```
    # This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)


# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

import os
    for dirname, _, filenames in os.walk('/kaggle/input'):
        for filename in filenames:
            print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session
```

```
/kaggle/input/ppg-signal-with-blood-sugar-level-data/clean-dataset.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_21_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_09_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_07_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_13_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_14_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_18_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0006.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_18_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_12_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
```

```
(2)/PPG_Dataset/Labels/label_03_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_04_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_15_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_21_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_02_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0006.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_22_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_15_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_03_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_18_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_11_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_18_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_11_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_19_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_11_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_10_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_03_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0007.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_02_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_23_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_20_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_18_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_04_0002.csv
```

```
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_19_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_10_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_17_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_20_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_16_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_10_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_17_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_02_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_08_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0005.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_11_0004.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_15_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_20_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0004.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0005.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_06_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_21_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_04_0004.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_04_0004.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_11_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/Total.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_15_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
```

```
(2)/PPG_Dataset/Labels/Total.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_22_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_16_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_02_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_22_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_13_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_15_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_07_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_08_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_16_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_06_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0007.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_13_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_02_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_08_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_11_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_18_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0006.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_11_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_11_0004.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_07_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_02_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0002.csv
```

```
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_04_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0005.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0004.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_07_0004.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0005.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_14_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_23_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_20_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_03_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_07_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_04_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_23_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_17_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_04_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_13_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_16_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_09_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_16_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_23_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_17_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_06_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_07_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_19_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
```

```
(2)/PPG_Dataset/Labels/label_10_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_04_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_19_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0006.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_04_0005.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_04_0005.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_22_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0004.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_09_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_20_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_09_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_12_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_14_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_15_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_07_0004.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_07_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_22_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_14_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_22_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_08_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_16_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_01_0004.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_21_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_06_0001.mat
```

```
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_20_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Labels/label_05_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_18_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_13_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_19_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_18_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_10_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0004.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_13_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_07_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_11_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_02_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_21_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_07_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_20_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_15_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_08_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0007.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_02_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_22_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_21_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_11_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
```

```
(2)/PPG_Dataset/RawData/signal_11_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_02_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_19_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_20_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_06_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0005.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_22_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_18_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_04_0004.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_16_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_20_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_20_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_11_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_15_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_02_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_09_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_04_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_15_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_08_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_07_0004.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0007.csv
```

```
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_09_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_22_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_20_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_08_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_16_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_14_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0004.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_10_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_11_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_11_0004.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_22_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_18_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_13_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0004.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0004.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_03_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_11_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0006.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_22_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_03_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_14_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
```

```
(2)/PPG_Dataset/RawData/signal_23_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_20_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_14_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_17_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_04_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_15_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0005.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_04_0004.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_07_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_22_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_23_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_06_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_17_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_02_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_09_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_07_0004.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_18_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_07_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0006.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_04_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_06_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_16_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_21_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_08_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_10_0001.csv
```

```
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_04_0005.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0006.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_19_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_12_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_15_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_07_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_01_0005.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_21_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_02_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_16_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_18_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_15_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_06_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_17_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_19_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_09_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_04_0005.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_07_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_11_0004.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_16_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_04_0003.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_14_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_23_0002.mat
```

```
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_12_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_04_0003.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_16_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_17_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_03_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_13_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_04_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0002.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0005.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_03_0001.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_05_0006.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_23_0001.csv
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/RawData/signal_10_0002.mat
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_04_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_13_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_13_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_04_0003.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_15_0003.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_07_0004.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_01_0005.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_19_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_08_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_04_0005.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_10_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
```

```
(2)/PPG_Dataset/Figures/fig_07_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_11_0004.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_02_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_18_0003.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_01_0003.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_11_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_02_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_22_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_09_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_15_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_18_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_17_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_20_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_05_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_05_0006.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_22_0003.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_06_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_23_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_16_0003.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_09_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_01_0006.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_19_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_01_0007.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_14_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_01_0002.jpg
```

```
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_21_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_17_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_05_0003.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_15_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_04_0004.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_18_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_04_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_20_0003.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_21_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_02_0003.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_14_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_01_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_03_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_01_0004.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_05_0004.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_12_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_05_0005.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_23_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_11_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_07_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_05_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_03_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_16_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_16_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
```

```
(2)/PPG_Dataset/Figures/fig_07_0003.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_20_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_22_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_08_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_06_0001.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_10_0002.jpg
/kaggle/input/ppg-signal-with-blood-sugar-level-data/archive
(2)/PPG_Dataset/Figures/fig_11_0003.jpg
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df=pd.read_csv('/kaggle/input/ppg-signal-with-blood-sugar-level-data/
clean-dataset.csv')

df.dtypes
```

```
PPG_Signal          int64
Patient_Id          int64
Heart_Rate        float64
Systolic_Peak     float64
Diastolic_Peak    float64
Pulse_Area        float64
index               int64
Gender              int64
Age                 int64
Glucose_level       int64
Height              int64
Weight              int64
pl                  int64
dtype: object
```

```
df
```

```
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
505.0
2              511           1        77.0          522.0
505.0
3              511           1        77.0          522.0
505.0
4              511           1        77.0          522.0
```

```
505.0
...             ...          ...         ...             ...
...
844941          513           23        83.0           516.0
510.0
844942          513           23        83.0           516.0
510.0
844943          513           23        83.0           516.0
510.0
844944          513           23        83.0           516.0
510.0
844945          513           23        83.0           516.0
510.0

        Pulse_Area  index  Gender  Age  Glucose_level  Height  Weight
pl
0            393.0      0       1   38             99     180      53
1
1            393.0      1       1   38            102     180      53
2
2            393.0      2       1   38            103     180      53
3
3            393.0      3       1   38            128     180      53
4
4            393.0      4       1   38            130     180      53
5
...            ...    ...     ...  ...            ...     ...     ...
...
844941       366.0     43       1   27            108     173      57
1463368
844942       366.0     42       1   27            100     173      57
1463369
844943       366.0     43       1   27            108     173      57
1463370
844944       366.0     42       1   27            100     173      57
1463371
844945       366.0     43       1   27            108     173      57
1463372

[844946 rows x 13 columns]
```

```python
# Check for missing values in each column
missing_values = df.isnull().sum()

# Print the count of missing values for each column
print("Missing Values in Each Column:")
print(missing_values)

#if we have missing values, can lead to inaccuracies in the model
training portion
```

```
Missing Values in Each Column:
PPG_Signal        0
Patient_Id        0
Heart_Rate        0
Systolic_Peak     0
Diastolic_Peak    0
Pulse_Area        0
index             0
Gender            0
Age               0
Glucose_level     0
Height            0
Weight            0
pl                0
dtype: int64
```

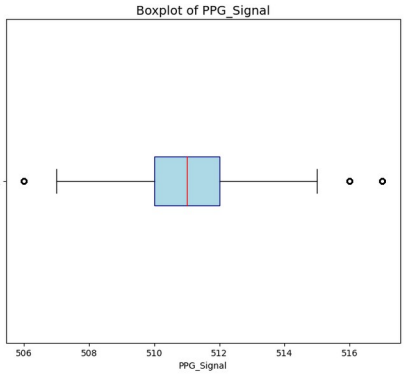
```
#Loop through each column and print the unique values
for column in df.columns:
    unique_values = df[column].unique()
    count_values  = len(df[column].unique())
    print(f"Column: {column}")
    print(f"Unique Values: {unique_values}\n")
    print(f"total count unique values : { count_values}\n")
```

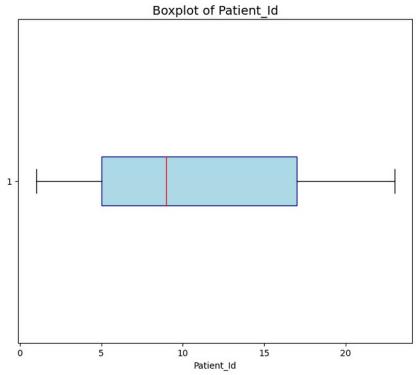
```
Column: PPG_Signal
Unique Values: [511 512 513 514 515 516 517 510 509 508 507 506]

total count unique values : 12

Column: Patient_Id
Unique Values: [ 1  2  3  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
20 21 22 23]

total count unique values : 22

Column: Heart_Rate
Unique Values: [77. 75. 80. 79. 81. 76. 83. 65. 61. 63. 70. 85. 84.
88. 86. 89. 93. 87.
 90. 78. 82. 64. 67. 68. 66. 74. 73. 71. 91. 62. 72. 92.]

total count unique values : 32

Column: Systolic_Peak
Unique Values: [522. 520. 521. 518. 519. 524. 523. 526. 525. 527. 528.
516. 514. 515.
 517. 529.]

total count unique values : 16

Column: Diastolic_Peak
```

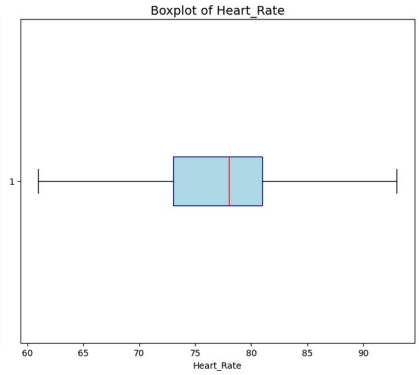
```
Unique Values: [505. 507. 508. 506. 509. 504. 511. 510. 512.]

total count unique values : 9

Column: Pulse_Area
Unique Values: [393.  406.  383.  385.  386.  375.  394.  380.  376.
396.  399.  369.
 468.  481.5 467.  438.  434.  355.  398.  365.  347.  356.  353.
379.
 345.  363.  367.  364.  378.  312.  370.  349.  377.  366.  321.
324.
 338.  390.  388.  374.  381.  362.  373.  405.  384.  480.  455.
446.
 466.  459.  389.  412.  410.  309.5 400.  477.  465.  354.  417.
433.
 426.  322.  313.  475.  335.  402.  401.  334.  427.  395.  428.
404.
 416.  432.  479.  421.  422.  408.  343.  323.  450.  391.  423.
392.
 342.  431.  333.  346.  357.  403.  424.  397. ]

total count unique values : 92

Column: index
Unique Values: [ 0  1  2  3  4  5  6 31 32 33 44 45 51 52 53 54 55 56
57 58 59 60 61 62
 63 64 65 66  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26
 27 28 30 29 34 35 36 38 37 39 40 41 42 43]

total count unique values : 62

Column: Gender
Unique Values: [1 0]

total count unique values : 2

Column: Age
Unique Values: [38 25 33 23 31 39 37 22 61 50 51 45 24 26 48 27]

total count unique values : 16

Column: Glucose_level
Unique Values: [ 99 102 103 128 130 134 136 108 111 118 120 127  94
96 106 110 129  88
 146 124 100 113  95 115 183 139 112 140]

total count unique values : 28

Column: Height
```

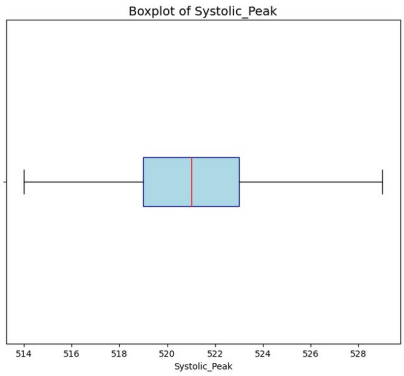
```
Unique Values: [180 187 175 165 179 172 182 161 178 157 169 170 154
173]

total count unique values : 14

Column: Weight
Unique Values: [ 53  75 103  56  60  93  63  90  62  61  96  83  89
55  42  88  50  57]

total count unique values : 18

Column: pl
Unique Values: [      1       2       3 ... 1463370 1463371 1463372]

total count unique values : 844946
```

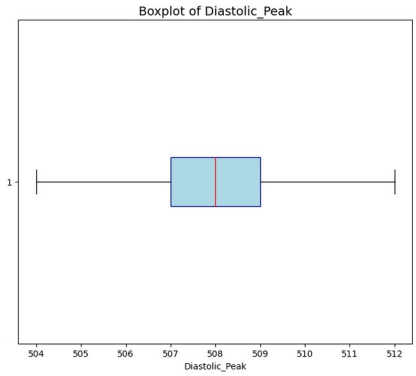
```python
def plot_boxplots_alternative(df):
    num_columns = df.select_dtypes(include='number').columns
    num_features = len(num_columns)
    rows = (num_features // 3) + 1
    cols = min(num_features, 3)
    fig, axes = plt.subplots(rows, cols, figsize=(20, rows * 6))
    axes = axes.flatten()  # Flatten in case of more subplots

    for i, col in enumerate(num_columns):
        axes[i].boxplot(df[col], vert=False, patch_artist=True,
                        boxprops=dict(facecolor='lightblue',
color='navy'),
                        medianprops=dict(color='red'))
        axes[i].set_title(f'Boxplot of {col}', fontsize=14)
        axes[i].set_xlabel(col)

    for j in range(i + 1, len(axes)):
        fig.delaxes(axes[j])

    plt.tight_layout()
    plt.show()

plot_boxplots_alternative(df)
```

