**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:A screenshot of a computer screen

AI-generated content may be incorrect.

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.