PART-1: IDENTIFYING THE RELEVANT ECG FEATURES

-By S Ashwin Kumar

Extracting features using HeartPy module in Python

The dataset provided has 3500 entries of ECG readings that can be converted into relevant features using the HeartPy python module. The ECG reading can be converted into the heart rate data and then to following features:

- `BPM – heart rate (BPM), is calculated as the average beat-beat interval across the entire analysed signal (segment).`
- `IBI – interbeat interval.`
- `SDNN – standard deviation of RR intervals.`
- `SDSD – standard deviation of successive differences. See figure 8.`
- `RMSSD – root mean square of successive differences. See figure 8.`
- `PNN20 – proportion of successive differences above 20ms.`
- `PNN50 – proportion of successive differences above 50ms.`
- `HR_MAD – median absolute deviation of RR intervals.`
- `SD1 – standard deviation perpendicular to identity line (Poincaré parameters[7]).`
- `SD2 – standard deviation a long identtiy line.`
- `S – area of ellipse described by SD1 and SD2.`
- `SD1/SD2 – ratio.`
- `BREATHING RATE – that is the frequency with which the heart beats is strongly influenced by.`

The following functions of HeartPy module are used:

- **heartpy.get_data()**
  This returns a 1-dimensional `numpy.ndarray` containing the heart rate data.

  `get_data(filename, delim = ',', column_name = 'None')` requires one argument:

  filename: absolute or relative path to a valid (delimited .csv/.txt or matlab .mat) file;

- **heartpy.process()**
  This returns a dictionary containing the 13 features (as listed above) as keys, in the dictionary with respective values.

The working and usage of the functions can be observed below:

extract features from ECG using HeartPy

```
np.savetxt("GFG.csv",
           list(df.ECG[0].split(',')),
           delimiter =", ",
           fmt ='% s')
data = hp.get_data('GFG.csv')
wd, m = hp.process(data, sample_rate)
m
```

```
{'bpm': 78.77461706783369,
 'ibi': 761.6666666666667,
 'sdnn': 61.300477469687756,
 'sdsd': 83.71856417292621,
 'rmssd': 93.85792086611183,
 'pnn20': 0.4594594594594595,
 'pnn50': 0.10810810810810811,
 'hr_mad': 28.333333333333428,
 'sd1': 66.36509585293453,
 'sd2': 57.02489215080812,
 's': 11889.239379158029,
 'sd1/sd2': 1.1637916942907194,
 'breathingrate': 0.2419633598340823}
```

All the ECG readings of the dataset are converted to the above features and saved in a list using the above piece of code and iterating through each row of the data as follows:

```
features=[]           # list to store extracted features as 2D list
noisy_indices=[]      # List to store indices of readings that are noisy or are anomalies

for i in tqdm.tqdm_notebook(range(len(df))):

    # Save each reading as csv file
    np.savetxt("GFG.csv",
           list(df.ECG[i].split(',')),
           delimiter =", ",
           fmt ='% s')

    # Load the saved file using HeartPy
    data = hp.get_data('GFG.csv')

    #run analysis
    try:
        wd, m = hp.process(data, sample_rate)
        # Data stored in wd, m as dictionary


        #Append computed measures to features

        temp = list(m.values())
        temp.append(df['Classification'][i])
        features.append(temp)

    except Exception as e:
        print(e)
        noisy_indices.append(i)