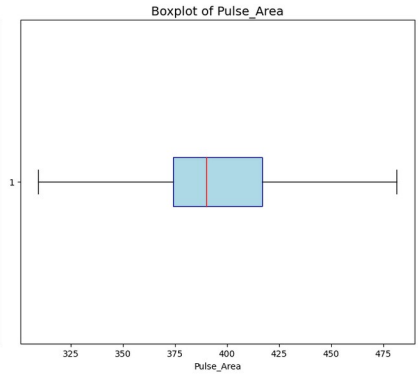

Boxplot of PPG_Signal
1
506
508
510
512
514
516
PPG_Signal

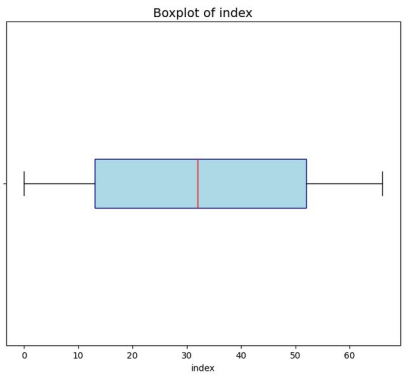

Boxplot of Patient_Id
1
0
5
10
15
20
Patient_Id

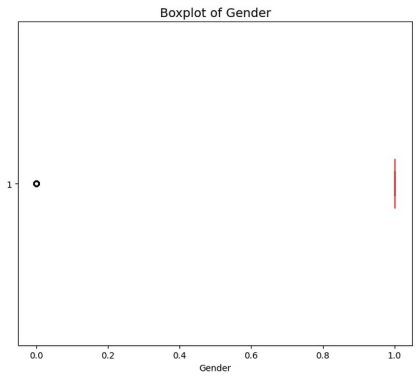

Boxplot of Heart_Rate
1
60
65
70
75
80
85
90
Heart_Rate

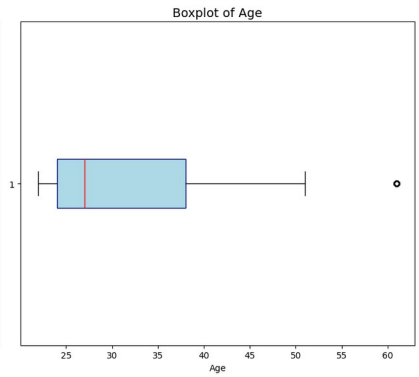

Boxplot of Systolic_Peak
1
514
516
518
520
522
524
526
528
Systolic_Peak

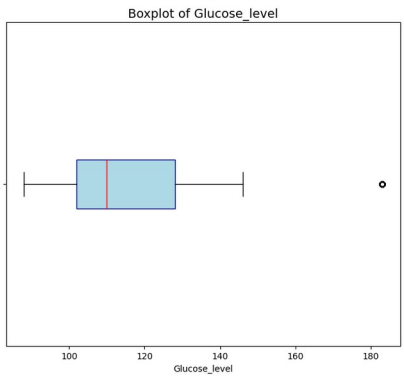

Boxplot of Diastolic_Peak
1
504
505
506
507
508
509
510
511
512
Diastolic_Peak

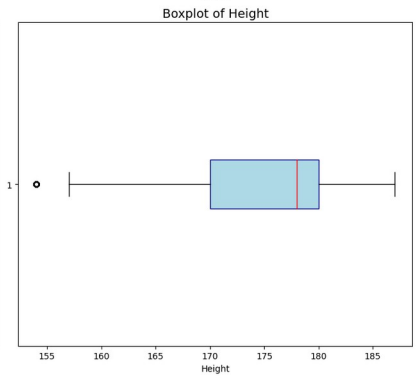

Boxplot of Pulse_Area
1
325
350
375
400
425
450
475
Pulse_Area

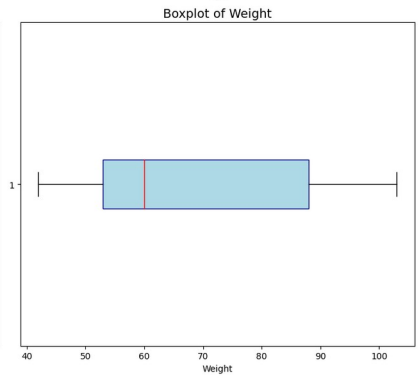

Boxplot of index
1
0
10
20
30
40
50
60
index

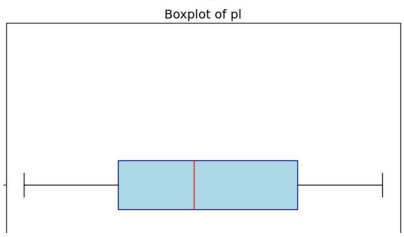

Boxplot of Gender
1
0.0
0.2
0.4
0.6
0.8
1.0
Gender


Boxplot of Age
1
25
30
35
40
45
50
55
60
Age


Boxplot of Glucose_level
1
100
120
140
160
180
Glucose_level


Boxplot of Height
1
155
160
165
170
175
180
185
Height


Boxplot of Weight
1
40
50
60
70
80
90
100
Weight


Boxplot of pl
1

```
df

        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
505.0
2              511           1        77.0          522.0
505.0
3              511           1        77.0          522.0
505.0
4              511           1        77.0          522.0
505.0
...            ...         ...         ...            ...
...
844941         513          23        83.0          516.0
510.0
844942         513          23        83.0          516.0
510.0
844943         513          23        83.0          516.0
510.0
844944         513          23        83.0          516.0
510.0
844945         513          23        83.0          516.0
510.0

        Pulse_Area  index  Gender  Age  Glucose_level  Height  Weight
pl
0            393.0      0       1   38             99     180      53
1
1            393.0      1       1   38            102     180      53
2
2            393.0      2       1   38            103     180      53
3
3            393.0      3       1   38            128     180      53
4
4            393.0      4       1   38            130     180      53
5
...            ...    ...     ...  ...            ...     ...     ...
...
844941       366.0     43       1   27            108     173      57
1463368
844942       366.0     42       1   27            100     173      57
1463369
844943       366.0     43       1   27            108     173      57
1463370
844944       366.0     42       1   27            100     173      57
1463371
844945       366.0     43       1   27            108     173      57
```

```
1463372

[844946 rows x 13 columns]
```

```python
import pandas as pd


# Define the bins (age ranges) and the corresponding labels
bins = [20, 30, 40, 50, 60,70]  # Age ranges (e.g., 0-20, 21-30, etc.)
labels = [1,2,3,4,5]  # Corresponding labels

# Create a new column 'Age Range' based on the age values
df['Age Range'] = pd.cut(df['Age'], bins=bins, labels=labels,
right=False)

# Display the DataFrame with the new 'Age Range' column
print(df)
```

```
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
505.0
2              511           1        77.0          522.0
505.0
3              511           1        77.0          522.0
505.0
4              511           1        77.0          522.0
505.0
...            ...         ...         ...            ...
...
844941         513          23        83.0          516.0
510.0
844942         513          23        83.0          516.0
510.0
844943         513          23        83.0          516.0
510.0
844944         513          23        83.0          516.0
510.0
844945         513          23        83.0          516.0
510.0

        Pulse_Area  index  Gender  Age  Glucose_level  Height  Weight
\
0            393.0      0       1   38             99     180      53

1            393.0      1       1   38            102     180      53

2            393.0      2       1   38            103     180      53
```

```
3            393.0      3       1   38            128     180      53

4            393.0      4       1   38            130     180      53

...            ...    ...     ...  ...            ...     ...     ...

844941       366.0     43       1   27            108     173      57

844942       366.0     42       1   27            100     173      57

844943       366.0     43       1   27            108     173      57

844944       366.0     42       1   27            100     173      57

844945       366.0     43       1   27            108     173      57


             pl Age Range
0             1         2
1             2         2
2             3         2
3             4         2
4             5         2
...         ...       ...
844941  1463368         1
844942  1463369         1
844943  1463370         1
844944  1463371         1
844945  1463372         1

[844946 rows x 14 columns]

df

        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
505.0
2              511           1        77.0          522.0
505.0
3              511           1        77.0          522.0
505.0
4              511           1        77.0          522.0
505.0
...            ...         ...         ...            ...
...
844941         513          23        83.0          516.0
510.0
844942         513          23        83.0          516.0
```

```
510.0
844943         513          23        83.0          516.0
510.0
844944         513          23        83.0          516.0
510.0
844945         513          23        83.0          516.0
510.0

         Pulse_Area  index  Gender  Age  Glucose_level  Height  Weight
\
0             393.0      0       1   38             99     180      53

1             393.0      1       1   38            102     180      53

2             393.0      2       1   38            103     180      53

3             393.0      3       1   38            128     180      53

4             393.0      4       1   38            130     180      53

...             ...    ...     ...  ...            ...     ...     ...

844941        366.0     43       1   27            108     173      57

844942        366.0     42       1   27            100     173      57

844943        366.0     43       1   27            108     173      57

844944        366.0     42       1   27            100     173      57

844945        366.0     43       1   27            108     173      57


              pl Age Range
0              1          2
1              2          2
2              3          2
3              4          2
4              5          2
...          ...        ...
844941   1463368          1
844942   1463369          1
844943   1463370          1
844944   1463371          1
844945   1463372          1

[844946 rows x 14 columns]
```

```python
# Drop columns without modifying the original DataFrame
df = df.drop(columns=['Age'])
```

```
# Check the result
df.head()
```

```
   PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0         511           1        77.0          522.0           505.0

1         511           1        77.0          522.0           505.0

2         511           1        77.0          522.0           505.0

3         511           1        77.0          522.0           505.0

4         511           1        77.0          522.0           505.0


   Pulse_Area  index  Gender  Glucose_level  Height  Weight  pl Age
Range
0       393.0      0       1             99     180      53   1
2
1       393.0      1       1            102     180      53   2
2
2       393.0      2       1            103     180      53   3
2
3       393.0      3       1            128     180      53   4
2
4       393.0      4       1            130     180      53   5
2
```

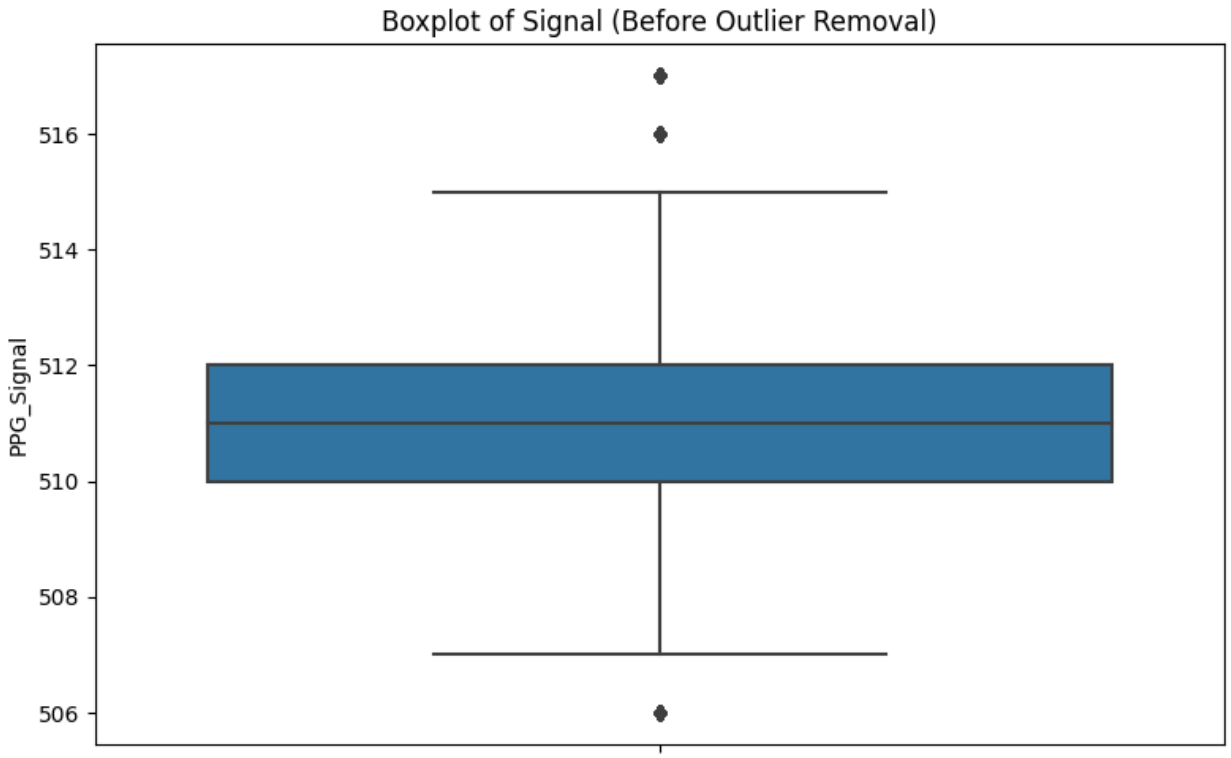
```
df.columns
```

```
Index(['PPG_Signal', 'Patient_Id', 'Heart_Rate', 'Systolic_Peak',
       'Diastolic_Peak', 'Pulse_Area', 'index', 'Gender',
'Glucose_level',
       'Height', 'Weight', 'pl', 'Age Range'],
      dtype='object')
```

