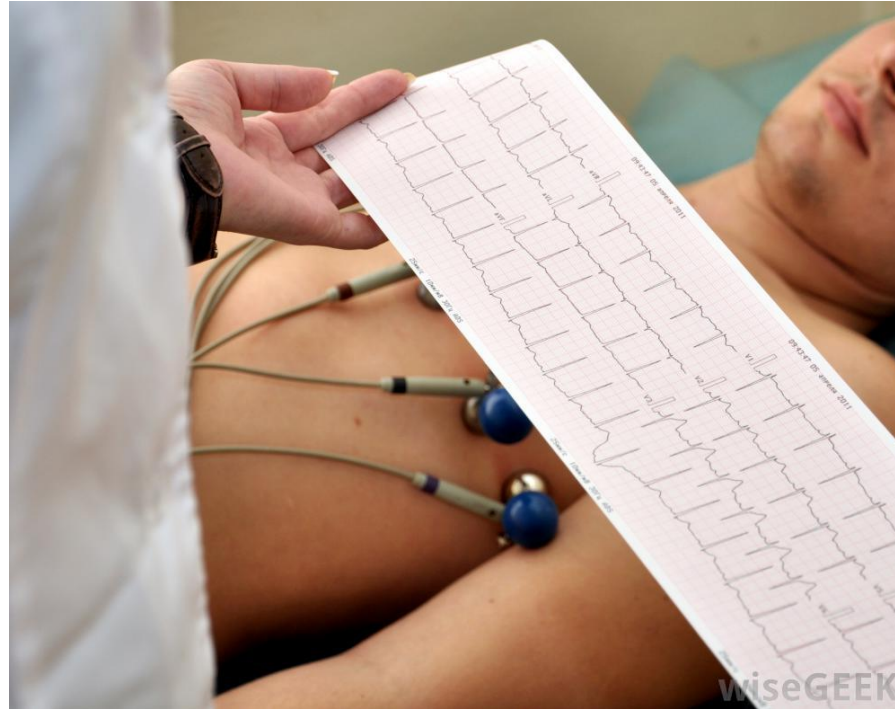
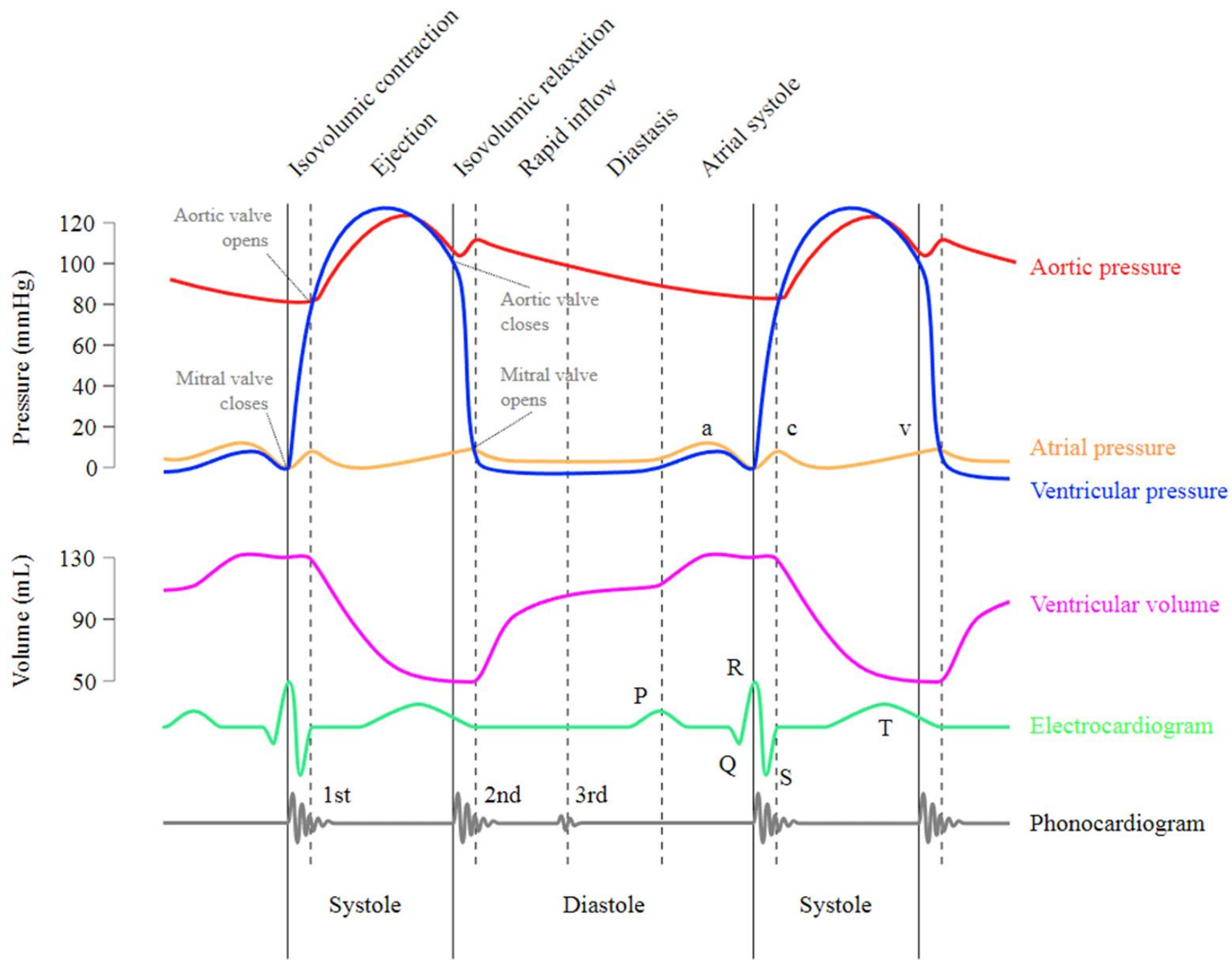


