Experiment Name:

Analysis and Detection of Abnormalities in PPG Signals Using Statistical Features and Peak Detection.

Objectives:

1. To analyze photoplethysmography (PPG) signals from a csv file by calculating statistical features such as mean, median, mode, and standard deviation.
2. To detect abnormalities in PPG signals using peak detection techniques.
3. To visualize PPG signals and highlight detected abnormal peaks.
4. To save the results of abnormal peak detection for further analysis.

Theory:

Photoplethysmography (PPG) is a non-invasive optical technique used to measure blood volume changes in the microvascular bed of tissue. PPG signals are widely used in healthcare for monitoring vital signs, such as heart rate and oxygen saturation. However, these signals may contain abnormalities due to physiological variations or artifacts such as motion or noise.

In this experiment:

- Statistical features are calculated for each PPG signal to understand its distribution and variability. This includes:
    - **Mean**: Average value of the signal, indicating the baseline level.
    - **Median**: Middle value, showing the central tendency.
    - **Mode**: Most frequent value, reflecting repetitive patterns.
    - **Standard Deviation**: Measure of variability or dispersion.
- Abnormalities are detected by identifying peaks that deviate significantly from the normal pattern. A peak detection algorithm is

used with a threshold based on the mean and standard deviation of each PPG signal.

- The detected abnormal peaks are visualized on the PPG signal graphs for better understanding and interpretation.
- The results, including the time and indices of abnormal peaks, are saved for further analysis or medical evaluation.

Source Code with Output :

```
# Step 1: Import necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import find_peaks


# Step 2: Load the dataset

data = pd.read_csv('E:\Downloads\s10_run.csv')
data.head()
# Extract relevant columns
time = pd.to_datetime(data['time'])
ecg = data['ecg']
pleth = data[['pleth_1', 'pleth_2', 'pleth_3', 'pleth_4', 'pleth_5', 'pleth_6']]
# Display the extracted columns
print("Time:", time.head())
print("ECG:", ecg.head())
print("Pleth:", pleth.head())


# Step 2 Output :
```

```
Time:
 0 2021-01-01 11:22:48.305804
1 2021-01-01 11:22:48.307804
2 2021-01-01 11:22:48.309804
3 2021-01-01 11:22:48.311804
4 2021-01-01 11:22:48.313804
Name: time, dtype: datetime64[ns]
ECG:
0 33664
1 33866
2 34155
3 34366
4 34538
Name: ecg, dtype: int64
Pleth: pleth_1 pleth_2 pleth_3 pleth_4 pleth_5 pleth_6
0 65589 69333 3164 90553 103093 5652
1 65589 69333 3164 90558 103077 5647
2 65595 69334 3181 90544 103098 5660
3 65591 69349 3175 90544 103098 5660
4 65583 69343 3186 90554 103093 5651
```

```python
# Step 3: Calculating statistical data

# Function to calculate statistical features for PPG signals
def calculate_statistical_features(pleth_signal):
    features = {}
    for i in range(pleth_signal.shape[1]):
        signal = pleth_signal.iloc[:, i]
        features[f'pleth_{i + 1}'] = {
            'mean': np.mean(signal),
            'median': np.median(signal),
            'mode': signal.mode()[0],
```

```
            'std_dev': np.std(signal)
        }
    return pd.DataFrame(features)

# Calculate features
statistical_features = calculate_statistical_features(pleth)
# Display the calculated features
statistical_features
```

Step 3 Output :

	pleth_1	pleth_2	pleth_3	pleth_4	pleth_5	pleth_6
mean	64789.504940	66658.713972	3371.902358	90909.679105	102872.013101	5930.212096
median	64766.000000	66587.000000	3375.000000	90938.000000	102891.000000	5939.000000
mode	64583.000000	65920.000000	3411.000000	90970.000000	103221.000000	5913.000000
std_dev	400.341738	1348.538883	96.272787	169.333787	415.090266	151.723615

Step 4: Define and Detect abnormalities from PPG Signal

```
# Function to detect abnormalities in PPG signal
def detect_abnormalities(pleth_signal):
    abnormal_indices = []
    for i in range(pleth_signal.shape[1]):
        # Find peaks in the PPG signal with a threshold
        signal = pleth_signal.iloc[:, i]
                peaks, _ = find_peaks(signal, height=np.mean(signal) + 2 *
                                      np.std(signal))
```