Remove Outlier in dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Create a synthetic dataset (replace with your actual dataset)
np.random.seed(42)

# Print available columns to verify if 'Signal' is present
#print("Columns in the dataset:", df.columns)

# List of columns to handle outliers (in this case, just the 'Signal'
```

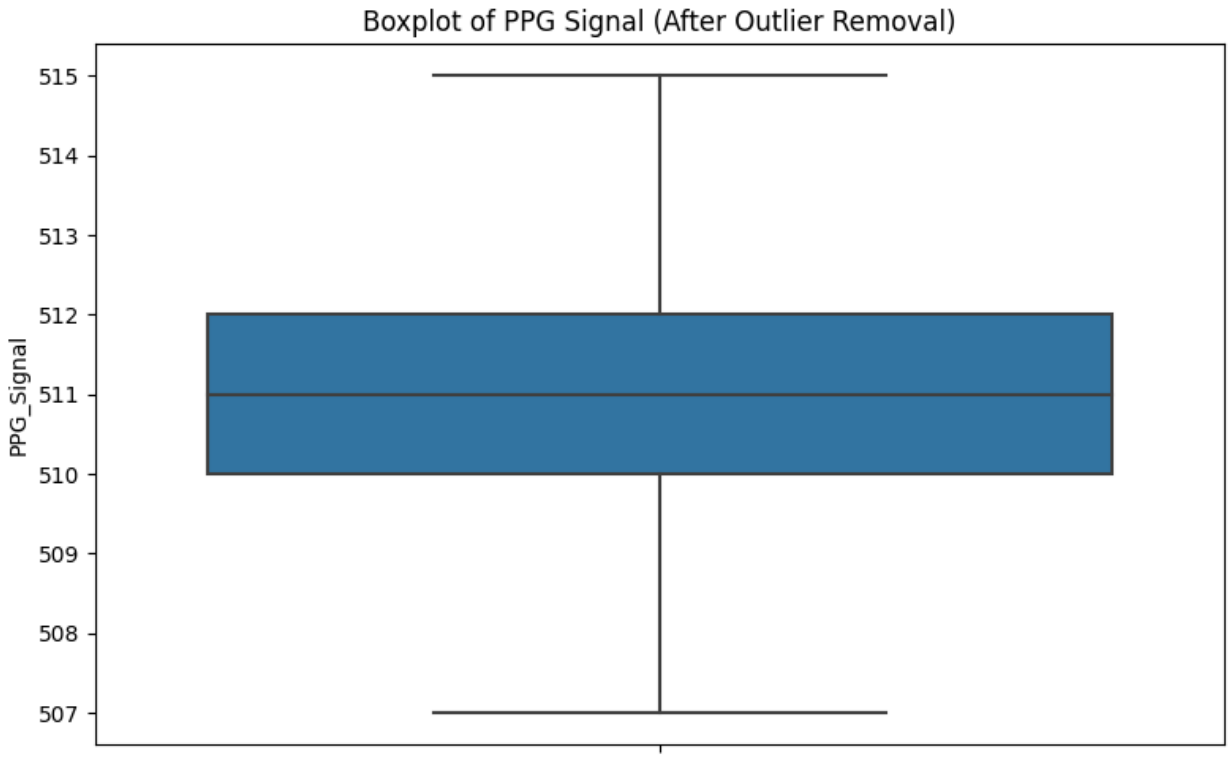
```
column)
columns_with_outliers = ['PPG_Signal']

# Create a boxplot for the 'Signal' column before handling outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=df['PPG_Signal'])
plt.title('Boxplot of Signal (Before Outlier Removal)')
plt.tight_layout()
plt.show()

# Define a function to handle outliers using the IQR method
def find_outliers(df, column, iqr_multiplier=1.5):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - iqr_multiplier * IQR
    upper_bound = Q3 + iqr_multiplier * IQR

    # Identify the indices of rows that are outliers
    outlier_indices = df[(df[column] < lower_bound) | (df[column] >
upper_bound)].index
    return outlier_indices

# Find the outliers in the 'Signal' column
outlier_indices = find_outliers(df, 'PPG_Signal')

# Show the dataset rows that contain outliers
print("Outliers Found at Indices:")
print(outlier_indices)

# Show dataset rows containing the outliers
print("\nRows containing outliers:")
print(df.loc[outlier_indices])

# Remove outliers from the dataset
df = df.drop(outlier_indices)

# Print the size of the dataset after removing outliers
print("\nDataset size after removing outliers:", df.shape)

# Create a boxplot for the 'Signal' column after handling outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=df['PPG_Signal'])
plt.title('Boxplot of PPG Signal (After Outlier Removal)')
plt.tight_layout()
plt.show()

# Optional: Save the cleaned data to a new CSV
# df_no_outliers.to_csv('/mnt/data/cleaned_data.csv', index=False)
```




```
Outliers Found at Indices:
Index([   769,    770,    771,    772,    773,    774,    775,    776,
777,
          778,
       ...
       814354, 814355, 814356, 814357, 814358, 814359, 814360, 814361,
814362,
       814363],
      dtype='int64', length=48385)

Rows containing outliers:
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
769            516           1        77.0          522.0
505.0
770            516           1        77.0          522.0
505.0
771            516           1        77.0          522.0
505.0
772            516           1        77.0          522.0
505.0
773            516           1        77.0          522.0
505.0
...            ...         ...         ...            ...
...
814359         506          22        73.0          528.0
```

```
508.0
814360          506           22         73.0           528.0
508.0
814361          506           22         73.0           528.0
508.0
814362          506           22         73.0           528.0
508.0
814363          506           22         73.0           528.0
508.0

         Pulse_Area  index  Gender  Glucose_level  Height  Weight
pl  \
769          393.0      6       1            136     180      53
770
770          393.0      0       1             99     180      53
771
771          393.0      1       1            102     180      53
772
772          393.0      2       1            103     180      53
773
773          393.0      3       1            128     180      53
774
...            ...    ...     ...            ...     ...     ...
...
814359       412.0     39       0             88     170      50
1423279
814360       412.0     40       0            108     170      50
1423280
814361       412.0     41       0            124     170      50
1423281
814362       412.0     39       0             88     170      50
1423282
814363       412.0     40       0            108     170      50
1423283

       Age Range
769            2
770            2
771            2
772            2
773            2
...          ...
814359         1
814360         1
814361         1
814362         1
814363         1

[48385 rows x 13 columns]
```

```
Dataset size after removing outliers: (796561, 13)
```



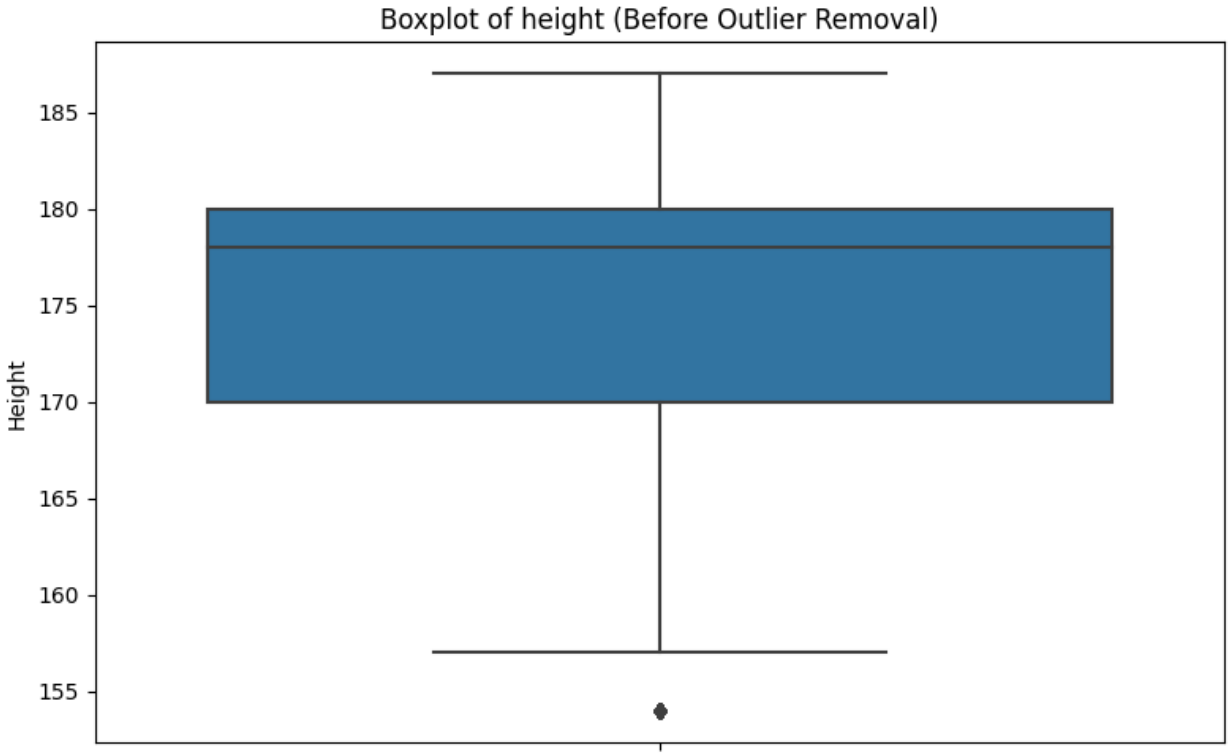

```
df
```

	PPG_Signal	Patient_Id	Heart_Rate	Systolic_Peak	Diastolic_Peak	\
0	511	1	77.0	522.0	505.0	
1	511	1	77.0	522.0	505.0	
2	511	1	77.0	522.0	505.0	
3	511	1	77.0	522.0	505.0	
4	511	1	77.0	522.0	505.0	
...	...	...	...	...	...	
844941	513	23	83.0	516.0	510.0	
844942	513	23	83.0	516.0	510.0	
844943	513	23	83.0	516.0	510.0	
844944	513	23	83.0	516.0		

```
510.0
844945         513           23        83.0          516.0
510.0

         Pulse_Area  index  Gender  Glucose_level  Height  Weight
pl  \
0             393.0      0       1             99     180      53
1
1             393.0      1       1            102     180      53
2
2             393.0      2       1            103     180      53
3
3             393.0      3       1            128     180      53
4
4             393.0      4       1            130     180      53
5
...             ...    ...     ...            ...     ...     ...
...
844941        366.0     43       1            108     173      57
1463368
844942        366.0     42       1            100     173      57
1463369
844943        366.0     43       1            108     173      57
1463370
844944        366.0     42       1            100     173      57
1463371
844945        366.0     43       1            108     173      57
1463372

       Age Range
0              2
1              2
2              2
3              2
4              2
...          ...
844941         1
844942         1
844943         1
844944         1
844945         1

[796561 rows x 13 columns]

df.columns

Index(['PPG_Signal', 'Patient_Id', 'Heart_Rate', 'Systolic_Peak',
       'Diastolic_Peak', 'Pulse_Area', 'index', 'Gender',
'Glucose_level',
```

```
       'Height', 'Weight', 'pl', 'Age Range'],
      dtype='object')

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Create a synthetic dataset (replace with your actual dataset)
np.random.seed(42)

# Print available columns to verify if 'Signal' is present
#print("Columns in the dataset:", df.columns)

# List of columns to handle outliers (in this case, just the 'Signal'
column)
columns_with_outliers = ['Height']

# Create a boxplot for the 'Signal' column before handling outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=df['Height'])
plt.title('Boxplot of height (Before Outlier Removal)')
plt.tight_layout()
plt.show()

# Define a function to handle outliers using the IQR method
def find_outliers(df, column, iqr_multiplier=1.5):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - iqr_multiplier * IQR
    upper_bound = Q3 + iqr_multiplier * IQR

    # Identify the indices of rows that are outliers
    outlier_indices = df[(df[column] < lower_bound) | (df[column] >
upper_bound)].index
    return outlier_indices