ECG Signal Classification

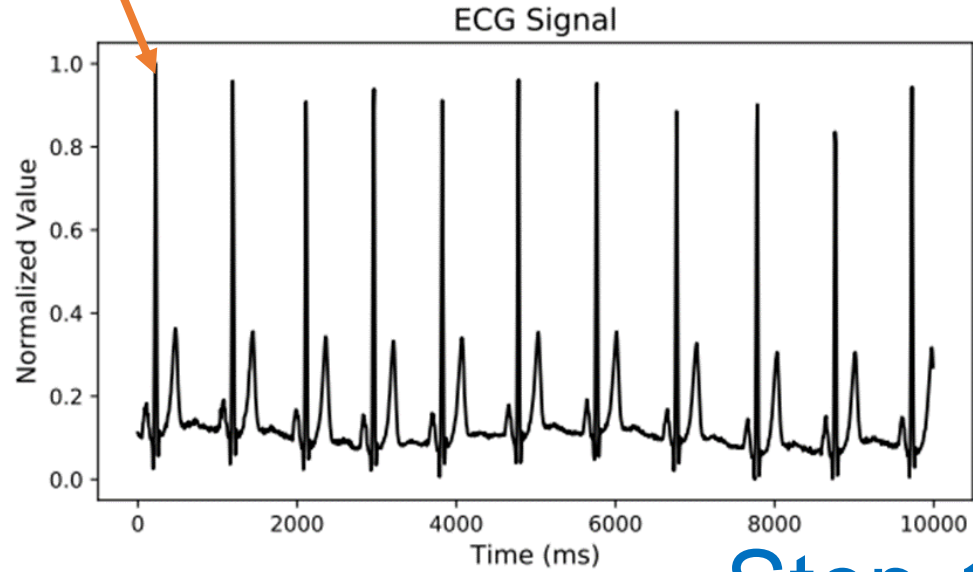
Electrocardiography (ECG) is the process of recording the electrical activity of the heart over a period of time using electrodes placed over the skin.



<https://tunedtolife.com/tips-for-medical-professionals-to-get-the-most-out-of-their-ecg-electrodes-and-machines/>