```

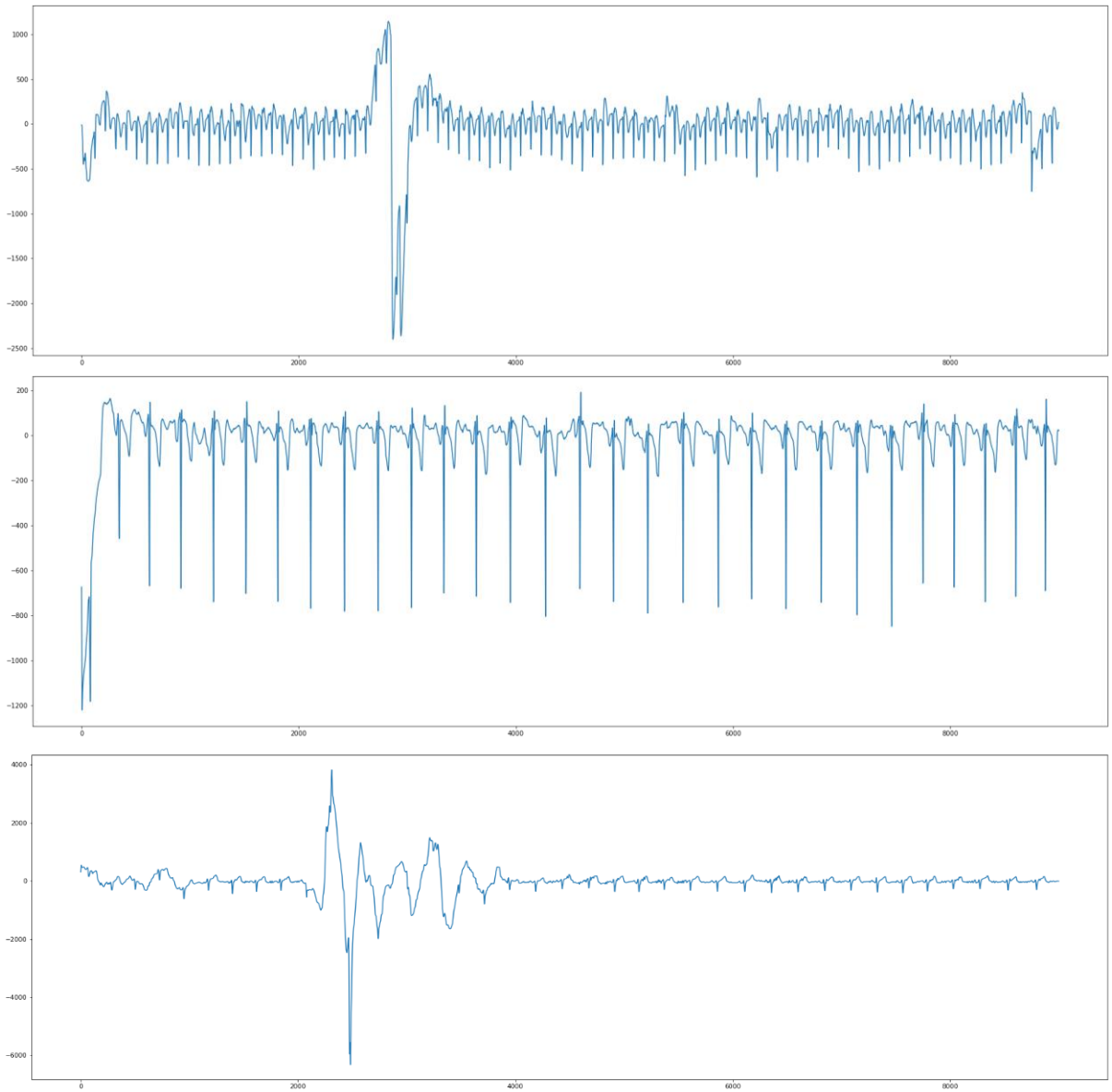
```
<ipython-input-6-31d5b204f285>:4: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  for i in tqdm.tqdm_notebook(range(len(df))):
```