# Find the outliers in the 'Signal' column
outlier_indices = find_outliers(df, 'Height')

# Show the dataset rows that contain outliers
print("Outliers Found at Indices:")
print(outlier_indices)

# Show dataset rows containing the outliers
print("\nRows containing outliers:")
print(df.loc[outlier_indices])

# Remove outliers from the dataset
```

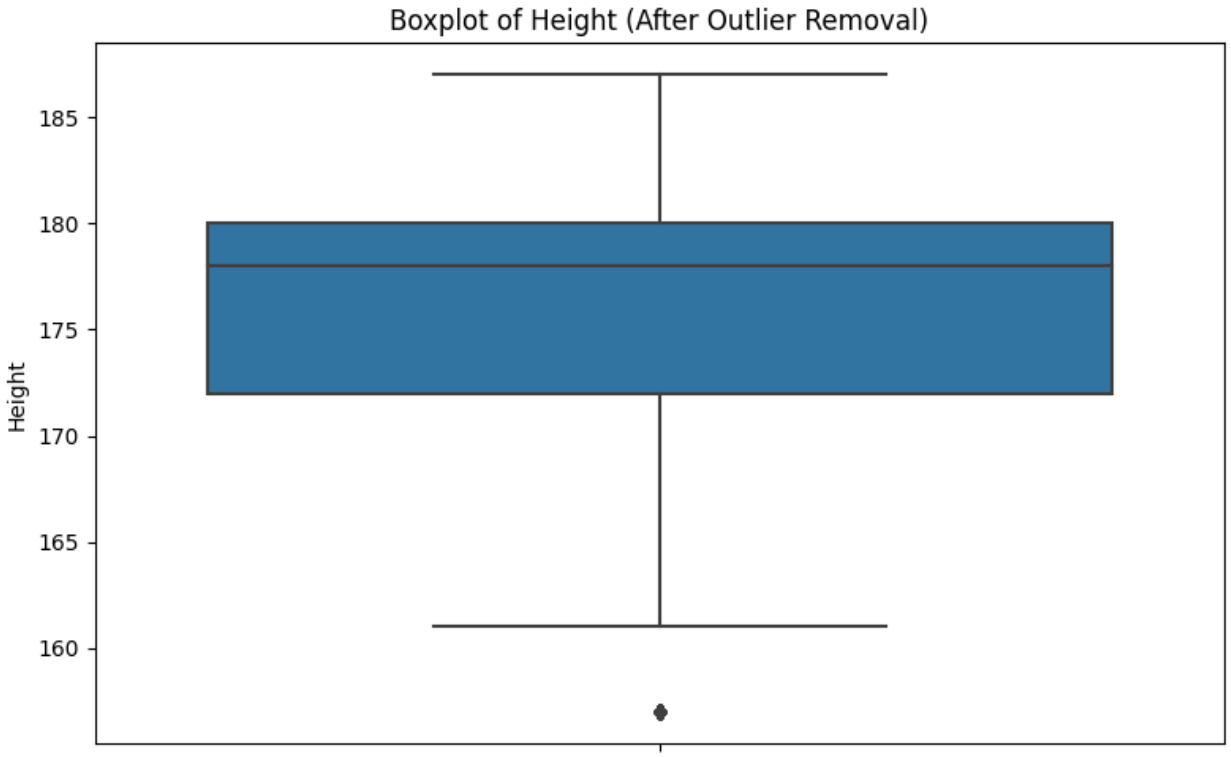
```
df = df.drop(outlier_indices)

# Print the size of the dataset after removing outliers
print("\nDataset size after removing outliers:", df.shape)

# Create a boxplot for the 'Signal' column after handling outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=df['Height'])
plt.title('Boxplot of Height (After Outlier Removal)')
plt.tight_layout()
plt.show()

# Optional: Save the cleaned data to a new CSV
# df_no_outliers.to_csv('/mnt/data/cleaned_data.csv', index=False)
```




```
Outliers Found at Indices:
Index([704505, 704506, 704507, 704508, 704509, 704510, 704511, 704512,
704513,
       704514,
       ...
       731761, 731762, 731763, 731764, 731765, 731766, 731767, 731768,
731769,
       731770],
      dtype='int64', length=24985)

Rows containing outliers:
```

```
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
704505         510          20        79.0          521.0
506.0
704506         510          20        79.0          521.0
506.0
704507         510          20        79.0          521.0
506.0
704508         510          20        79.0          521.0
506.0
704509         510          20        79.0          521.0
506.0
...            ...         ...         ...            ...
...
731766         507          20        65.0          521.0
507.0
731767         507          20        65.0          521.0
507.0
731768         507          20        65.0          521.0
507.0
731769         507          20        65.0          521.0
507.0
731770         507          20        65.0          521.0
507.0

        Pulse_Area  index  Gender  Glucose_level  Height  Weight
pl  \
704505       388.0     34       0            120     154      42
1252852
704506       388.0     35       0            124     154      42
1252853
704507       388.0     36       0            136     154      42
1252854
704508       388.0     34       0            120     154      42
1252855
704509       388.0     35       0            124     154      42
1252856
...            ...    ...     ...            ...     ...     ...
...
731766       468.0     34       0            120     154      42
1313893
731767       468.0     35       0            124     154      42
1313894
731768       468.0     36       0            136     154      42
1313895
731769       468.0     34       0            120     154      42
1313896
731770       468.0     35       0            124     154      42
1313897
```

```
       Age Range
704505          1
704506          1
704507          1
704508          1
704509          1
...           ...
731766          1
731767          1
731768          1
731769          1
731770          1

[24985 rows x 13 columns]

Dataset size after removing outliers: (771576, 13)
```




```
df

        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
505.0
```

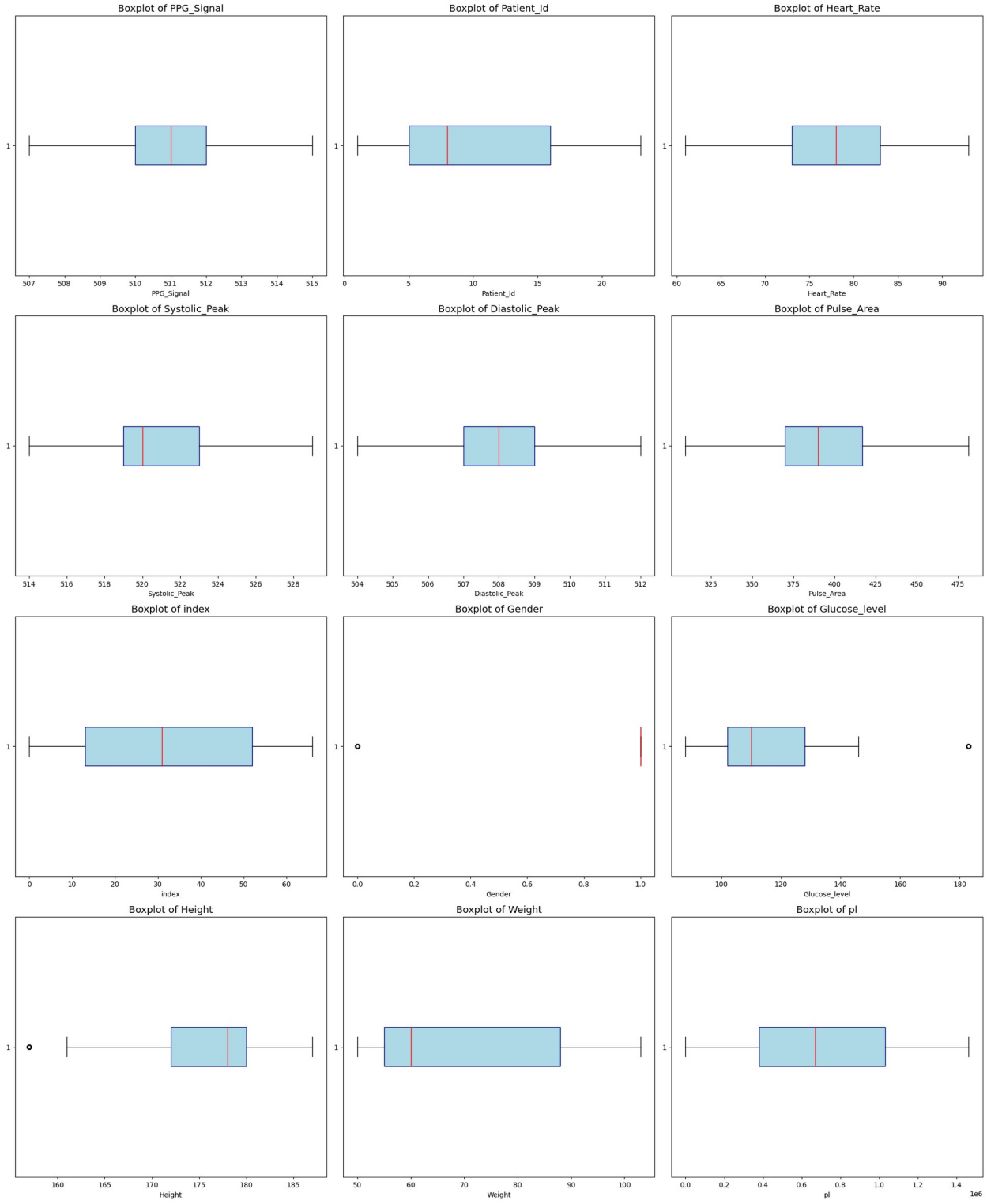
```
2               511           1        77.0          522.0
505.0
3               511           1        77.0          522.0
505.0
4               511           1        77.0          522.0
505.0
...             ...         ...         ...            ...
...
844941          513          23        83.0          516.0
510.0
844942          513          23        83.0          516.0
510.0
844943          513          23        83.0          516.0
510.0
844944          513          23        83.0          516.0
510.0
844945          513          23        83.0          516.0
510.0

         Pulse_Area  index  Gender  Glucose_level  Height  Weight
pl  \
0             393.0      0       1             99     180      53
1
1             393.0      1       1            102     180      53
2
2             393.0      2       1            103     180      53
3
3             393.0      3       1            128     180      53
4
4             393.0      4       1            130     180      53
5
...             ...    ...     ...            ...     ...     ...
...
844941        366.0     43       1            108     173      57
1463368
844942        366.0     42       1            100     173      57
1463369
844943        366.0     43       1            108     173      57
1463370
844944        366.0     42       1            100     173      57
1463371
844945        366.0     43       1            108     173      57
1463372

       Age Range
0              2
1              2
2              2
3              2
```

```
4               2
...           ...
844941          1
844942          1
844943          1
844944          1
844945          1

[771576 rows x 13 columns]
```

```
df.columns
```

```
Index(['PPG_Signal', 'Patient_Id', 'Heart_Rate', 'Systolic_Peak',
       'Diastolic_Peak', 'Pulse_Area', 'index', 'Gender',
'Glucose_level',
       'Height', 'Weight', 'pl', 'Age Range'],
      dtype='object')
```

```
def plot_boxplots_alternative(df):
    num_columns = df.select_dtypes(include='number').columns
    num_features = len(num_columns)
    rows = (num_features // 3) + 1
    cols = min(num_features, 3)
    fig, axes = plt.subplots(rows, cols, figsize=(20, rows * 6))
    axes = axes.flatten()  # Flatten in case of more subplots

    for i, col in enumerate(num_columns):
        axes[i].boxplot(df[col], vert=False, patch_artist=True,
                        boxprops=dict(facecolor='lightblue',
color='navy'),
                        medianprops=dict(color='red'))
        axes[i].set_title(f'Boxplot of {col}', fontsize=14)
        axes[i].set_xlabel(col)

    for j in range(i + 1, len(axes)):
        fig.delaxes(axes[j])

    plt.tight_layout()
    plt.show()

plot_boxplots_alternative(df)
```

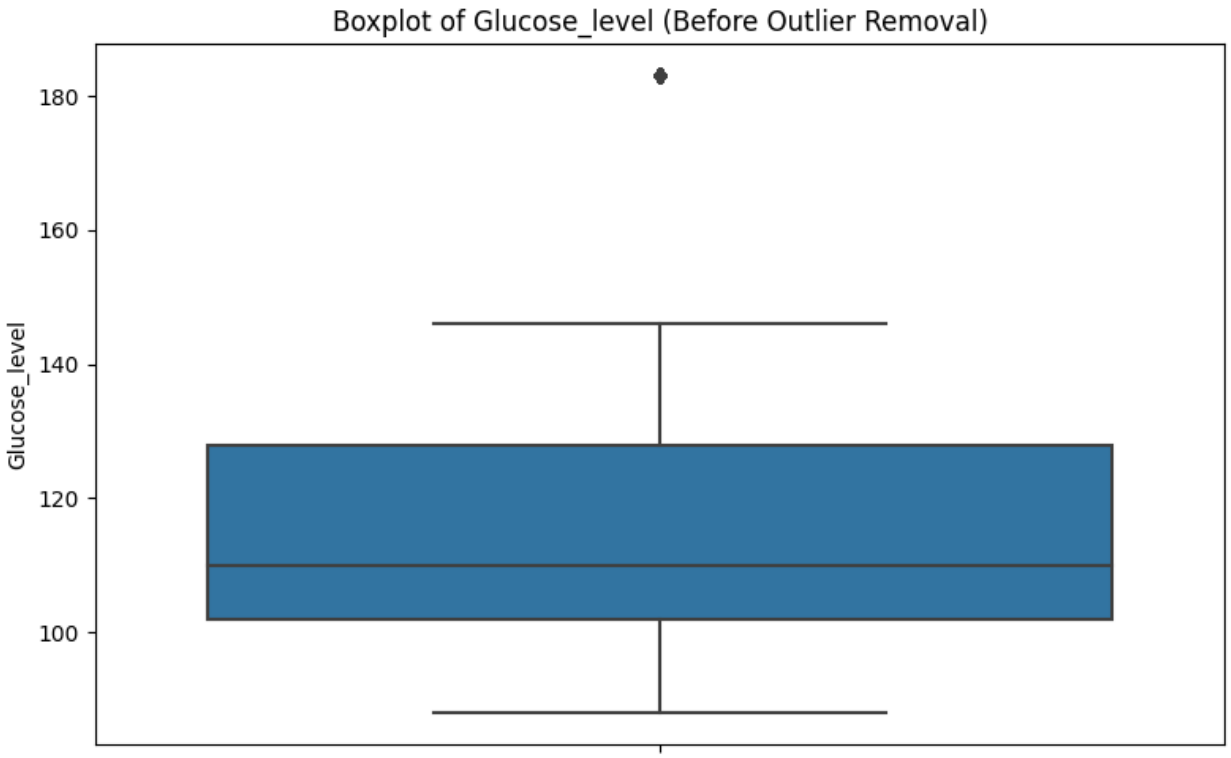



```
df.columns
```

```
Index(['PPG_Signal', 'Patient_Id', 'Heart_Rate', 'Systolic_Peak',
       'Diastolic_Peak', 'Pulse_Area', 'index', 'Gender',
```

