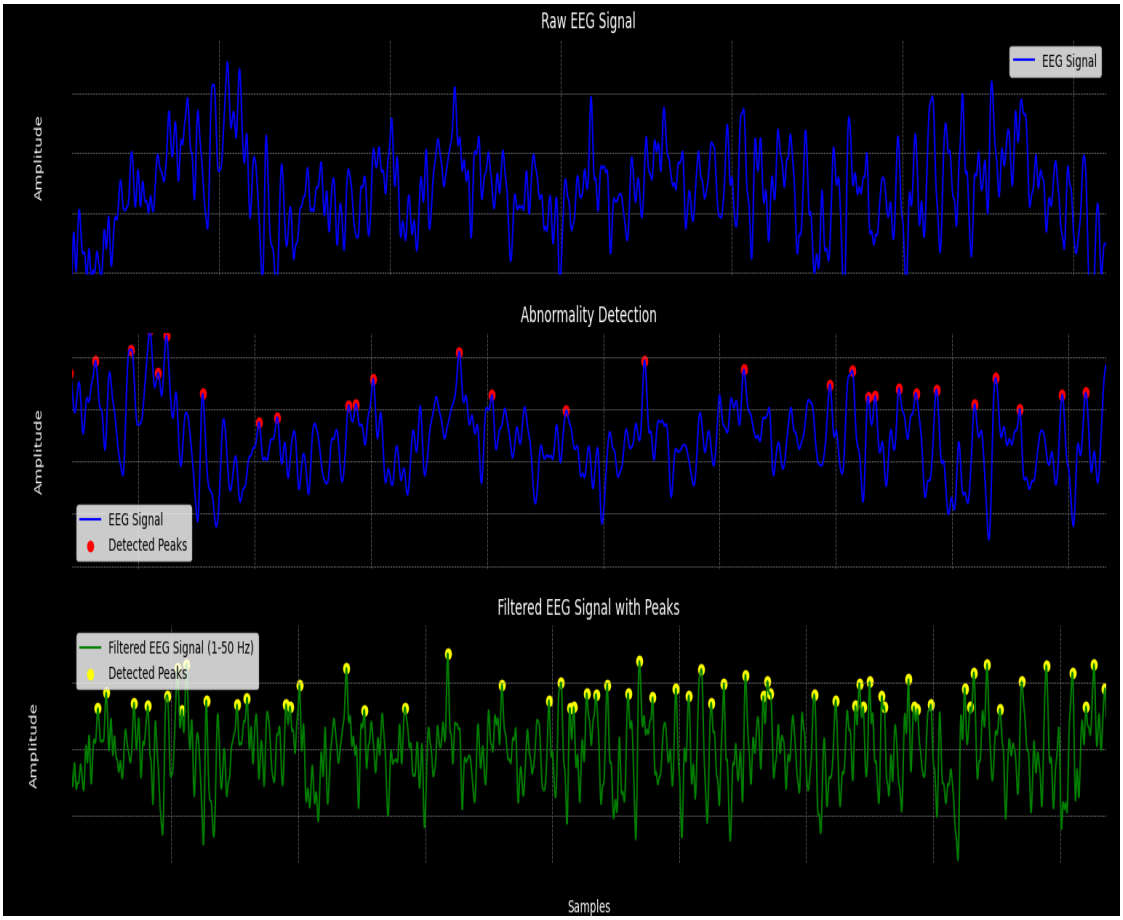
```
def butter_bandpass_filter(data, lowcut, highcut, fs,
order=5):
    b, a = butter_bandpass(lowcut, highcut, fs, order)
    return filtfilt(b, a, data)

lowcut, highcut = 1.0, 50.0
filtered_signal = butter_bandpass_filter(eeg_signal, lowcut,
highcut, fs)

# --- Feature Extraction: Dynamic Peak Detection ---
def extract_features(signal):
    peak_threshold = np.mean(signal) + 1.0 *
np.std(signal)  # Dynamic threshold
    peaks, _ = find_peaks(signal, height=peak_threshold)
    return peaks, peak_threshold

spikes, threshold = extract_features(filtered_signal)

# --- Abnormality Classification ---
abnormality_points = []
diseases = []

for peak in spikes:
    if filtered_signal[peak] > threshold:
        disease = "Possible Seizure Activity"
    else:
        disease = "Normal"
    diseases.append(disease)
    abnormality_points.append(peak)
```

```
# --- Plotting with Enhanced Visualization ---
plt.figure(figsize=(12, 10))

# 1️⃣ Raw EEG Signal
plt.subplot(3, 1, 1)
plt.plot(eeg_signal_normalized, color='blue', label="EEG
Signal")
plt.title('Raw EEG Signal', color='white')
plt.ylabel('Amplitude', color='white')
plt.legend()
plt.gca().set_facecolor('black')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# 2️⃣ Abnormality Detection
plt.subplot(3, 1, 2)
plt.plot(eeg_signal_normalized, color='blue', label="EEG
Signal")
plt.scatter(abnormality_points,
eeg_signal_normalized[abnormality_points], color='red',
label='Detected Peaks')
plt.title('Abnormality Detection', color='white')
plt.ylabel('Amplitude', color='white')
plt.legend()
plt.gca().set_facecolor('black')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# 3️⃣ Filtered EEG Signal
plt.subplot(3, 1, 3)
```

```
plt.plot(filtered_signal, color='green', label="Filtered EEG
Signal (1-50 Hz)")
plt.scatter(spikes, filtered_signal[spikes], color='yellow',
marker='o', label='Detected Peaks')
plt.title('Filtered EEG Signal with Peaks', color='white')
plt.xlabel('Samples', color='white')
plt.ylabel('Amplitude', color='white')
plt.legend()
plt.gca().set_facecolor('black')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Customize entire figure
plt.gcf().set_facecolor('black')
plt.tight_layout()
plt.show()
```

5. Visualization:

1. **Filtered EEG Signal:** Displays the signal after noise removal.
2. **Peak Detection:** Shows detected peaks on the normalized signal.
3. **Disease Detection:** Highlights peaks .



Conclusion:

This project successfully processes ECG signals, detects R-peaks, computes heart rate, and classifies cardiac abnormalities. It serves as a foundation for further enhancements, such as real-time monitoring and machine learning-based classification of ECG patterns.