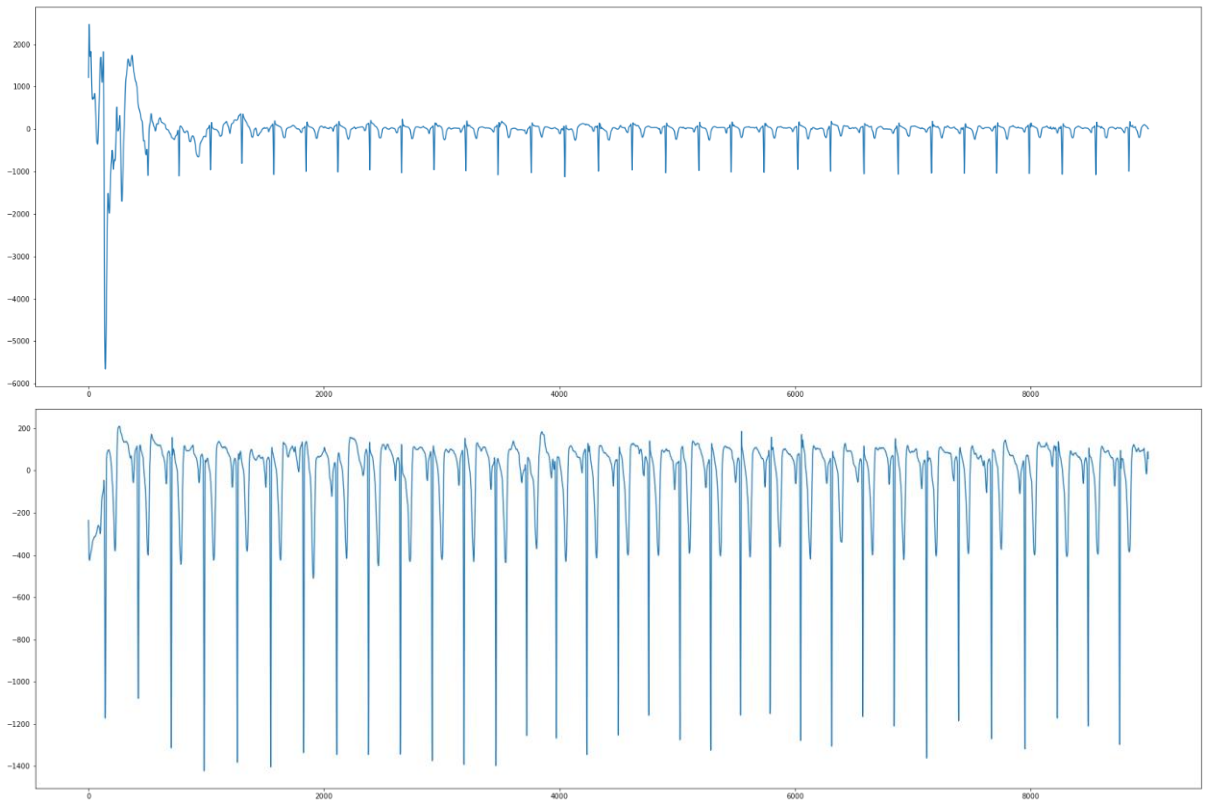
```
100% 3500/3500 [09:28<00:00, 6.16it/s]
```

The 19 ECG readings are processed i.e., filtered and scaled using the following functions of the HeartPy module:

