Cairo University
Faculty of Engineering
Healthcare Engineering and Management
Credit Hours System

SBEN429 - Biomedical Data Analytics

Course Final Project

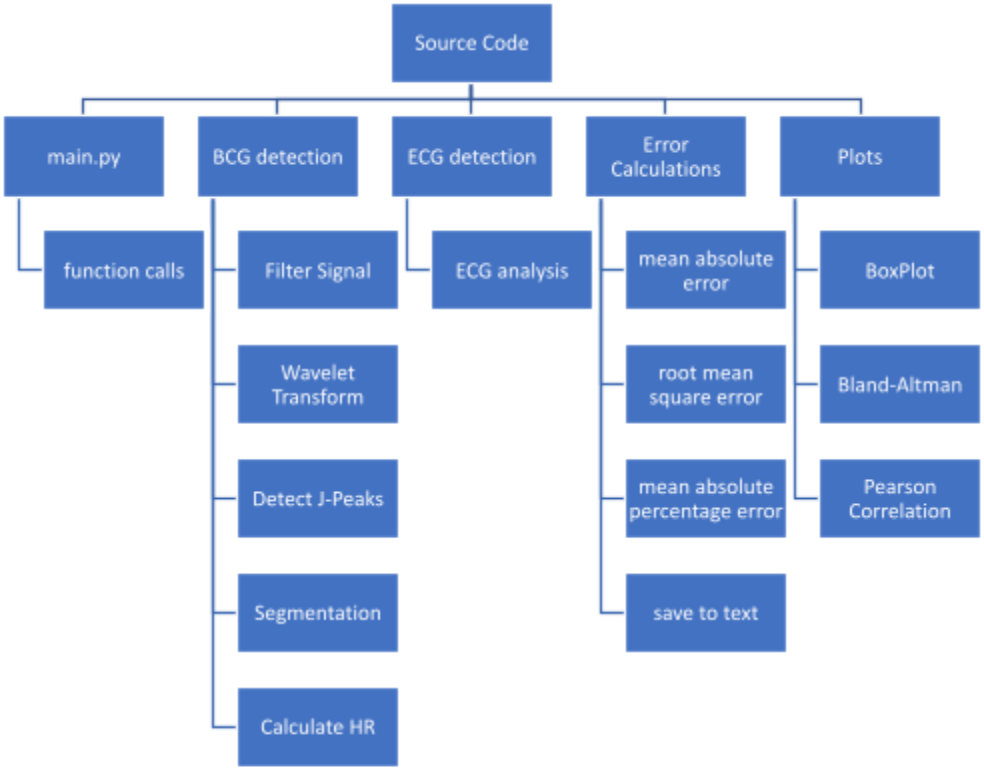
Student Name	Student ID
Zeyad Mansour	1190368
Abdullah Saeed	1190222
Clara Ashraf Moriss Youssef	1190462
Hassan Samy	1190025

Date: 30/12/2022

Project Statement:

A sensor is placed under a subject mattress approximately below a patient's chest and stomach. The sensor measures the mechanical activity of the heart (BCG signal) besides the movement of the chest and stomach. An algorithm is needed to detect the signal, analyze the signal by calculating the errors and finally show a visual representation by three different plots (Bland Altman Plot, Pearson Correlation Plot, Boxplot)

Structure Chart:



Implementation:

The **main()** function imports the dataset and then:

- Loops over all .csv files
- Stores the ECG and BCG data streams of each file (patient)
- Film 1 of the datasets is used as the BCG data
- Down samples the signal from 1000 Hz to 50 Hz
- **ecg_analysis()** and **bcg_analysis()** are called on the respective signals and arrays containing their heart rates are returned
- The heart rates are passed into **errors_calc()**, which returns the MAE, RSME, MAPE
- Heart rates and errors of each patient are saved to **results\output.txt**
- **plots()** draws and saves the three plots in **results** as well

```
def main():

    # assign directory
    directory = 'datasets'
    open('results\output.txt', 'w').close()

    # iterate over all files in the directory
    for filename in os.listdir(directory):
        path = os.path.join(directory, filename)
        # check file validity
        if os.stat(path).st_size != 0:
            rawData = pd.read_csv(path, sep=",", header=None, skiprows=1).values

            # ECG data stored and downsampled from 1000 Hz to 50 Hz
            ecg_data = rawData[:, 0]
            ecg_data = resample(ecg_data, round(50 * len(ecg_data) / 1000))

            # BCG data stored and downsampled from 1000 Hz to 50 Hz
            bcg_data = rawData[:, 1]
            bcg_data = resample(bcg_data, round(50 * len(bcg_data) / 1000))

            # Analyze the two signals and return their respective heart rate estimates
            ecg_hr = ecg_analysis(ecg_data)
            bcg_hr = bcg_analysis(bcg_data)

            # Calculate MAE, RMSE, and MAPE between the two signals
            err1, err2, err3 = errors_calc(ecg_hr, bcg_hr)

            # Save results in .txt file (results/output.txt)
            save_to_txt(os.path.splitext(filename)[0], ecg_hr, bcg_hr, err1, err2, err3)

            # Save plots for selected patient
            if filename == 'X1006.csv':
                plots(ecg_hr, bcg_hr, os.path.splitext(filename)[0])
```