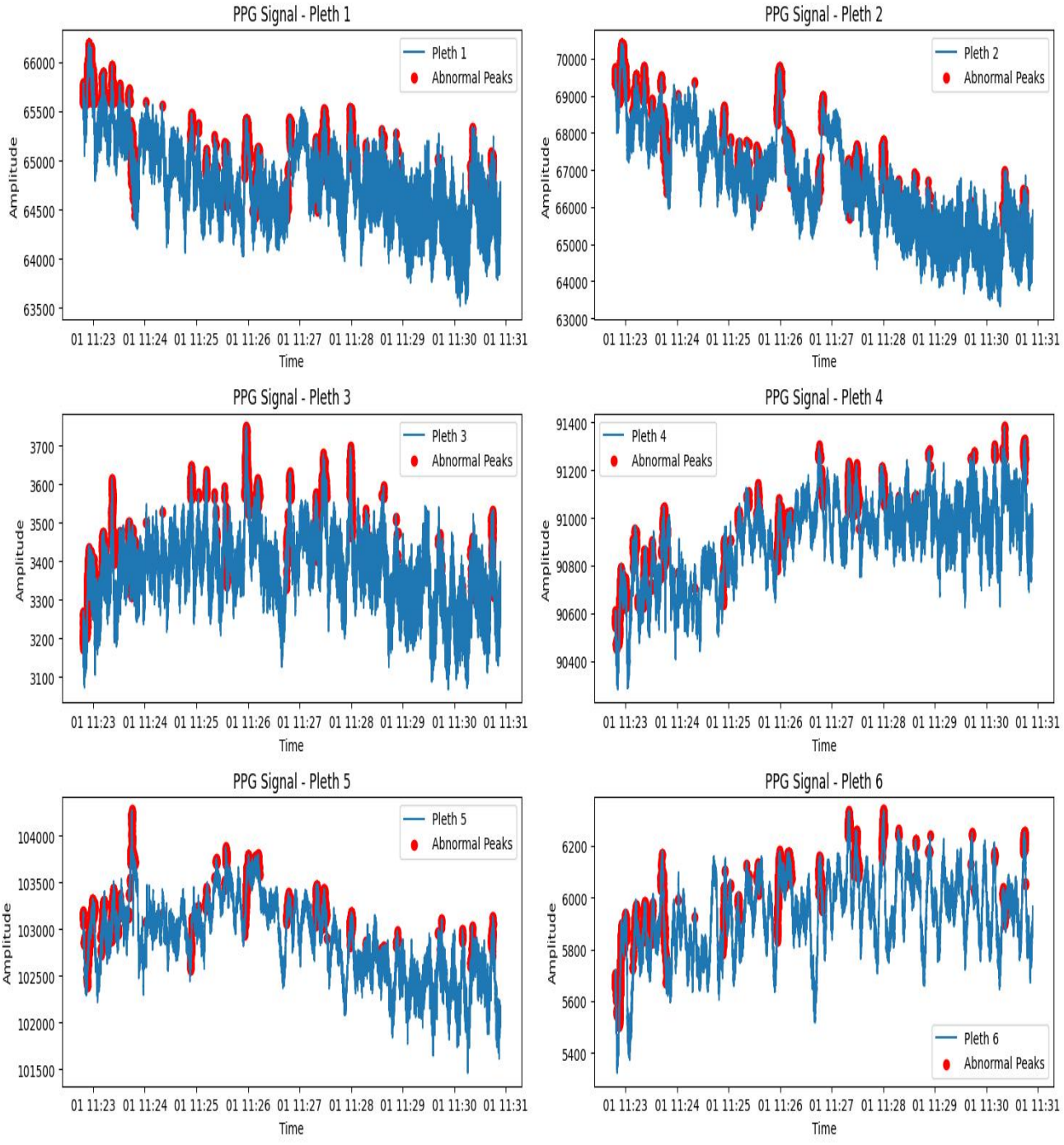
```
        abnormal_indices.extend(peaks)
    return abnormal_indices

# Detect abnormalities in the PPG signals
abnormal_peaks_indices = detect_abnormalities(pleth)

# Display detected abnormal peak indices
print("Detected Abnormal Peak Indices:", abnormal_peaks_indices)
```

```
# Step 4 Output :(Some Detected Abnormal Peak)

Detected Abnormal Peak Indices: [np.int64(2), np.int64(7), np.int64(10),
np.int64(17), np.int64(22), np.int64(27), np.int64(29), np.int64(32),
np.int64(35), np.int64(40), np.int64(43), np.int64(45), np.int64(48),
np.int64(51), np.int64(54), np.int64(59), np.int64(63), np.int64(66),
np.int64(68), np.int64(74), np.int64(78), np.int64(83), np.int64(86),
np.int64(89), np.int64(100), np.int64(105), np.int64(110), np.int64(116),
np.int64(122), np.int64(126), np.int64(131), np.int64(133), np.int64(137),
np.int64(141), np.int64(143), np.int64(154), np.int64(164), np.int64(166)…..
```

```
# Step 5:Visualize the Results

# Visualize the PPG signals and detected abnormalities
plt.figure(figsize=(15, 10))
for i in range(pleth.shape[1]):
   plt.subplot(3, 2, i + 1)
   plt.plot(time, pleth.iloc[:, i], label=f'Pleth {i + 1}')
   plt.scatter(time[abnormal_peaks_indices],
   pleth.iloc[abnormal_peaks_indices, i], color='red', label='Abnormal Peaks')
   plt.title(f'PPG Signal - Pleth {i + 1}')
   plt.xlabel('Time')
   plt.ylabel('Amplitude')
```

```
    plt.legend()
plt.tight_layout()
plt.show()
```