- **heartpy.remove_baseline_wander()**
- **heartpy.scale_data()**

Following are a few graphs of noisy signals

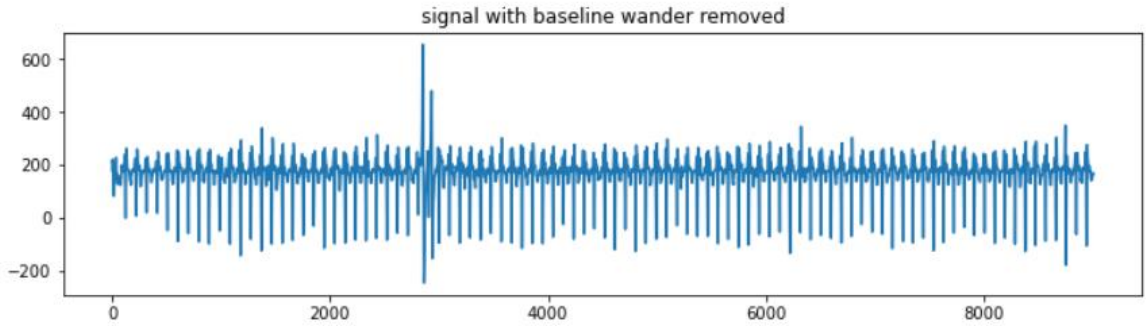


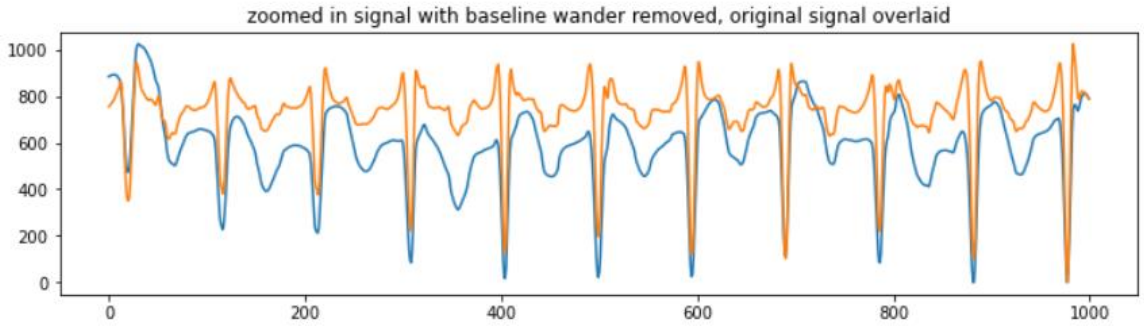


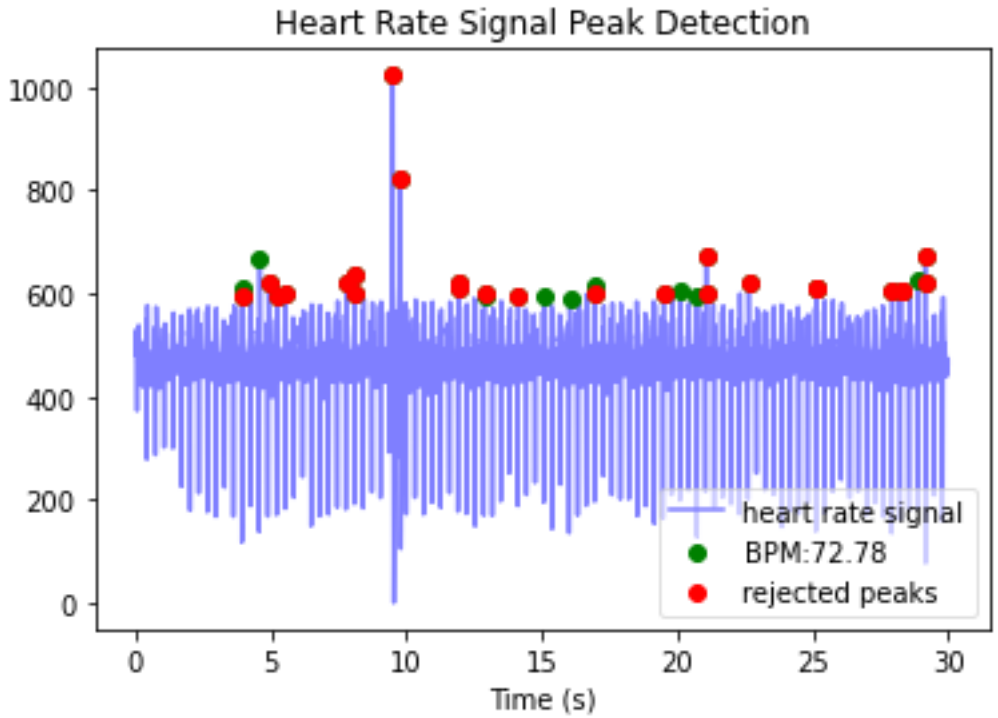
These types of signals can be filtered and scaled using the following piece of code:

```
def filter_and_visualise(data, sample_rate):
    '''
    function that filters using remove_baseline_wander
    and visualises result
    '''

    filtered = hp.remove_baseline_wander(data, sample_rate)

    plt.figure(figsize=(12,3))
    plt.title('signal with baseline wander removed')
    plt.plot(filtered)
    plt.show()

    #And let's plot both original and filtered signal, and zoom in to show peaks are not moved
    #We'll also scale both signals with hp.scale_data
    #This is so that they have the same amplitude so that the overlap is better visible
    plt.figure(figsize=(12,3))
    plt.title('zoomed in signal with baseline wander removed, original signal overlaid')
    plt.plot(hp.scale_data(data[200:1200]))
    plt.plot(hp.scale_data(filtered[200:1200]))
    plt.show()

    return filtered
```

After processing the below noisy signal we get a filtered signal as shown below:







Similarly, we perform filtering and scaling for all the data and store it in a list which will be joined to the first list that contains data processed from clean signals.