BCG detection and analysis:

Achieved by adapting the provided BCG detection algorithm provided in the project statement and repurposed to this application.

```
def bcg_analysis(data):

    data_stream = data
    utc_time = np.linspace(0, len(data_stream) / 50, len(data_stream))

    start_point, end_point, window_shift, fs = 0, 500, 500, 50
    # ================================================================================================
    data_stream, utc_time = detect_patterns(start_point, end_point, window_shift, data_stream, utc_time, plot=1)

    w = modwt(data_stream, 'bior3.9', 4)
    dc = modwtmra(w, 'bior3.9')
    wavelet_cycle = dc[4]
    # ================================================================================================
    # Vital Signs estimation - (10 seconds window is an optimal size for vital signs measurement)
    t1, t2, window_length, window_shift = 0, 500, 500, 500
    limit = int(math.floor(data_stream.size / window_shift))
    # Heart Rate
    beats = vitals(t1, t2, window_shift, limit, wavelet_cycle, fs, mpd=1, plot=0)

    return beats
```

Preprocessing the imported BCG signal

- **detect_patterns()** returns a filtered signal after removing body movements
- **modwt()** and **modwtmra()** apply further preconditioning using wavelet transform

Detect R-Peaks

- Using **detect_peaks()** to return the j-peaks of the BCG signal

Segmentation

- **vitals()** segments the signal into 10 second windows

HR Calculation

- **compute_rate_unknown_time()** calculates the heart rate of each window using

$$HR_N = \frac{N-1}{t_N} * 60 \text{ where } t_N = \frac{n(k)-n(i)+1}{f_s}$$

- Array of heart rates of all windows is returned

```
def compute_rate_unknown_time(beats, fs, mpd):

    indices = detect_peaks(beats, mpd=mpd)

    diff_sample = indices[-1] - indices[0] + 1
    t_N = diff_sample / fs
    heartRate = (len(indices) - 1) / t_N * 60
    return np.round(heartRate, decimals=2), indices
```

ECG analysis

Repeats the same wavelet transform applied on the BCG signal.

Heartpy's **process_segmentwise()** is then used to segment the signal into 10 second windows, and returns each window's heart rate.

```
def ecg_analysis(data):

    sample_rate = 50

    w = modwt(data, 'bior3.9', 4)
    dc = modwtmra(w, 'bior3.9')
    wavelet_cycle = dc[4]

    # Processes ECG, while segmenting the signal to 10 second windows
    # Returns a dictionary with signal stats
    wd, m = hp.process_segmentwise(wavelet_cycle, sample_rate, 10)

    # Return heartrates
    return [round(x, 1) for x in m['bpm']]
```

Error Calculations

Subprogram name: errors_calc()

Parameters: Calculated ECG and BCG heart rates

Purpose: Calculate mean absolute error, root mean square error , and mean absolute percentage error

Algorithm:

- Initialize all three calculations by zero
- Loop over the size of BCG and ECG outputs (knowing they're equal in length)
- Calculate mean error, its percentage and root mean square by using their equations
- Round the results

Plots

Subprogram name: plots()

Parameters: Calculated ECG and BCG heart rates of one patient

Purpose: Graphic representation of 3 plots (boxplot, Bland-Altman, Pearson Correlation)