Step 5 Output :



```
# Step 6:Save the peaks to a new csv file

# Save the results to a new DataFrame and export to CSV
results = pd.DataFrame({
    'time': time[abnormal_peaks_indices],
    'abnormal_peak_index': abnormal_peaks_indices,
})

# Save results to a CSV file
results.to_csv('abnormal_peaks_results.csv', index=False)

print("Abnormal peaks results saved to 'abnormal_peaks_results.csv'")

# Step 6 Output :
Abnormal peaks results saved to 'abnormal_peaks_results.csv'



# Step 7: Some Values from the file ‘abnormal_peaks_results.csv’

time,abnormal_peak_index
2021-01-01 11:22:48.309804,2
2021-01-01 11:22:48.319804,7
2021-01-01 11:22:48.325804,10
2021-01-01 11:22:48.339804,17
2021-01-01 11:22:48.349804,22
2021-01-01 11:22:48.359804,27
2021-01-01 11:22:48.363804,29
2021-01-01 11:22:48.369804,32
2021-01-01 11:22:48.375804,35
2021-01-01 11:22:48.385804,40
2021-01-01 11:22:48.391804,43
```

2021-01-01 11:22:48.395804,45
2021-01-01 11:22:48.401804,48
2021-01-01 11:22:48.407804,51
2021-01-01 11:22:48.413804,54
2021-01-01 11:22:48.423804,59
2021-01-01 11:22:48.431804,63
2021-01-01 11:22:48.437804,66
2021-01-01 11:22:48.441804,68
2021-01-01 11:22:48.453804,74
2021-01-01 11:22:48.461804,78
2021-01-01 11:22:48.471804,83
2021-01-01 11:22:48.477804,86
2021-01-01 11:22:48.483804,89
2021-01-01 11:22:48.505804,100
2021-01-01 11:22:48.515804,105
2021-01-01 11:22:48.525804,110
2021-01-01 11:22:48.537804,116
2021-01-01 11:22:48.549804,122
2021-01-01 11:22:48.557804,126
2021-01-01 11:22:48.567804,131
2021-01-01 11:22:48.571804,133
2021-01-01 11:22:48.579804,137
2021-01-01 11:22:48.587804,141
2021-01-01 11:22:48.591804,143
2021-01-01 11:22:48.613804,154
2021-01-01 11:22:48.633804,164
2021-01-01 11:22:48.637804,166
2021-01-01 11:22:48.665804,180