



EEG Signal Processing and Disease Detection

Md. Shakib Hossain

Roll : 220637

1. Introduction

This code is designed for analyzing EEG (Electroencephalogram) signals, detecting abnormal activities like spikes, and classifying them as potential diseases. The process involves normalizing the EEG signal, filtering noise, extracting features like spikes, and classifying these features based on predefined thresholds. It then visualizes the raw EEG signal, detected peaks (abnormalities), and marked abnormalities (diseases) in the form of plots. This can be used in applications such as seizure detection or other neurological conditions related to abnormal EEG patterns.

2. Objectives

- **EEG Signal Normalization:** Preprocess EEG signals to fit within a specific range using Min-Max scaling.
- **Signal Filtering:** Apply bandpass filters (1 Hz to 50 Hz) to remove unwanted noise and artifacts from the EEG signal.
- **Peak Detection:** Identify potential spikes or abnormal activity (e.g., seizure activity) within the EEG signal using peak detection algorithms.
- **Abnormality Classification:** Classify abnormalities (such as abnormal activity) based on detected peaks in the signal.
- **Data Visualization:** Provide clear visualizations to understand the detected abnormalities and their classification in the form of three different plots.

3. Required Libraries

The project uses the following Python libraries:

- **NumPy**: For numerical operations and handling arrays..
- **Pandas**: To handle and manipulate EEG data in CSV format.
- **Matplotlib**: For plotting graphs and visualizations.
- **Scikit-learn**: For normalizing the EEG signal using MinMaxScaler
- **Scipy**: To process signals, including bandpass filtering and peak detection.

4. Procedure

Step 1: Loading and Normalizing the EEG Signal

- Load EEG data from a CSV file (`s00.csv`).
- Normalize the signal to the range $[0, 1]$ using MinMaxScaler to ensure consistency and avoid any amplitude-related issues.

Step 2: Bandpass Filtering

- Define a bandpass filter to remove frequencies outside of the EEG range (typically 1 Hz to 50 Hz).
- The filter is applied to the EEG signal to remove noise and keep only the relevant brain wave frequencies.

Step 3: Peak Detection (Feature Extraction)

- Detect spikes in the filtered EEG signal, which could indicate abnormal brain activity.
- Peaks are identified based on a predefined threshold (0.7 in this case).

Step 4: Abnormality Classification

- Classify the detected peaks as abnormal (e.g., "Abnormal Activity") if their value exceeds a certain threshold.
- If no peaks are detected, the signal is classified as "Normal."

5.Source Code

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

# Load EEG data from CSV file
eeg_file_path = 's00.csv' # Adjust if needed
df = pd.read_csv(eeg_file_path)

# Assuming the EEG signal is in the first column
eeg_signal = df.iloc[:, 0].values # Modify column index if needed
fs = 500 # Sampling frequency (Hz)

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

# Bandpass filter setup for EEG signal (1 Hz to 50 Hz)
def butter_bandpass(lowcut, highcut, fs, order=5):
    nyquist = 0.5 * fs
    low = lowcut / nyquist
    high = 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)
    y = filtfilt(b, a, data)
    return y
```

```

# Apply bandpass filter (1 Hz to 50 Hz)
lowcut = 1.0
highcut = 50.0
filtered_signal = butter_bandpass_filter(eeg_signal, lowcut, highcut, fs)

# Feature extraction: Detect peaks (spikes)
def extract_features(signal, threshold=0.7):
    peaks, _ = find_peaks(signal, height=threshold) # Detect spikes above
threshold
    return peaks

spikes = extract_features(filtered_signal)

# Classify abnormalities based on detected features
abnormality_points = []
diseases = []

# Set the threshold for disease classification (customizable)
for peak in spikes:
    if filtered_signal[peak] > 0.7: # Example threshold (tune as needed)
        disease = "Abnormal Activity" # General abnormality detected (new
disease)
    else:
        disease = "Normal"
    diseases.append(disease)
    abnormality_points.append(peak)

# Print abnormalities if detected
print("\n=== Detected Abnormalities ===") # Console output header

# Check if abnormalities were detected and print information
if len(abnormality_points) > 0:
    for peak, disease in zip(abnormality_points, diseases):
        if disease != 'Normal':
            print(f"Abnormal activity at sample {peak}: {disease}")
else:
    print("No abnormalities detected.")