```
'Glucose_level',
       'Height', 'Weight', 'pl', 'Age Range'],
      dtype='object')

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Create a synthetic dataset (replace with your actual dataset)
np.random.seed(42)

# Print available columns to verify if 'Signal' is present
#print("Columns in the dataset:", df.columns)

# List of columns to handle outliers (in this case, just the 'Signal'
column)
columns_with_outliers = ['Glucose_level']

# Create a boxplot for the 'Signal' column before handling outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=df['Glucose_level'])
plt.title('Boxplot of Glucose_level (Before Outlier Removal)')
plt.tight_layout()
plt.show()

# Define a function to handle outliers using the IQR method
def find_outliers(df, column, iqr_multiplier=1.5):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - iqr_multiplier * IQR
    upper_bound = Q3 + iqr_multiplier * IQR

    # Identify the indices of rows that are outliers
    outlier_indices = df[(df[column] < lower_bound) | (df[column] >
upper_bound)].index
    return outlier_indices

# Find the outliers in the 'Signal' column
outlier_indices = find_outliers(df, 'Glucose_level')

# Show the dataset rows that contain outliers
print("Outliers Found at Indices:")
print(outlier_indices)

# Show dataset rows containing the outliers
print("\nRows containing outliers:")
print(df.loc[outlier_indices])
```

```
# Remove outliers from the dataset
df = df.drop(outlier_indices)

# Print the size of the dataset after removing outliers
print("\nDataset size after removing outliers:", df.shape)

# Create a boxplot for the 'Signal' column after handling outliers
plt.figure(figsize=(8, 5))
sns.boxplot(y=df['Height'])
plt.title('Boxplot of Glucose_level (After Outlier Removal)')
plt.tight_layout()
plt.show()

# Optional: Save the cleaned data to a new CSV
# df_no_outliers.to_csv('/mnt/data/cleaned_data.csv', index=False)
```

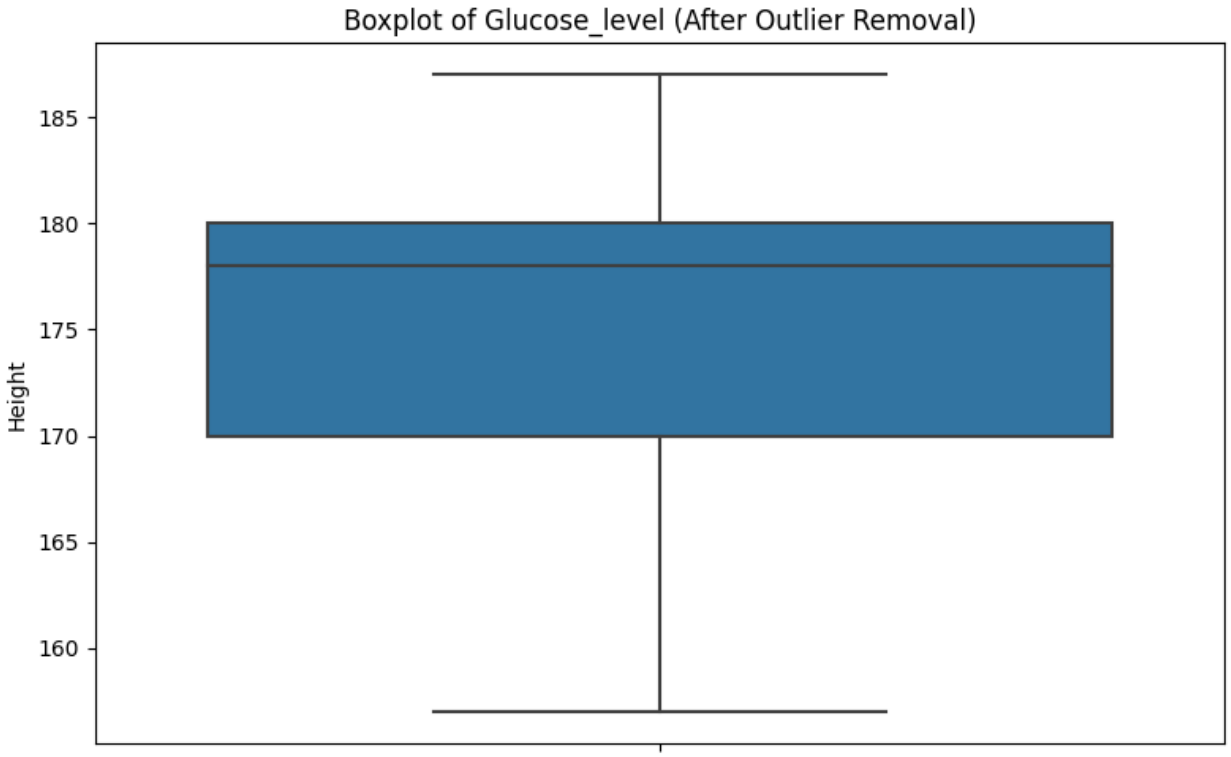



```
Outliers Found at Indices:
Index([522867, 522869, 522871, 522873, 522875, 522877, 522879, 522881,
522883,
       522885,
       ...
       693060, 693063, 693066, 693069, 693072, 693075, 693078, 693081,
693084,
       693087],
      dtype='int64', length=26602)
```

```
Rows containing outliers:
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
522867         512          13        76.0          514.0
511.0
522869         512          13        76.0          514.0
511.0
522871         512          13        76.0          514.0
511.0
522873         512          13        76.0          514.0
511.0
522875         512          13        76.0          514.0
511.0
...            ...         ...         ...            ...
...
693075         511          18        93.0          522.0
510.0
693078         511          18        93.0          522.0
510.0
693081         511          18        93.0          522.0
510.0
693084         511          18        93.0          522.0
510.0
693087         511          18        93.0          522.0
510.0

        Pulse_Area  index  Gender  Glucose_level  Height  Weight
pl  \
522867       401.0     15       1            183     175      96
884030
522869       401.0     15       1            183     175      96
884032
522871       401.0     15       1            183     175      96
884034
522873       401.0     15       1            183     175      96
884036
522875       401.0     15       1            183     175      96
884038
...            ...    ...     ...            ...     ...     ...
...
693075       323.0     28       1            183     180      90
1202688
693078       323.0     28       1            183     180      90
1202691
693081       323.0     28       1            183     180      90
1202694
693084       323.0     28       1            183     180      90
1202697
693087       323.0     28       1            183     180      90
```

```
1202700
```

```
       Age Range
522867          4
522869          4
522871          4
522873          4
522875          4
...           ...
693075          1
693078          1
693081          1
693084          1
693087          1

[26602 rows x 13 columns]

Dataset size after removing outliers: (744974, 13)
```




```
df
```

```
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
```

```
505.0
2              511            1        77.0          522.0
505.0
3              511            1        77.0          522.0
505.0
4              511            1        77.0          522.0
505.0
...            ...          ...         ...            ...
...
844941         513           23        83.0          516.0
510.0
844942         513           23        83.0          516.0
510.0
844943         513           23        83.0          516.0
510.0
844944         513           23        83.0          516.0
510.0
844945         513           23        83.0          516.0
510.0

        Pulse_Area  index  Gender  Glucose_level  Height  Weight
pl  \
0            393.0      0       1             99     180      53
1
1            393.0      1       1            102     180      53
2
2            393.0      2       1            103     180      53
3
3            393.0      3       1            128     180      53
4
4            393.0      4       1            130     180      53
5
...            ...    ...     ...            ...     ...     ...
...
844941       366.0     43       1            108     173      57
1463368
844942       366.0     42       1            100     173      57
1463369
844943       366.0     43       1            108     173      57
1463370
844944       366.0     42       1            100     173      57
1463371
844945       366.0     43       1            108     173      57
1463372

       Age Range
0              2
1              2
2              2
```

```
3              2
4              2
...          ...
844941         1
844942         1
844943         1
844944         1
844945         1

[744974 rows x 13 columns]
```

```python
# Calculate skewness for each numeric column in the DataFrame and print the result
numeric_columns = df.select_dtypes(include=['number'])  # Select only numeric columns
for column in numeric_columns.columns:
    skew_value = df[column].skew()
    print(f"Skewness of '{column}': {skew_value:.4f}")
```

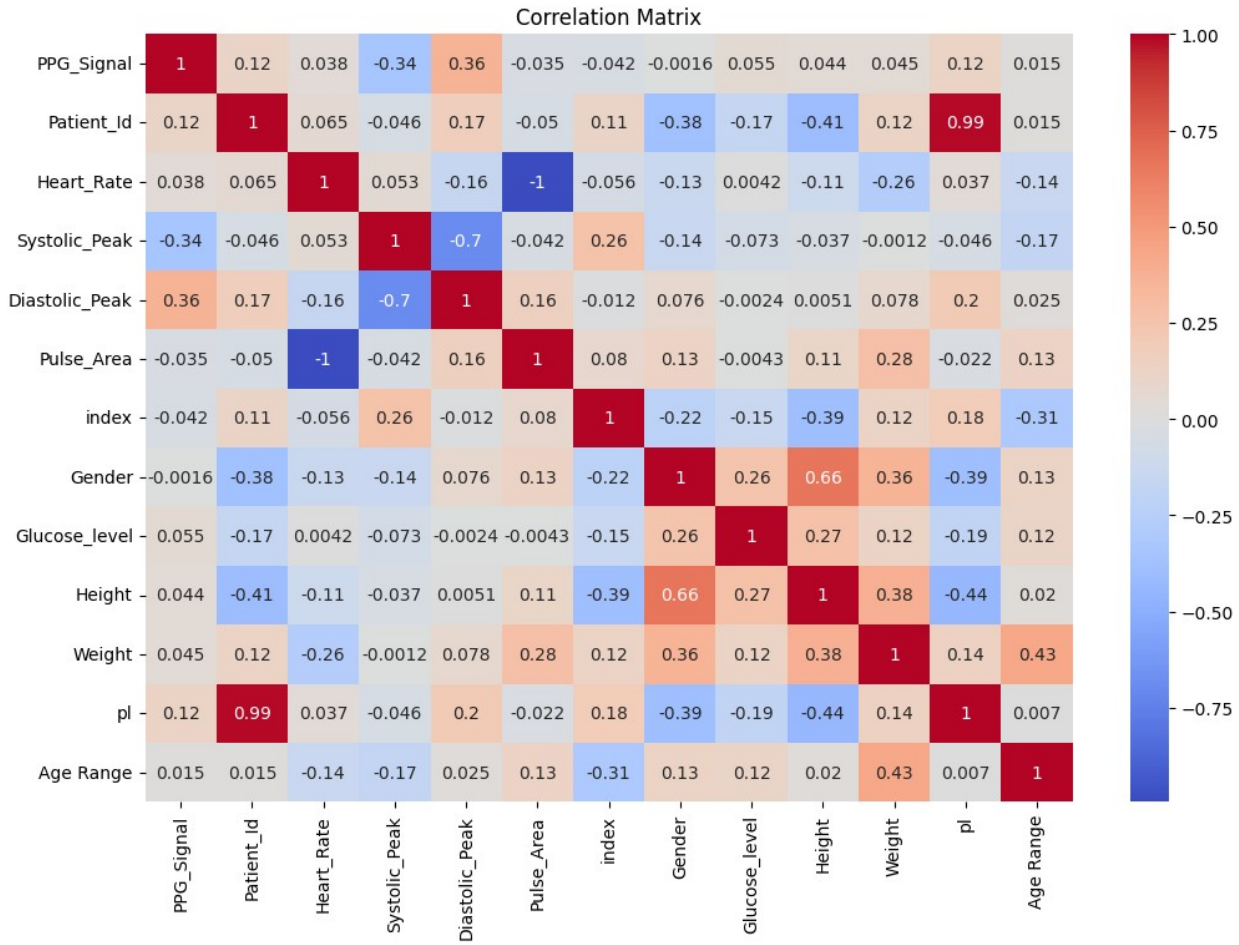
```
Skewness of 'PPG_Signal': 0.2806
Skewness of 'Patient_Id': 0.4458
Skewness of 'Heart_Rate': -0.2021
Skewness of 'Systolic_Peak': -0.0570
Skewness of 'Diastolic_Peak': 0.0812
Skewness of 'Pulse_Area': 0.3989
Skewness of 'index': 0.0101
Skewness of 'Gender': -1.3466
Skewness of 'Glucose_level': 0.5332
Skewness of 'Height': -0.2644
Skewness of 'Weight': 0.7997
Skewness of 'pl': 0.1683
```