Merging the Data

Finally, both the datasets are now merged using list ' **extend**() ' method as shown below.

```
# Merging normal data and processed noisy data
# l : Processed features of clean signals
# noisy_l : Processed features of noisy signals
l.extend(noisy_l)
```

The dataset is now saved using the below code.

```
import csv

# field names
fields = ['bpm', 'ibi', 'sdnn', 'sdsd', 'rmssd', 'pnn20', 'pnn50', 'hr_mad', 'sd1', 'sd2', 's',
'sd1/sd2', 'breathingrate','Classification']

with open('pro_data.csv', 'w') as f:

    # using csv.writer method from CSV package
    write = csv.writer(f)

    write.writerow(fields)
    write.writerows(l)
```

The processed dataset looks like this:

bpm	ibi	sdnn	sdsd	rmssd	pnn20	pnn50	hr_mad	sd1	sd2	s	sd1/sd2	breathingrate	Classification
78.77462	761.6667	61.30048	83.71856	93.85792	0.459459	0.108108	28.33333	66.3651	57.02489	11889.24	1.163792	0.24196336	N
363.6364	165	114.6468	102.2823	155.5125	0.857143	0.714286	66.66667	106.4411	110.2126	36854.5	0.965779	0.303674461	N
64.42541	931.3095	169.7111	133.6252	219.9796	0.923077	0.807692	135	155.3302	183.698	89641.69	0.845574	0.191806046	A
151.8405	395.1515	179.2034	142.9934	360.8724	1	1	140	254.4423	130.2903	104148.1	1.952888	0.230255584	N
62.66682	957.4444	51.76645	20.78204	29.98084	0.413793	0.103448	45	20.97347	69.33283	4568.347	0.302504	0.139314572	N
99.68932	601.8699	111.5767	99.05609	150.8604	0.837838	0.621622	73.33333	106.5236	113.6025	38017.5	0.937688	0.162192847	A
191.993	312.5114	154.5365	130.303	225.4576	0.954545	0.909091	113.3333	159.4197	143.3677	71803.09	1.111964	0.263365815	N
67.25829	892.0833	35.38234	24.06441	26.81411	0.064516	0.064516	1.666667	18.95983	47.01286	2800.277	0.40329	0.140173816	N
114.3664	524.6296	207.9059	184.7653	249.7243	0.7	0.6	116.6667	173.7342	242.3193	132258.4	0.716964	0.317830279	N
108.2164	554.4444	270.4318	23.09401	540.4936	1	1	83.33333	375.2202	16.04201	18910.15	23.38985	0.066840452	N
63.83733	939.8889	19.67294	15.02596	29.06615	0.642857	0	18.33333	20.532	18.52769	1195.095	1.10818	0.319307458	N
199.9146	300.1282	170.4004	177.4678	317.552	0.933333	0.866667	68.33333	211.3277	122.6125	81403.13	1.723541	0.128287364	N
58.11138	1032.5	8.433917	4.478607	9.514987	0	0	6.666667	6.671238	9.0115	188.8658	0.740303	0.242222914	N
188.3985	318.4739	91.02466	84.29032	141.4452	0.949367	0.734177	20	99.88514	77.81009	24416.68	1.283704	0.227238297	A
130.9552	458.172	200.9509	170.0192	255.5854	0.7	0.6	173.3333	180.0655	214.3223	121240.5	0.840162	0.140914535	N
123.1672	487.1429	203.4483	49.95593	387.799	1	1	100	271.7883	41.50866	35442.09	6.547748	0.234755561	N
176.1348	340.6481	126.7283	67.76763	227.9965	1	0.969231	120	161.2129	78.7913	39905.04	2.046074	0.32647731	N