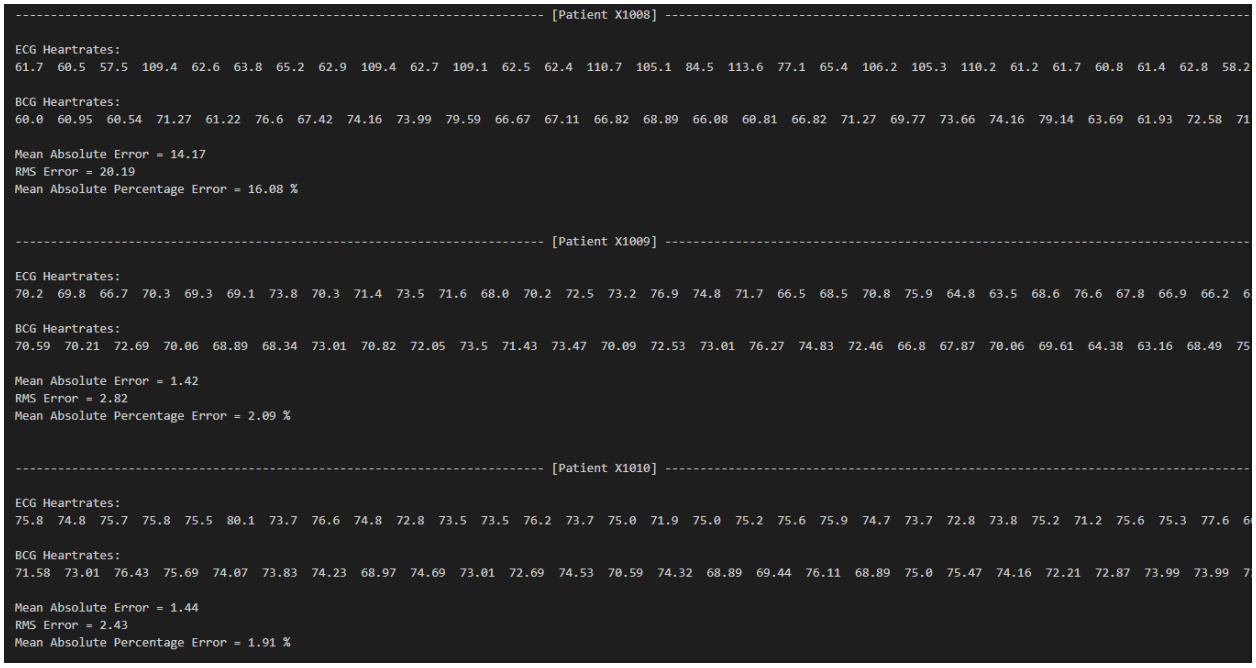
Algorithm:

1- Takes the data as parameters, data consists of ECG and BCG results from previous functions
2- Plot boxplot by a function that takes data and labels as parameters (plt)
3- A function called mean_diff_plot plots Bland Altman plot of data
4- Set a figure size and scatter the data of ECG and BCG results
5- Save the three plots, each in a file with its name in **results/**

Results

All output files and plots are saved in **results/** after program runs.