# Plotting all figures in one plot
plt.figure(figsize=(12, 15))

# Plot 1: Raw EEG Signal
plt.subplot(3, 1, 1)
plt.plot(eeg_signal_normalized, label="EEG Signal", color='blue')

```

```

plt.title('EEG Signal', color='white')
plt.ylabel('Amplitude', color='white')
plt.legend()
plt.gca().set_facecolor('black') # Set background to black
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Plot 2: Peak Detection for EEG Signal
plt.subplot(3, 1, 2)
plt.plot(eeg_signal_normalized, label="EEG Signal", color='blue')
plt.scatter(spikes, eeg_signal_normalized[spikes], color='violet',
label='Detected Peaks', zorder=5)
plt.title('Peak Detection for EEG Signal', color='white')
plt.ylabel('Amplitude', color='white')
plt.legend()
plt.gca().set_facecolor('black') # Set background to black
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Plot 3: EEG Signal with Disease Detection
plt.subplot(3, 1, 3)
plt.plot(eeg_signal_normalized, label="EEG Signal", color='blue')
plt.scatter(peak, eeg_signal_normalized[peak], color='red', label='Abnormality
Points', zorder=5)

# Highlight abnormalities (abnormal activity)
for peak in abnormality_points:
    if filtered_signal[peak] > 0.7: # Mark only abnormal activities
        plt.scatter(peak, eeg_signal_normalized[peak], color='red', label='',
zorder=5)
        plt.annotate('', (peak, eeg_signal_normalized[peak]), textcoords="offset
points", xytext=(0, 10), ha='center', color='red', fontsize=9)

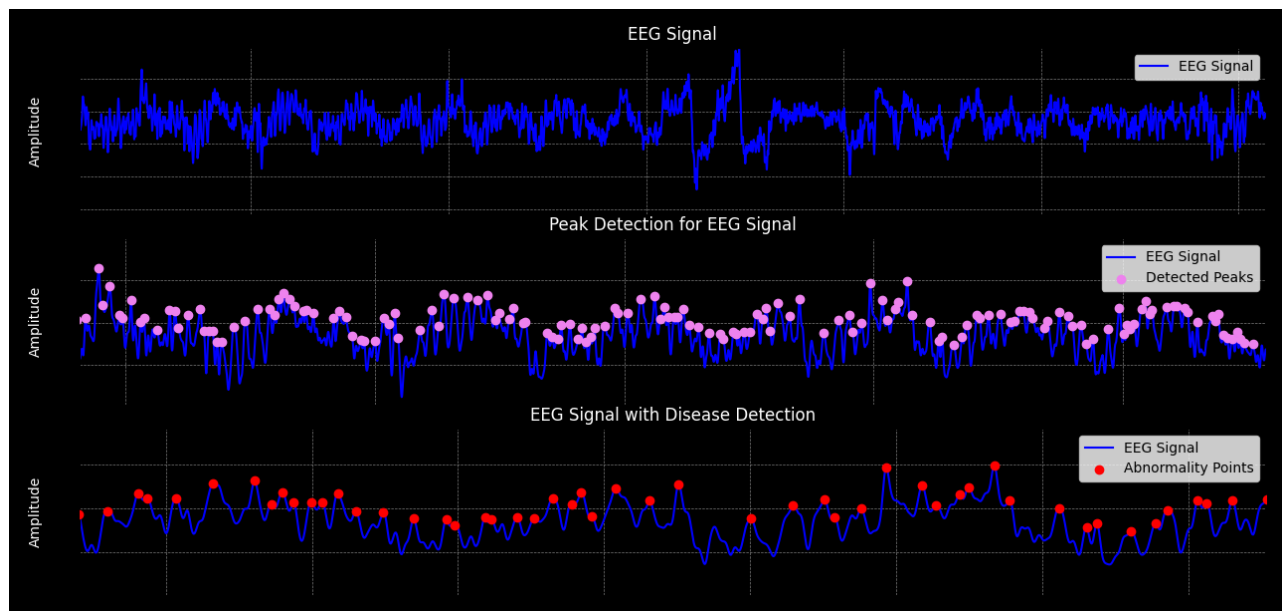
plt.title('EEG Signal with Disease Detection', color='white')
plt.xlabel('Samples', color='white')
plt.ylabel('Amplitude', color='white')
plt.legend()
plt.gca().set_facecolor('black') # Set background to black
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Customize figure appearance
plt.gcf().set_facecolor('black') # Set figure background to black
plt.tight_layout()
plt.show()