```python
df
```

```
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
505.0
2              511           1        77.0          522.0
505.0
3              511           1        77.0          522.0
505.0
4              511           1        77.0          522.0
505.0
...            ...         ...         ...            ...
...
844941         513          23        83.0          516.0
510.0
```

```
844942          513           23        83.0           516.0
510.0
844943          513           23        83.0           516.0
510.0
844944          513           23        83.0           516.0
510.0
844945          513           23        83.0           516.0
510.0

         Pulse_Area  index  Gender  Glucose_level  Height  Weight
pl  \
0             393.0      0       1             99     180      53
1
1             393.0      1       1            102     180      53
2
2             393.0      2       1            103     180      53
3
3             393.0      3       1            128     180      53
4
4             393.0      4       1            130     180      53
5
...             ...    ...     ...            ...     ...     ...
...
844941        366.0     43       1            108     173      57
1463368
844942        366.0     42       1            100     173      57
1463369
844943        366.0     43       1            108     173      57
1463370
844944        366.0     42       1            100     173      57
1463371
844945        366.0     43       1            108     173      57
1463372

        Age Range
0               2
1               2
2               2
3               2
4               2
...           ...
844941          1
844942          1
844943          1
844944          1
844945          1

[744974 rows x 13 columns]
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
import seaborn as sns
import matplotlib.pyplot as plt

import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title('Correlation Matrix')
plt.show()
```



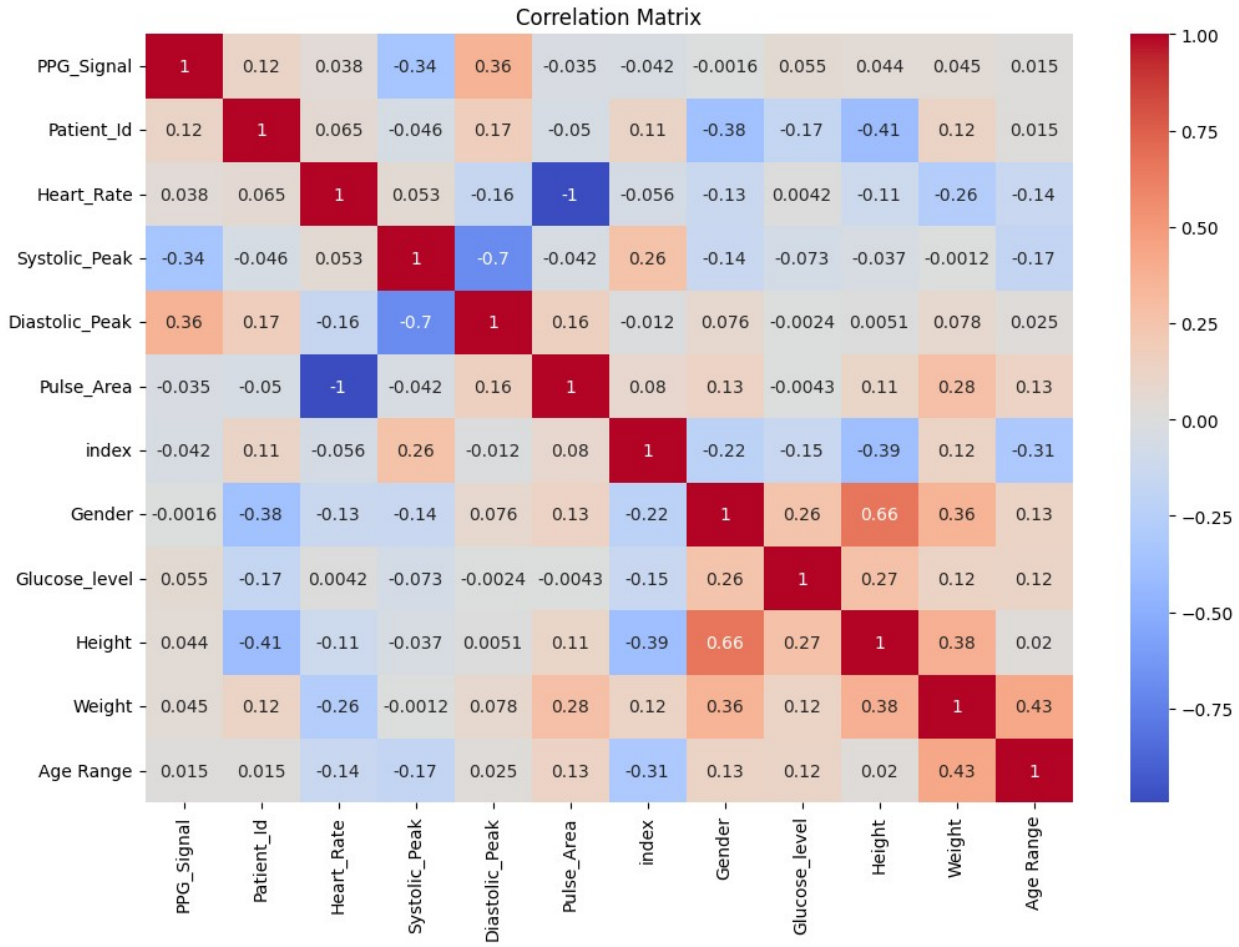

```
df.columns

Index(['PPG_Signal', 'Patient_Id', 'Heart_Rate', 'Systolic_Peak',
       'Diastolic_Peak', 'Pulse_Area', 'index', 'Gender',
```

```
'Glucose_level',
       'Height', 'Weight', 'pl', 'Age Range'],
      dtype='object')

# Drop columns without modifying the original DataFrame
#merged_df1 = merged_df1.drop(columns=['pl'])
# Drop columns without modifying the original DataFrame
df = df.drop(columns=['pl',])

# Check the result
#df.head()

df
```

```
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
505.0
2              511           1        77.0          522.0
505.0
3              511           1        77.0          522.0
505.0
4              511           1        77.0          522.0
505.0
...            ...         ...         ...            ...
...
844941         513          23        83.0          516.0
510.0
844942         513          23        83.0          516.0
510.0
844943         513          23        83.0          516.0
510.0
844944         513          23        83.0          516.0
510.0
844945         513          23        83.0          516.0
510.0

        Pulse_Area  index  Gender  Glucose_level  Height  Weight Age
Range
0            393.0      0       1             99     180      53
2
1            393.0      1       1            102     180      53
2
2            393.0      2       1            103     180      53
2
3            393.0      3       1            128     180      53
2
4            393.0      4       1            130     180      53
```

```
2
...            ...    ...     ...            ...     ...     ...
...
844941       366.0     43       1            108     173      57
1
844942       366.0     42       1            100     173      57
1
844943       366.0     43       1            108     173      57
1
844944       366.0     42       1            100     173      57
1
844945       366.0     43       1            108     173      57
1

[744974 rows x 12 columns]
```

```python
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title('Correlation Matrix')
plt.show()
```




```
df.columns
```

```
Index(['PPG_Signal', 'Patient_Id', 'Heart_Rate', 'Systolic_Peak',
       'Diastolic_Peak', 'Pulse_Area', 'index', 'Gender',
'Glucose_level',
       'Height', 'Weight', 'Age Range'],
      dtype='object')
```

```
# Drop columns without modifying the original DataFrame
#df = df.drop(columns=['Systolic_Peak'])

# Check the result
#df.head()



# Apply one-hot encoding to the 'Gender' column
#df = pd.get_dummies(df, columns=['Gender','Age Range'],dtype=int)

# Display the result
df
```

```
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
505.0
2              511           1        77.0          522.0
505.0
3              511           1        77.0          522.0
505.0
4              511           1        77.0          522.0
505.0
...            ...         ...         ...            ...
...
844941         513          23        83.0          516.0
510.0
844942         513          23        83.0          516.0
510.0
844943         513          23        83.0          516.0
510.0
844944         513          23        83.0          516.0
510.0
844945         513          23        83.0          516.0
510.0

        Pulse_Area  index  Gender  Glucose_level  Height  Weight Age
Range
0            393.0      0       1             99     180      53
2
1            393.0      1       1            102     180      53
2
2            393.0      2       1            103     180      53
2
3            393.0      3       1            128     180      53
2
4            393.0      4       1            130     180      53
2
...            ...    ...     ...            ...     ...     ...
...
844941       366.0     43       1            108     173      57
1
844942       366.0     42       1            100     173      57
1
844943       366.0     43       1            108     173      57
1
844944       366.0     42       1            100     173      57
1
844945       366.0     43       1            108     173      57
1
```

```
[744974 rows x 12 columns]
```

```
# Check for missing values in each column
missing_values = df.isnull().sum()

# Print the count of missing values for each column
print("Missing Values in Each Column:")
print(missing_values)
```

```
Missing Values in Each Column:
PPG_Signal        0
Patient_Id        0
Heart_Rate        0
Systolic_Peak     0
Diastolic_Peak    0
Pulse_Area        0
index             0
Gender            0
Glucose_level     0
Height            0
Weight            0
Age Range         0
dtype: int64
```

```
df
```

```
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
505.0
2              511           1        77.0          522.0
505.0
3              511           1        77.0          522.0
505.0
4              511           1        77.0          522.0
505.0
...            ...         ...         ...            ...
...
844941         513          23        83.0          516.0
510.0
844942         513          23        83.0          516.0
510.0
844943         513          23        83.0          516.0
510.0
844944         513          23        83.0          516.0
510.0
844945         513          23        83.0          516.0
```

```
510.0

        Pulse_Area  index  Gender  Glucose_level  Height  Weight Age
Range
0            393.0      0       1             99     180      53
2
1            393.0      1       1            102     180      53
2
2            393.0      2       1            103     180      53
2
3            393.0      3       1            128     180      53
2
4            393.0      4       1            130     180      53
2
...            ...    ...     ...            ...     ...     ...
...
844941       366.0     43       1            108     173      57
1
844942       366.0     42       1            100     173      57
1
844943       366.0     43       1            108     173      57
1
844944       366.0     42       1            100     173      57
1
844945       366.0     43       1            108     173      57
1

[744974 rows x 12 columns]
```

```python
# Step 5: Apply inverse transformation to convert the scaled data back
to its original values
#df_original = scaler.inverse_transform(df)

# Step 6: Convert back to a DataFrame with original column names

#df_original = pd.DataFrame(df_original, columns=df.columns)
#print("\nOriginal Dataset after Inverse Transformation:\n",
df_original)

#df_original

X = df.drop(['Glucose_level'], axis=1)
Y = df[['Glucose_level']]

X
```

```
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
```

```
505.0
2               511            1        77.0          522.0
505.0
3               511            1        77.0          522.0
505.0
4               511            1        77.0          522.0
505.0
...             ...          ...         ...            ...
...
844941          513           23        83.0          516.0
510.0
844942          513           23        83.0          516.0
510.0
844943          513           23        83.0          516.0
510.0
844944          513           23        83.0          516.0
510.0
844945          513           23        83.0          516.0
510.0
```

	Pulse_Area	index	Gender	Height	Weight	Age Range
0	393.0	0	1	180	53	2
1	393.0	1	1	180	53	2
2	393.0	2	1	180	53	2
3	393.0	3	1	180	53	2
4	393.0	4	1	180	53	2
...	...	...	...	...	...	...
844941	366.0	43	1	173	57	1
844942	366.0	42	1	173	57	1
844943	366.0	43	1	173	57	1
844944	366.0	42	1	173	57	1
844945	366.0	43	1	173	57	1

```
[744974 rows x 11 columns]
```

```
X.columns
```

```
Index(['PPG_Signal', 'Patient_Id', 'Heart_Rate', 'Systolic_Peak',
       'Diastolic_Peak', 'Pulse_Area', 'index', 'Gender', 'Height',
'Weight',
       'Age Range'],
      dtype='object')
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size =
0.20, random_state = 0)
```

Min Max Scaler

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Initialize the MinMaxScaler
scaler_features = MinMaxScaler()
scaler_target = MinMaxScaler()
```

Fit the Scaler Only on the Training Set:

```
X_train_scaled = scaler_features.fit_transform(X_train)

X_train_scaled
```

```
array([[0.5       , 0.        , 0.5       , ..., 0.76666667,
0.05660377,
        0.25      ],
       [0.625     , 1.        , 0.28125   , ..., 0.53333333,
0.13207547,
        0.        ],
       [0.5       , 0.18181818, 0.625     , ..., 0.6       ,
0.11320755,
        0.        ],
       ...,
       [0.5       , 0.        , 0.46875   , ..., 0.76666667,
0.05660377,
        0.25      ],
       [0.375     , 0.45454545, 0.6875    , ..., 0.83333333,
0.22641509,
        0.        ],
       [0.125     , 0.22727273, 0.59375   , ..., 0.26666667,
0.18867925,
        0.25      ]])
```

