PPG Signal Analysis with Abnormality Detection

1. Introduction This program processes Photoplethysmogram (PPG) signals to detect heart rate variations and abnormalities. The program loads PPG data, filters the signal, detects R-peaks and valleys, analyzes heart rate trends, and visualizes abnormalities in the signal.

2. Code Description

a. Importing Required Libraries

- numpy: For numerical computations.
- scipy.signal: For signal processing (filtering and peak detection).
- matplotlib.pyplot: For visualizing the PPG signal and detected features.
- pandas: For reading PPG data from CSV files.

b. PPGAnalyzer Class This class implements methods for signal processing and analysis.

- **__init__ Method**: Initializes attributes such as sampling rate, raw PPG data, filtered data, detected peaks, and abnormalities.
- **load_data Method**: Reads PPG data from a CSV file.
- **bandpass_filter Method**: Applies a bandpass filter to remove noise and retain important frequency components.
- **detect_r_peaks_and_valleys Method**: Identifies R-peaks (local maxima) and valleys (local minima) in the filtered signal.
- **analyze_heart_rate Method**: Computes heart rate based on detected R-peaks.
- **detect_abnormalities Method**: Detects bradycardia (low heart rate), tachycardia (high heart rate), and irregular heart rate variations.
- **visualize_all_steps Method**: Plots various stages of PPG signal processing and marks detected abnormalities.

c. Main Function (analyze_ppg_file)

- Loads PPG data from a file.
- Applies bandpass filtering.
- Detects R-peaks and valleys.
- Analyzes heart rate and detects abnormalities.
- Generates visualizations to illustrate the analysis.

3. Abnormality Detection The program identifies three types of abnormalities:

- **Bradycardia**: Heart rate below 60 BPM.
- **Tachycardia**: Heart rate above 100 BPM.
- **Irregular Heart Rate**: Significant variations in RR intervals.