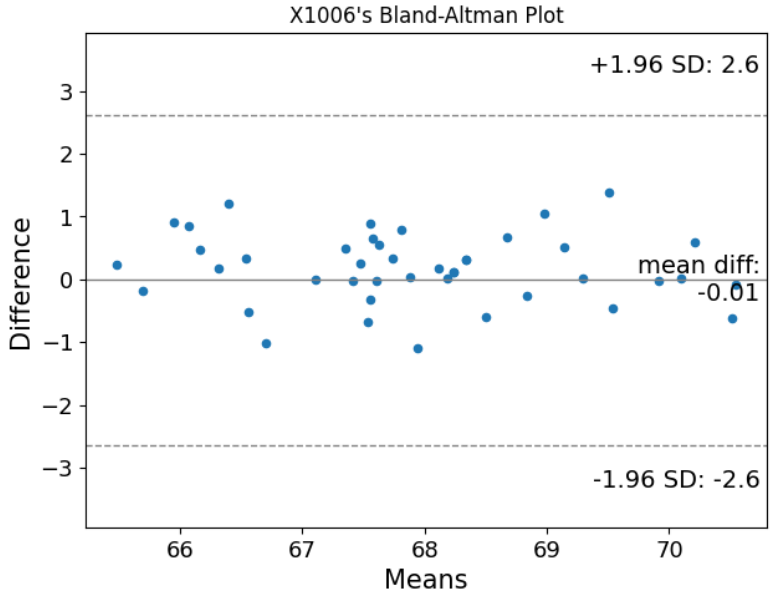
ECG + BCG Heart Rates and Errors

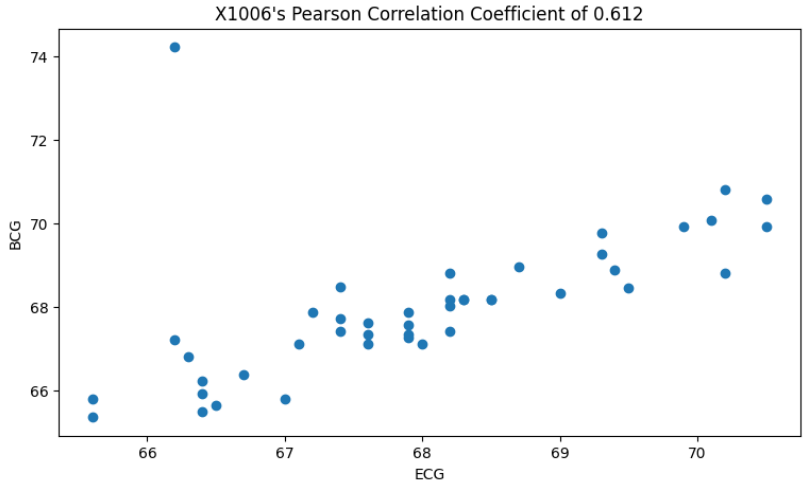


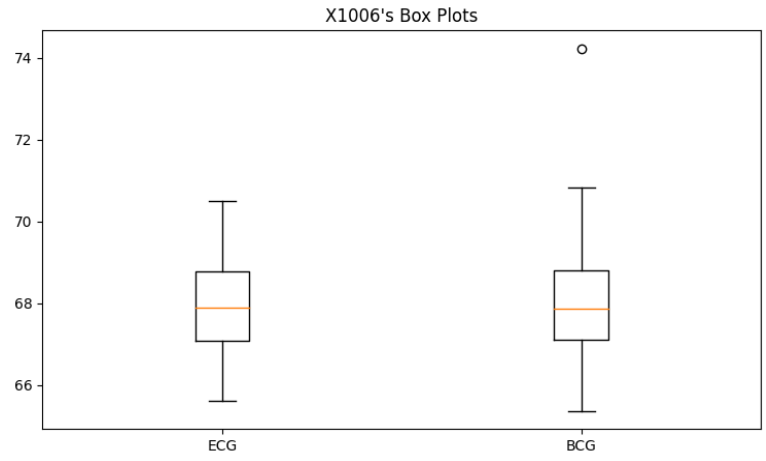
```
------------------------------------------------------------------------ [Patient X1008] ------------------------------------------------------------------------

ECG Heartrates:
61.7  60.5  57.5  109.4  62.6  63.8  65.2  62.9  109.4  62.7  109.1  62.5  62.4  110.7  105.1  84.5  113.6  77.1  65.4  106.2  105.3  110.2  61.2  61.7  60.8  61.4  62.8  58.2

BCG Heartrates:
60.0  60.95  60.54  71.27  61.22  76.6  67.42  74.16  73.99  79.59  66.67  67.11  66.82  68.89  66.08  60.81  66.82  71.27  69.77  73.66  74.16  79.14  63.69  61.93  72.58  71

Mean Absolute Error = 14.17
RMS Error = 20.19
Mean Absolute Percentage Error = 16.08 %


------------------------------------------------------------------------ [Patient X1009] ------------------------------------------------------------------------

ECG Heartrates:
70.2  69.8  66.7  70.3  69.3  69.1  73.8  70.3  71.4  73.5  71.6  68.0  70.2  72.5  73.2  76.9  74.8  71.7  66.5  68.5  70.8  75.9  64.8  63.5  68.6  76.6  67.8  66.9  66.2  6

BCG Heartrates:
70.59  70.21  72.69  70.06  68.89  68.34  73.01  70.82  72.05  73.5  71.43  73.47  70.09  72.53  73.01  76.27  74.83  72.46  66.8  67.87  70.06  69.61  64.38  63.16  68.49  75

Mean Absolute Error = 1.42
RMS Error = 2.82
Mean Absolute Percentage Error = 2.09 %


------------------------------------------------------------------------ [Patient X1010] ------------------------------------------------------------------------

ECG Heartrates:
75.8  74.8  75.7  75.8  75.5  80.1  73.7  76.6  74.8  72.8  73.5  73.5  76.2  73.7  75.0  71.9  75.0  75.2  75.6  75.9  74.7  73.7  72.8  73.8  75.2  71.2  75.6  75.3  77.6  6

BCG Heartrates:
71.58  73.01  76.43  75.69  74.07  73.83  74.23  68.97  74.69  73.01  72.69  74.53  70.59  74.32  68.89  69.44  76.11  68.89  75.0  75.47  74.16  72.21  72.87  73.99  73.99  7

Mean Absolute Error = 1.44
RMS Error = 2.43
Mean Absolute Percentage Error = 1.91 %
```

Sample from **results\output.txt**

Bland-Altman, Pearson Correlation, & Box Plots







Issues Faced:

- The **compute_rate()** function used in the reference algorithm of calculating BCG heart rates depended on a time argument to compute the heart rates, however the dataset we were provided did **not** include timestamps.

  This was solved by implementing, $\mathrm{HR_N} = \frac{\mathrm{N-1}}{\mathrm{t_N}} * 60$ where $\mathrm{t_N} = \frac{\mathrm{n(k)-n(i)+1}}{\mathrm{f_s}}$

  which no longer relied on timestamps and used the Fs instead.

- When analyzing the ECG of the 40 patients, X1012 returns **nan** in one of the windows' heart rate calculations. This also results in the error calculations returning **nan.**

  Proposed solution would be varying the window length, or perhaps applying overlaps between the windows. This however would need to be applied on both the ECG and the BCG to return the same lengths of windows.