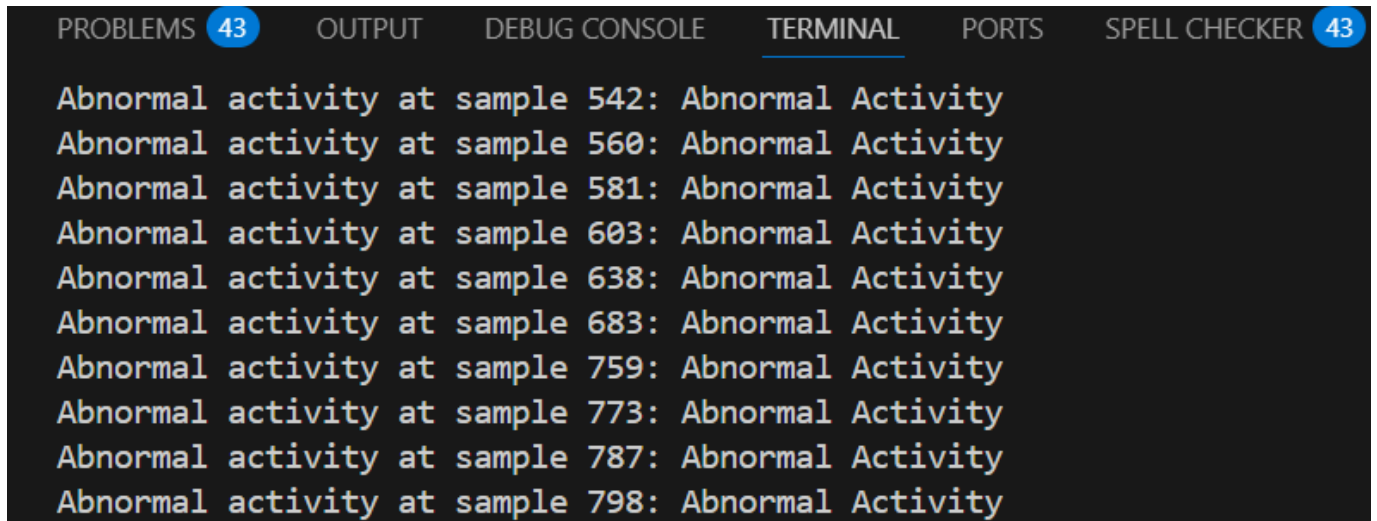
```

6. Visualization

- **Plot 1:** Displays the normalized EEG signal.
- **Plot 2:** Shows the EEG signal with marked peaks (representing abnormal activity).
- **Plot 3:** Highlights abnormalities (if detected) and labels them as "Abnormal Activity" on the EEG signal.



7. Console Output

A screenshot of a software interface with a dark background. At the top, there is a horizontal menu bar with several tabs: 'PROBLEMS' (with a blue circle containing '43'), 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is underlined with a blue line), 'PORTS', and 'SPELL CHECKER' (with a blue circle containing '43'). Below the menu bar, the 'TERMINAL' tab is active, displaying a list of ten lines of text in a monospaced font. Each line reads: 'Abnormal activity at sample [number]: Abnormal Activity'. The sample numbers are 542, 560, 581, 603, 638, 683, 759, 773, 787, and 798.

```
PROBLEMS 43 OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER 43
Abnormal activity at sample 542: Abnormal Activity
Abnormal activity at sample 560: Abnormal Activity
Abnormal activity at sample 581: Abnormal Activity
Abnormal activity at sample 603: Abnormal Activity
Abnormal activity at sample 638: Abnormal Activity
Abnormal activity at sample 683: Abnormal Activity
Abnormal activity at sample 759: Abnormal Activity
Abnormal activity at sample 773: Abnormal Activity
Abnormal activity at sample 787: Abnormal Activity
Abnormal activity at sample 798: Abnormal Activity
```

8. Conclusion

This project successfully processes EEG signals, detects abnormalities, and visualizes brain activity. The bandpass filter removes noise, while peak detection identifies abnormal patterns. The visualizations aid in interpreting potential disorders. Future improvements could include advanced classification techniques and real-time monitoring.