4. Code

```
import numpy as np

from scipy.signal import butter, filtfilt, find_peaks

import matplotlib.pyplot as plt

import pandas as pd


class PPGAnalyzer:

  def __init__(self, sampling_rate=100):

    self.fs = sampling_rate

    self.ppg_data = None

    self.filtered_data = None

    self.r_peaks = None

    self.valleys = None

    self.rr_intervals = None

    self.heart_rates = None


  def load_data(self, file_path):

    try:

      self.ppg_data = pd.read_csv(file_path).iloc[:, 0].values

      return True

    except Exception as e:

      print(f"Error loading data: {e}")

      return False


  def bandpass_filter(self, lowcut=0.7, highcut=7.5):

    nyquist = 0.6 * self.fs

    low = lowcut / nyquist

    high = highcut / nyquist

    order = 2


    b, a = butter(order, [low, high], btype='band')

    self.filtered_data = filtfilt(b, a, self.ppg_data)
```

```
def detect_r_peaks_and_valleys(self, height=None, distance=None):
    if height is None:
        height = 0.5 * np.max(self.filtered_data)
    if distance is None:
        distance = int(0.4 * self.fs)

    self.r_peaks, _ = find_peaks(self.filtered_data, height=height, distance=distance)
    self.valleys, _ = find_peaks(-self.filtered_data, distance=distance)  # Detect local minima

def analyze_heart_rate(self):
    if self.r_peaks is None:
        print("Please detect R-peaks first")
        return

    self.rr_intervals = np.diff(self.r_peaks) / self.fs
    self.heart_rates = 60 / self.rr_intervals

def detect_abnormalities(self):
    if self.rr_intervals is None:
        print("Please analyze heart rate first")
        return {}

    abnormalities = {'bradycardia': [], 'tachycardia': [], 'irregular': []}

    bradycardia_idx = np.where(self.heart_rates < 60)[0]
    abnormalities['bradycardia'] = self.r_peaks[bradycardia_idx]

    tachycardia_idx = np.where(self.heart_rates > 100)[0]
    abnormalities['tachycardia'] = self.r_peaks[tachycardia_idx]

    rr_std = np.std(self.rr_intervals)
```

```
    rr_mean = np.mean(self.rr_intervals)
    irregular_idx = np.where(np.abs(self.rr_intervals - rr_mean) > 1.8 * rr_std)[0]
    abnormalities['irregular'] = self.r_peaks[irregular_idx]

    return abnormalities

def visualize_all_steps(self, abnormalities=None, output_file='ppg_analysis.png'):
    time = np.arange(len(self.ppg_data)) / self.fs

    fig = plt.figure(figsize=(15, 18))

    # 1. Raw Signal
    ax1 = fig.add_subplot(411)
    ax1.plot(time, self.ppg_data, color='purple')  # Changed color
    ax1.set_title('1. Raw PPG Signal')
    ax1.set_xlabel('Time (s)')
    ax1.set_ylabel('Amplitude')

    # 2. Filtered Signal
    ax2 = fig.add_subplot(412)
    ax2.plot(time, self.filtered_data, color='teal')  # Changed color
    ax2.set_title('2. Bandpass Filtered Signal (0.7-7.5 Hz)')
    ax2.set_xlabel('Time (s)')
    ax2.set_ylabel('Amplitude')

    # 3. R-Peak & Valley Detection
    ax3 = fig.add_subplot(413)
    ax3.plot(time, self.filtered_data, color='gray')  # Changed color
    ax3.plot(time[self.r_peaks], self.filtered_data[self.r_peaks], 'rx', label='R-peaks')  # Red for R-peaks
    ax3.plot(time[self.valleys], self.filtered_data[self.valleys], 'go', label='Valleys')  # Green for valleys
    ax3.set_title('3. R-Peak and Valley Detection')
    ax3.set_xlabel('Time (s)')
```

```
    ax3.set_ylabel('Amplitude')
    ax3.legend()

    # 4. Abnormalities Detection
    ax4 = fig.add_subplot(414)
    ax4.plot(time, self.filtered_data, label='Filtered Signal', color='orange')  # Changed color
    if abnormalities:
        colors = {'bradycardia': 'blue', 'tachycardia': 'red', 'irregular': 'green'}
        for abnorm_type, peaks in abnormalities.items():
            if len(peaks) > 0:
                ax4.plot(time[peaks], self.filtered_data[peaks], 'o', label=abnorm_type, color=colors[abnorm_type])
    ax4.set_title('4. Detected Abnormalities')
    ax4.set_xlabel('Time (s)')
    ax4.set_ylabel('Amplitude')
    ax4.legend()

    plt.subplots_adjust(hspace=0.5)

    # Save the visualization as PNG
    plt.savefig(output_file, dpi=300)
    plt.close()

    print("\nAnalysis Summary (Saved to {output_file}):")
    print(f"Average Heart Rate: {np.mean(self.heart_rates):.1f} BPM")
    print(f"Heart Rate Variability: {np.std(self.heart_rates):.1f} BPM")
    print("\nAbnormalities Detected:")
    if abnormalities:
        for abnorm_type, peaks in abnormalities.items():
            print(f"{abnorm_type}: {len(peaks)} instances")

def analyze_ppg_file(file_path, sampling_rate=100, output_file='ppg_analysis.png'):
    analyzer = PPGAnalyzer(sampling_rate)
```