Transform the Test Set Using the Same Scaler:

```
X_test_scaled = scaler_features.transform(X_test)

X_test_scaled
```

```
array([[0.25      , 0.18181818, 0.5625    , ..., 0.6       ,
0.11320755,
        0.        ],
       [0.5       , 0.95454545, 1.        , ..., 0.43333333, 0.
,
        0.        ],
       [0.625     , 0.36363636, 0.5625    , ..., 0.5       ,
0.18867925,
        0.        ],
       ...,
```

```
       [0.625     , 0.04545455, 0.75      , ..., 1.        ,
0.47169811,
        0.        ],
       [0.375     , 0.45454545, 0.71875   , ..., 0.83333333,
0.22641509,
        0.        ],
       [0.5       , 0.18181818, 0.5625    , ..., 0.6       ,
0.11320755,
        0.        ]])
```

scale for the y_train and y_test

```
# Scale the target (Glucose_level) for y_train and y_test
y_train_scaled = scaler_target.fit_transform(y_train)
y_test_scaled = scaler_target.transform(y_test)

y_train_scaled
```

```
array([[0.82758621],
       [0.34482759],
       [0.37931034],
       ...,
       [0.25862069],
       [0.43103448],
       [0.13793103]])
```

```
y_test_scaled
```

```
array([[0.82758621],
       [0.62068966],
       [0.62068966],
       ...,
       [0.39655172],
       [0.20689655],
       [0.13793103]])
```

Apply Algorithms on dataset

Linear Regression Model

```
#write code here
from sklearn.linear_model import LinearRegression

#write code here
lr = LinearRegression()
lr.fit(X_train_scaled,y_train_scaled)
```

```
LinearRegression()
```

```
#write code here
lr_predict = lr.predict(X_test_scaled)

lr_predict
```

```
array([[0.41185275],
       [0.3215901 ],
       [0.33928684],
       ...,
       [0.50345417],
       [0.4803961 ],
       [0.42643983]])
```

Inverse transform X_test, y_test, and the predictions back to their original values

```
# Step 8: Inverse transform X_test, y_test, and the predictions back
to their original values
X_test_original = scaler_features.inverse_transform(X_test_scaled)
y_test_original = scaler_target.inverse_transform(y_test_scaled)
y_pred_original = scaler_target.inverse_transform(lr_predict)

X_test_original
```

```
array([[509.,   5.,  79., ..., 175.,  56.,   1.],
       [511.,  22.,  93., ..., 170.,  50.,   1.],
       [512.,   9.,  79., ..., 172.,  60.,   1.],
       ...,
       [512.,   2.,  85., ..., 187.,  75.,   1.],
       [510.,  11.,  84., ..., 182.,  62.,   1.],
       [511.,   5.,  79., ..., 175.,  56.,   1.]])
```

```
y_test_original
```

```
array([[136.],
       [124.],
       [124.],
       ...,
       [111.],
       [100.],
       [ 96.]])
```

```
y_pred_original
```

```
array([[111.88745976],
       [106.65222568],
       [107.6786369 ],
       ...,
       [117.20034192],
       [115.86297381],
       [112.73351039]])
```

combine dataset and show

```
df.columns
```

```
Index(['PPG_Signal', 'Patient_Id', 'Heart_Rate', 'Systolic_Peak',
       'Diastolic_Peak', 'Pulse_Area', 'index', 'Gender',
'Glucose_level',
       'Height', 'Weight', 'Age Range'],
      dtype='object')
```

```
# Step 9: Create a DataFrame for comparison of actual and predicted
values in the original scale
# Make sure that the number of columns in np.column_stack matches the
number of columns in the column names
#comparison_df = pd.DataFrame(
    #np.column_stack((X_test_original, y_test_original.flatten(),
y_pred_original.flatten())),
    #columns=[
        #'PPG_Signal', 'Systolic_Peak', 'Diastolic_Peak',
'Pulse_Area', 'index',
        #'Height', 'Weight', 'pl', 'Gender_0', 'Gender_1',
        #'Age Range_1', 'Age Range_2', 'Age Range_3', 'Age Range_4',
        #'Age Range_5', 'Patient_Id_1', 'Patient_Id_2',
'Patient_Id_3',
        #'Patient_Id_5', 'Patient_Id_6', 'Patient_Id_7',
'Patient_Id_8',
        #'Patient_Id_9', 'Patient_Id_10', 'Patient_Id_11',
'Patient_Id_12',
        #'Patient_Id_13', 'Patient_Id_14', 'Patient_Id_15',
'Patient_Id_16',
        #'Patient_Id_17', 'Patient_Id_18', 'Patient_Id_19',
'Patient_Id_21',
        #'Patient_Id_22', 'Patient_Id_23', 'Actual Glucose Level',
'Predicted Glucose Level'
    #]
#)

# Step 10: Display the comparison of actual and predicted values in
the original scale
#print("\nComparison of Actual and Predicted Glucose Levels (Original
Scale):\n", comparison_df)

#comparison_df
```

Evaluate Using Performance Metrics

Mean Absolute Error

Mean Absolute Error (MAE) in the context of a linear regression model is a performance metric that measures the average magnitude of the errors between the predicted values and the actual values. It is used to quantify how close the predictions of your model are to the actual outcomes.

- Lower MAE is better:

The smaller the MAE, the closer your predicted values are to the actual values. A perfect model would have an MAE of 0.

```
from sklearn.metrics import mean_absolute_error

mae = mean_absolute_error(y_test_scaled,lr_predict)
print(f"Mean Absolute Error of the tesing data : {mae}")
```

```
Mean Absolute Error of the tesing data : 0.19709807189448006
```

Mean Squared Error

Mean Squared Error (MSE) in the context of a linear regression model is a commonly used performance metric that measures the average of the squared differences between the predicted values and the actual values. It provides a way to quantify how well your regression model predicts the dependent variable.

A very low MSE on training data but high MSE on testing data often indicates overfitting — the model has learned the training data too well, including noise, and cannot generalize well to new data. In contrast, if MSE is high on both training and testing datasets, it may indicate underfitting, meaning the model is too simple to capture the underlying pattern.

The closer the MSE is to zero, the more accurate the model is, implying that the predicted values are very close to the actual values.

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_test_scaled,lr_predict)
print(f"Mean Squared Error of testing data : {mse}")
```

```
Mean Squared Error of testing data : 0.05703854735623502
```

Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is a popular metric for evaluating the performance of a linear regression model. It measures the average error between the predicted and actual values in the same units as the target variable (in this case, glucose levels). RMSE is essentially the square root of the Mean Squared Error (MSE), and it is widely used because it is more interpretable than MSE, as it provides the error in the same scale as the predicted values.

RMSE can be used to compare different models. When comparing models on the same dataset, the model with the lower RMSE is generally better at making accurate predictions.

```
# Calculate Mean Squared Error
#mse = mean_squared_error(y_test,lr_predict)
import numpy as np
# Calculate Root Mean Squared Error
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error of testing data : {rmse}")
```

```
Root Mean Squared Error of testing data : 0.23882744263638345
```

R-squared (R^2)

R-squared (R^2), also known as the coefficient of determination, is a statistical measure used to evaluate how well a linear regression model explains the variance in the dependent variable (in your case, glucose levels). It provides insight into how well the independent variables in the model are predicting the target variable.

Interpretation

R^2 =0

The model explains none of the variability of the target variable. In other words, the model's predictions are no better than simply predicting the mean of the target variable for all observations.

R^2 = 1

The model explains all the variability of the target variable, meaning it perfectly predicts the outcome.

R^2 between 0 and 1:

The model explains part of the variability of the outcome. The closer is to 1, the better the model fits the data.

For example, if

R^2 = 0.85

this means that 85% of the variance in the target variable (glucose levels) is explained by the model, while the remaining 15% is unexplained (due to factors not captured by the model).

R2_score

```
#write code here
#mean square error
from sklearn.metrics import r2_score
lr_r2_test= r2_score(y_test_scaled,lr_predict)
```

```
#print('R2 score for Linear Regression Testing Data is: ', lr_r2_train)
print('R2 score for Linear Regression Testing Data is: ', lr_r2_test)
```

```
R2 score for Linear Regression Testing Data is:  0.10819721957201589
```

Random Forest

```
# Importing the model from sklearn
from sklearn.ensemble import RandomForestRegressor

# Making instance and training the model
rf_reg = RandomForestRegressor(n_estimators=100, max_depth=5, random_state=42)

rf_reg.fit(X_train_scaled,y_train_scaled)
```

```
/tmp/ipykernel_93/3160336221.py:1: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  rf_reg.fit(X_train_scaled,y_train_scaled)
```

```
RandomForestRegressor(max_depth=5, random_state=42)
```

```
y_pred = rf_reg.predict(X_test_scaled)

# Evaluate the model's performance
mse3 = mean_squared_error(y_test_scaled, y_pred)
rmse3 = mean_squared_error(y_test_scaled, y_pred, squared=False)
mae3 = mean_absolute_error(y_test_scaled, y_pred)
r2 = r2_score(y_test_scaled, y_pred)

# Print evaluation metrics
print(f"Mean Squared Error (MSE): {mse3:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse3:.2f}")
print(f"Mean Absolute Error (MAE): {mae3:.2f}")
print(f"R-squared (R2): {r2:.2f}")
```

```
Mean Squared Error (MSE): 0.03
Root Mean Squared Error (RMSE): 0.17
Mean Absolute Error (MAE): 0.11
R-squared (R2): 0.54
```

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeRegressor

# Initialize the DecisionTreeRegressor (similar to DecisionTreeClassifier style)
lf_dt = DecisionTreeRegressor(max_depth=3, criterion='squared_error', random_state=100)
```

```
# Train the model on the training data
lf_dt.fit(X_train_scaled,y_train_scaled)
```

```
DecisionTreeRegressor(max_depth=3, random_state=100)
```

```
# Predict the target values for the test set
y_pred = lf_dt.predict(X_test_scaled)

# Evaluate the model's performance
mse4 = mean_squared_error(y_test_scaled, y_pred)
rmse4 = mean_squared_error(y_test_scaled, y_pred, squared=False)
mae4 = mean_absolute_error(y_test_scaled, y_pred)
r2 = r2_score(y_test_scaled, y_pred)

# Print evaluation metrics
print(f"Mean Squared Error (MSE): {mse4:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse4:.2f}")
print(f"Mean Absolute Error (MAE): {mae4:.2f}")
print(f"R-squared (R2): {r2:.2f}")
```

```
Mean Squared Error (MSE): 0.04
Root Mean Squared Error (RMSE): 0.21
Mean Absolute Error (MAE): 0.15
R-squared (R2): 0.33
```

lag.

```

```

Polynomial Features

This helps models capture non-linear relationships between the input features and the target variable, making simple linear models more expressive and able to approximate more complex patterns

```
#write code here
from sklearn.preprocessing import PolynomialFeatures

#write code here
poly_reg = PolynomialFeatures(degree=2)
```

Polynomial Feature Transformation

1- fit_transform()

- Purpose:

Combines the operations of fit() and transform() into one step.

- Usage:

Used when you need to compute the parameters (like mean, variance, etc.) of the transformation based on the training data, and then apply the transformation.

- Process:

Fit: Calculates the parameters required for the transformation (e.g., scaling factors). Transform: Applies the transformation using these parameters to the given data.

```
X_train_poly = poly_reg.fit_transform(X_train_scaled)
```

2- transform()

- Purpose: Applies a transformation to data using parameters that were computed from fit().
- Usage: Used when you already have the parameters from fit() (typically on training data) and you want to apply the same transformation to new data, like test data or unseen data.

```
X_test_poly = poly_reg.transform(X_test_scaled)
```

fit_transform() is used on the training dataset to learn the parameters and apply the transformation.

transform() is used on the test or unseen dataset to apply the learned transformation, ensuring consistency.