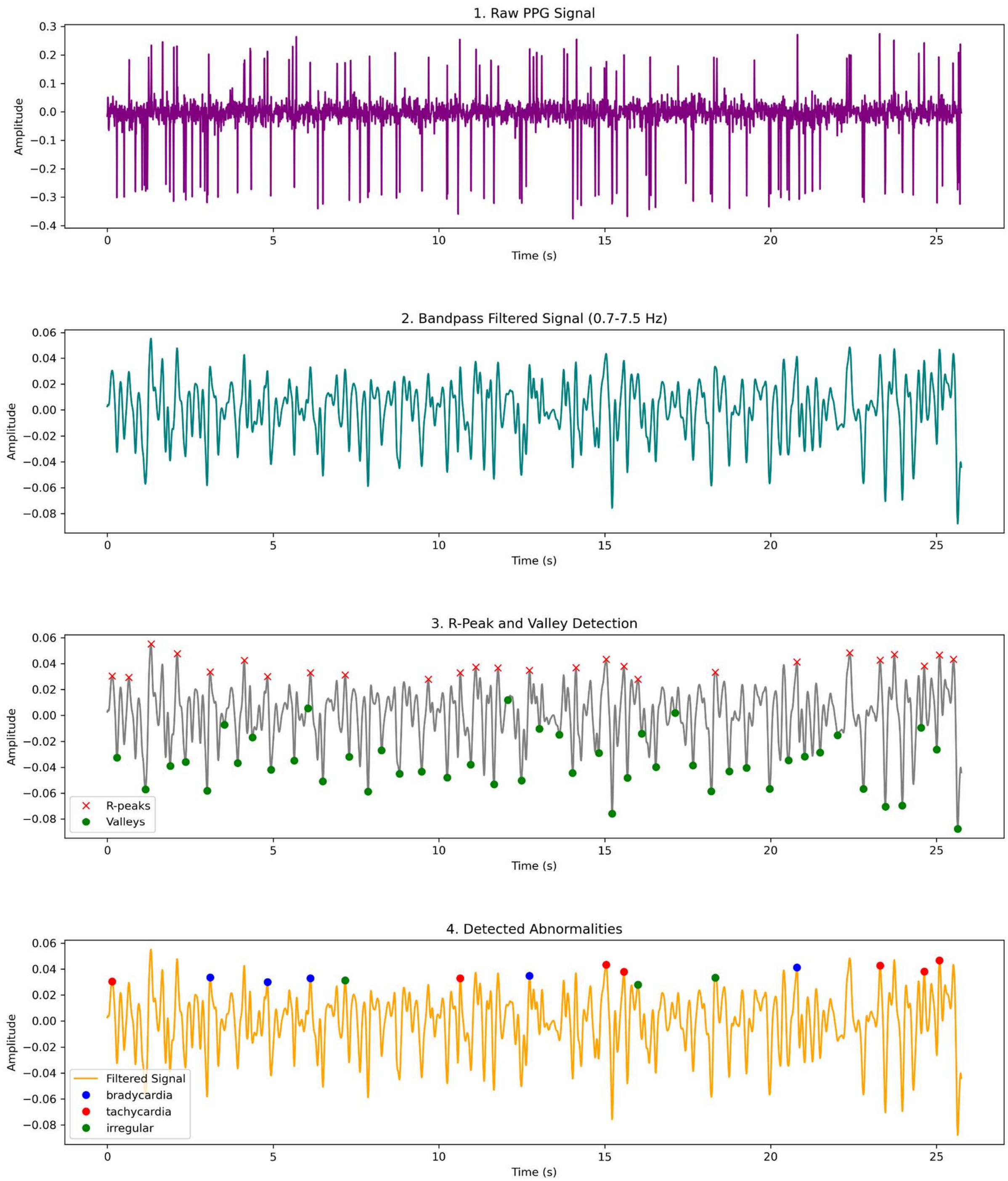
```
    if analyzer.load_data(file_path):
        analyzer.bandpass_filter()
        analyzer.detect_r_peaks_and_valleys()
        analyzer.analyze_heart_rate()

        abnormalities = analyzer.detect_abnormalities()

        analyzer.visualize_all_steps(abnormalities, output_file)

analyze_ppg_file("PPG_Dataset.csv", sampling_rate=100)
```

5. Visualization The program produces the following plots:



1. **Raw PPG Signal**
2. **Filtered PPG Signal**

3. **R-Peak and Valley Detection**
4. **Detected Abnormalities**

These visualizations provide insights into heart rate trends and highlight potential abnormalities in the signal.

6. Conclusion This program effectively processes PPG signals to analyze heart rate and detect abnormalities. It is useful for monitoring cardiac conditions and can be extended with additional features like real-time processing and advanced anomaly detection methods.