Using fit_transform() on the training data and transform() on the test data helps prevent data leakage, where information from the test set influences the model during training.

model training

```
#write code here

poly_model = LinearRegression()
poly_model.fit(X_train_poly, y_train_scaled)
```

```
LinearRegression()
```

```
#y_train_predicted = poly_model.predict(X_train_poly)
```

Model Prediction

```
y_test_predict  = poly_model.predict(X_test_poly)

#write code here
#y_train_predicted,y_test_predict =
poly_model.predict(X_train_poly),poly_model.predict(X_test_poly)
```

ploynomial Regression matrix performance

```
#from sklearn.metrics import mean_absolute_error

mae = mean_absolute_error(y_test_scaled,y_test_predict)
print(f"Mean Absolute Error of the tesing data : {mae}")
```

```
Mean Absolute Error of the tesing data : 0.06893090280093409
```

```
#from sklearn.metrics import mean_squared_error

mse1 = mean_squared_error(y_test_scaled,y_test_predict)
print(f"Mean Absolute Error of testing data : {mse}")
```

```
Mean Absolute Error of testing data : 0.05703854735623502
```

```
#import numpy as np
# Calculate Root Mean Squared Error
rmse = np.sqrt(mse1)
print(f"Root Mean Squared Error of testing data : {rmse}")
```

```
Root Mean Squared Error of testing data : 0.09313701562682071
```

```
r2_test = r2_score(y_test_scaled, y_test_predict)

#print('R2 score for Linear Regression Testing Data is: ',
lr_r2_train)

print('R2 score for Poly Regression  Testing Data is: ', r2_test)
```

```
R2 score for Poly Regression  Testing Data is:  0.8643733604885401
```

```
y_test_scaled
```

```
array([[0.82758621],
       [0.62068966],
       [0.62068966],
       ...,
       [0.39655172],
       [0.20689655],
       [0.13793103]])
```

```
y_test_predict
```

```
array([[0.78390151],
       [0.65652239],
       [0.55547893],
       ...,
       [0.41847203],
       [0.23054237],
       [0.18168164]])
```

```
# Inverse transform the scaled values
y_test_actual = scaler_target .inverse_transform(y_test_scaled)
y_test_predict_actual =
scaler_target .inverse_transform(y_test_predict)


# Show the model predictions vs actual values
y_test_actual = pd.Series(y_test_actual.flatten(), name='Actual
Glucose Level')
y_test_predict_actual = pd.Series(y_test_predict_actual.flatten(),
name='Predicted Glucose Level')
predictions = pd.DataFrame({'Actual': y_test_actual, 'Predicted':
y_test_predict_actual})
predictions
```

```
        Actual   Predicted
0        136.0  133.466288
1        124.0  126.078298
2        124.0  120.217778
3        136.0  123.102675
4         96.0   91.423398
...        ...         ...
148990   108.0  111.327062
148991   108.0  111.327062
148992   111.0  112.271378
148993   100.0  101.371457
148994    96.0   98.537535

[148995 rows x 2 columns]
```

after polynominal feature technique then make 2d data format

```
#only calculating feature

feature_names = poly_reg.get_feature_names_out()
#print("Feature names:", feature_names)

X_train_poly_2d = pd.DataFrame(X_train_poly, columns=feature_names)
X_train_poly_2d
```

```
          1     x0        x1       x2        x3     x4        x5
x6  \
0       1.0  0.500  0.000000  0.50000  0.400000  0.500  0.491279
0.090909
1       1.0  0.625  1.000000  0.28125  0.066667  0.875  0.747093
0.651515
2       1.0  0.500  0.181818  0.62500  0.400000  0.375  0.386628
0.818182
3       1.0  0.625  0.409091  1.00000  0.200000  0.625  0.000000
0.121212
```

```
4       1.0  0.500  0.000000  0.56250  0.400000  0.375  0.438953
0.075758
...     ...    ...       ...      ...       ...    ...       ...
...
595974  1.0  0.375  0.272727  0.00000  0.666667  0.625  1.000000
0.939394
595975  1.0  0.500  0.045455  0.28125  0.333333  0.375  0.747093
0.484848
595976  1.0  0.500  0.000000  0.46875  0.533333  0.375  0.520349
0.030303
595977  1.0  0.375  0.454545  0.68750  0.600000  0.625  0.328488
0.181818
595978  1.0  0.125  0.227273  0.59375  0.666667  0.000  0.404070
0.863636

         x7        x8  ...  x7^2     x7 x8     x7 x9  x7 x10      x8^2
\
0       1.0  0.766667  ...   1.0  0.766667  0.056604    0.25  0.587778

1       1.0  0.533333  ...   1.0  0.533333  0.132075    0.00  0.284444

2       1.0  0.600000  ...   1.0  0.600000  0.113208    0.00  0.360000

3       1.0  0.733333  ...   1.0  0.733333  0.754717    0.25  0.537778

4       1.0  0.766667  ...   1.0  0.766667  0.056604    0.25  0.587778

...     ...       ...  ...   ...       ...       ...     ...       ...

595974  1.0  0.733333  ...   1.0  0.733333  0.811321    0.25  0.537778

595975  1.0  1.000000  ...   1.0  1.000000  0.471698    0.00  1.000000

595976  1.0  0.766667  ...   1.0  0.766667  0.056604    0.25  0.587778

595977  1.0  0.833333  ...   1.0  0.833333  0.226415    0.00  0.694444

595978  0.0  0.266667  ...   0.0  0.000000  0.000000    0.00  0.071111


           x8 x9    x8 x10      x9^2    x9 x10   x10^2
0       0.043396  0.191667  0.003204  0.014151  0.0625
1       0.070440  0.000000  0.017444  0.000000  0.0000
2       0.067925  0.000000  0.012816  0.000000  0.0000
3       0.553459  0.183333  0.569598  0.188679  0.0625
4       0.043396  0.191667  0.003204  0.014151  0.0625
...          ...       ...       ...       ...     ...
595974  0.594969  0.183333  0.658241  0.202830  0.0625
595975  0.471698  0.000000  0.222499  0.000000  0.0000
595976  0.043396  0.191667  0.003204  0.014151  0.0625
```

```
595977  0.188679  0.000000  0.051264  0.000000  0.0000
595978  0.050314  0.066667  0.035600  0.047170  0.0625

[595979 rows x 78 columns]
```

```
X_test_poly_2d = pd.DataFrame(X_test_poly, columns=feature_names)
X_test_poly_2d
```

```
           1     x0        x1       x2        x3     x4        x5
x6  \
0       1.0  0.250  0.181818  0.56250  0.466667  0.375  0.456395
0.848485
1       1.0  0.500  0.954545  1.00000  0.800000  0.125  0.084302
0.621212
2       1.0  0.625  0.363636  0.56250  0.133333  0.750  0.462209
1.000000
3       1.0  0.875  0.772727  1.00000  0.600000  0.750  0.000000
0.409091
4       1.0  0.125  0.227273  0.43750  0.666667  0.000  0.555233
0.863636
...     ...    ...       ...      ...       ...    ...       ...
...
148990  1.0  0.375  0.727273  0.68750  0.333333  0.375  0.322674
0.378788
148991  1.0  0.375  0.727273  0.68750  0.333333  0.375  0.322674
0.378788
148992  1.0  0.625  0.045455  0.75000  0.466667  0.625  0.264535
0.484848
148993  1.0  0.375  0.454545  0.71875  0.333333  0.250  0.311047
0.151515
148994  1.0  0.500  0.181818  0.56250  0.466667  0.375  0.456395
0.787879

          x7        x8  ...  x7^2     x7 x8     x7 x9  x7 x10      x8^2
\
0       1.0  0.600000  ...   1.0  0.600000  0.113208     0.0  0.360000

1       0.0  0.433333  ...   0.0  0.000000  0.000000     0.0  0.187778

2       0.0  0.500000  ...   0.0  0.000000  0.000000     0.0  0.250000

3       1.0  0.766667  ...   1.0  0.766667  0.754717     0.0  0.587778

4       0.0  0.266667  ...   0.0  0.000000  0.000000     0.0  0.071111

...     ...       ...  ...   ...       ...       ...     ...       ...

148990  0.0  0.400000  ...   0.0  0.000000  0.000000     0.0  0.160000

148991  0.0  0.400000  ...   0.0  0.000000  0.000000     0.0  0.160000
```

```
148992  1.0  1.000000  ...   1.0  1.000000  0.471698     0.0  1.000000

148993  1.0  0.833333  ...   1.0  0.833333  0.226415     0.0  0.694444

148994  1.0  0.600000  ...   1.0  0.600000  0.113208     0.0  0.360000


           x8 x9    x8 x10      x9^2   x9 x10   x10^2
0       0.067925  0.000000  0.012816  0.00000  0.0000
1       0.000000  0.000000  0.000000  0.00000  0.0000
2       0.094340  0.000000  0.035600  0.00000  0.0000
3       0.578616  0.000000  0.569598  0.00000  0.0000
4       0.050314  0.066667  0.035600  0.04717  0.0625
...          ...       ...       ...      ...     ...
148990  0.037736  0.000000  0.008900  0.00000  0.0000
148991  0.037736  0.000000  0.008900  0.00000  0.0000
148992  0.471698  0.000000  0.222499  0.00000  0.0000
148993  0.188679  0.000000  0.051264  0.00000  0.0000
148994  0.067925  0.000000  0.012816  0.00000  0.0000

[148995 rows x 78 columns]

y_train_scaled

array([[0.82758621],
       [0.34482759],
       [0.37931034],
       ...,
       [0.25862069],
       [0.43103448],
       [0.13793103]])

y_test_scaled

array([[0.82758621],
       [0.62068966],
       [0.62068966],
       ...,
       [0.39655172],
       [0.20689655],
       [0.13793103]])

y_test_predict

array([[0.78390151],
       [0.65652239],
       [0.55547893],
       ...,
       [0.41847203],
       [0.23054237],
       [0.18168164]])
```

Model input then output

```
X.columns
```

```
Index(['PPG_Signal', 'Patient_Id', 'Heart_Rate', 'Systolic_Peak',
       'Diastolic_Peak', 'Pulse_Area', 'index', 'Gender', 'Height',
'Weight',
       'Age Range'],
      dtype='object')
```

```
X
```

```
        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
0              511           1        77.0          522.0
505.0
1              511           1        77.0          522.0
505.0
2              511           1        77.0          522.0
505.0
3              511           1        77.0          522.0
505.0
4              511           1        77.0          522.0
505.0
...            ...         ...         ...            ...
...
844941         513          23        83.0          516.0
510.0
844942         513          23        83.0          516.0
510.0
844943         513          23        83.0          516.0
510.0
844944         513          23        83.0          516.0
510.0
844945         513          23        83.0          516.0
510.0

        Pulse_Area  index  Gender  Height  Weight Age Range
0            393.0      0       1     180      53         2
1            393.0      1       1     180      53         2
2            393.0      2       1     180      53         2
3            393.0      3       1     180      53         2
4            393.0      4       1     180      53         2
...            ...    ...     ...     ...     ...       ...
844941       366.0     43       1     173      57         1
844942       366.0     42       1     173      57         1
844943       366.0     43       1     173      57         1
844944       366.0     42       1     173      57         1
844945       366.0     43       1     173      57         1

[744974 rows x 11 columns]
```

```
Y

        Glucose_level
0                  99
1                 102
2                 103
3                 128
4                 130
...               ...
844941            108
844942            100
844943            108
844944            100
844945            108

[744974 rows x 1 columns]
```

511 1 77.0 522.0 505.0 393.0 0 1 180 53 2

'PPG_Signal', 'Patient_Id', 'Heart_Rate', 'Systolic_Peak', 'Diastolic_Peak', 'Pulse_Area', 'index', 'Gender', 'Height', 'Weight', 'Age Range'

```
X_test

        PPG_Signal  Patient_Id  Heart_Rate  Systolic_Peak
Diastolic_Peak  \
305947         509           5        79.0          521.0
507.0
765658         511          22        93.0          526.0
505.0
413976         512           9        79.0          516.0
510.0
649421         514          18        93.0          523.0
510.0
335190         508           6        75.0          524.0
504.0
...            ...         ...         ...            ...
...
632822         510          17        83.0          519.0
507.0
633890         510          17        83.0          519.0
507.0
173720         512           2        85.0          521.0
509.0
455922         510          11        84.0          519.0
506.0
308325         511           5        79.0          521.0
507.0

        Pulse_Area  index  Gender  Height  Weight Age Range
```

```
305947       388.0     56       1     175      56          1
765658       324.0     41       0     170      50          1
413976       389.0     66       0     172      60          1
649421       309.5     27       1     180      90          1
335190       405.0     57       0     165      60          2
...            ...    ...     ...     ...     ...        ...
632822       365.0     25       0     169      55          1
633890       365.0     25       0     169      55          1
173720       355.0     32       1     187      75          1
455922       363.0     10       1     182      62          1
308325       388.0     52       1     175      56          1

[148995 rows x 11 columns]
```

y_test

```
        Glucose_level
305947            136
765658            124
413976            124
649421            136
335190             96
...               ...
632822            108
633890            108
173720            111
455922            100
308325             96

[148995 rows x 1 columns]
```

```
## # Define the column names
column_names = [
    'PPG_Signal(mV)', 'Patient_Id(ID number)', 'Heart_Rate(bpm)',
'Systolic_Peak(mmHg)', 'Diastolic_Peak(mmHg)',
    'Pulse_Area', 'index(integer)','Gender(1 for Male, 0 for Female)',
'Height(cm)', 'Weight(kg)','Age Range[1,2,3,4,5]'
]


# Get input from user (11 values) in a user-friendly way
input_values = []
for column_name in column_names:
    value = float(input(f"Enter value for {column_name}: "))
    input_values.append(value)



# Transform the input and make prediction
input_df = pd.DataFrame([input_values], columns=X.columns)  # Create a
```

```
DataFrame to maintain feature names
input_scaled = scaler_features.transform(input_df)
input_poly = poly_reg.transform(input_scaled)
output_scaled = poly_model.predict(input_poly)

# Convert the prediction back to the original glucose level scale
output = scaler_target.inverse_transform(output_scaled)

# Show the predicted glucose level in original value
print(f"Predicted Glucose Level: {output[0][0]}")




# Show the predicted glucose level in original value
#print(f"Predicted Glucose Level: {output_scaled[0][0]}")

X

df

min(X_test['Weight'])

min(X_test['index'])

min(X_test['PPG_Signal'])

max(X_test['PPG_Signal'])

max(X_test['Patient_Id'])
```