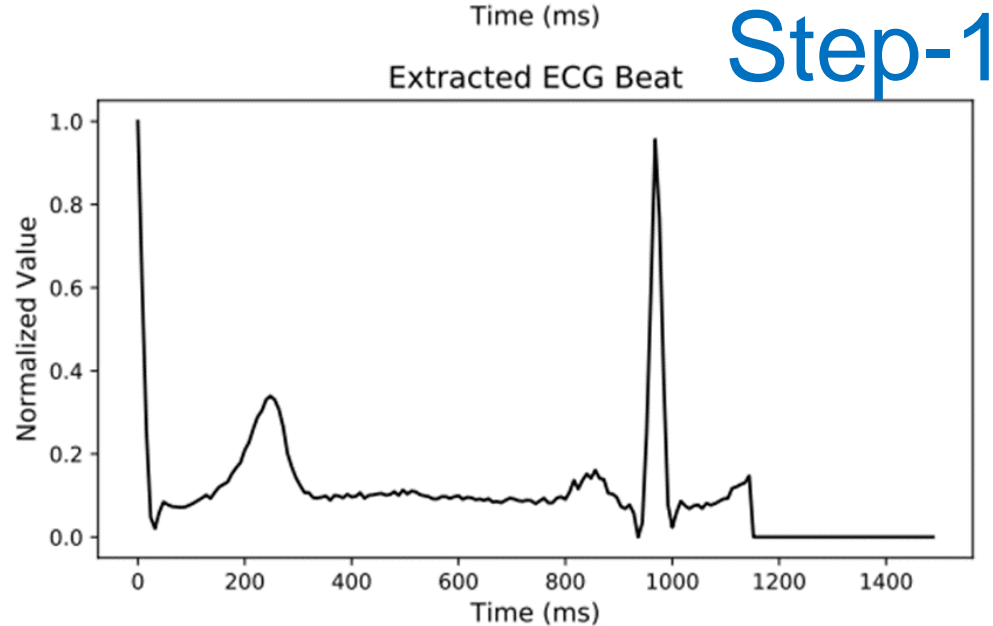
R-peak



This ECG project has two steps:

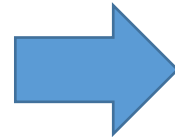
step-1: detect the R-peaks, so that we can extract individual ECG beats

step-2 : build a machine learning model to classify the ECG beats

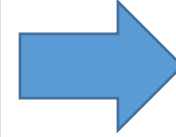


Step-1

Step-2

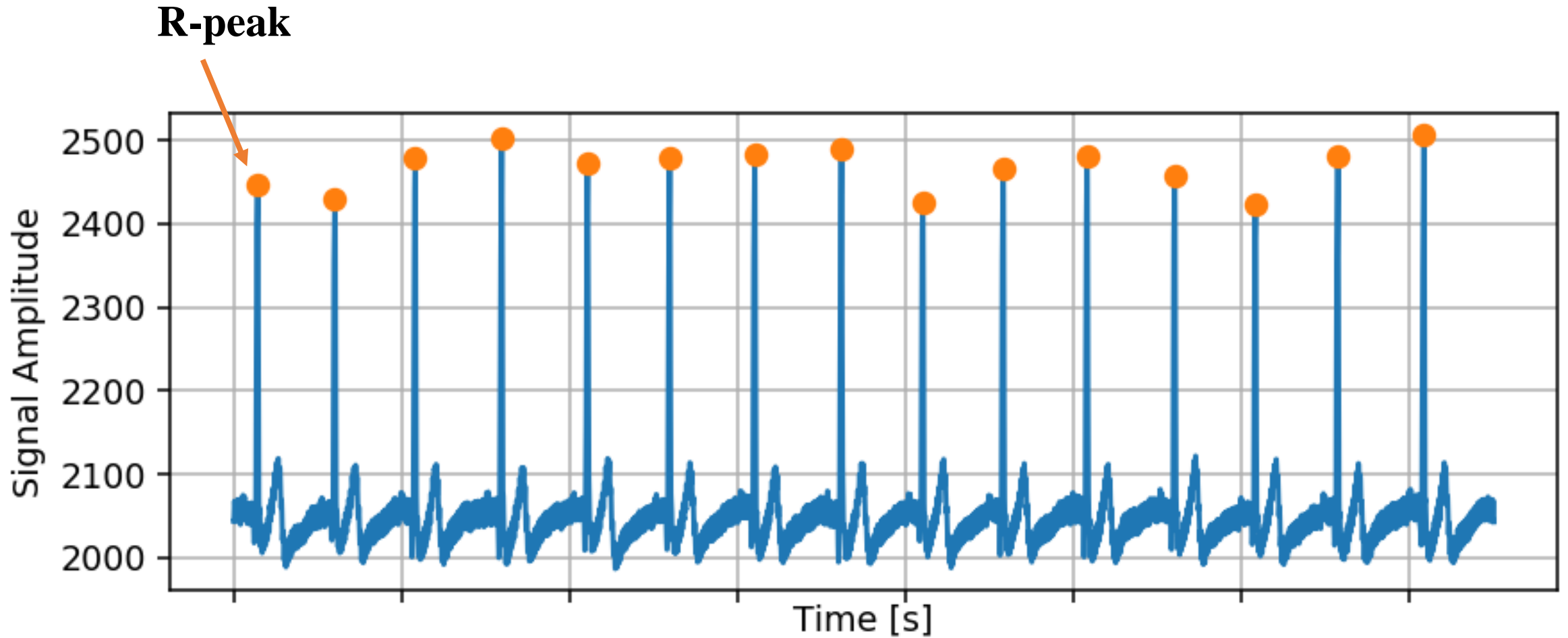


Machine
Learning
Model

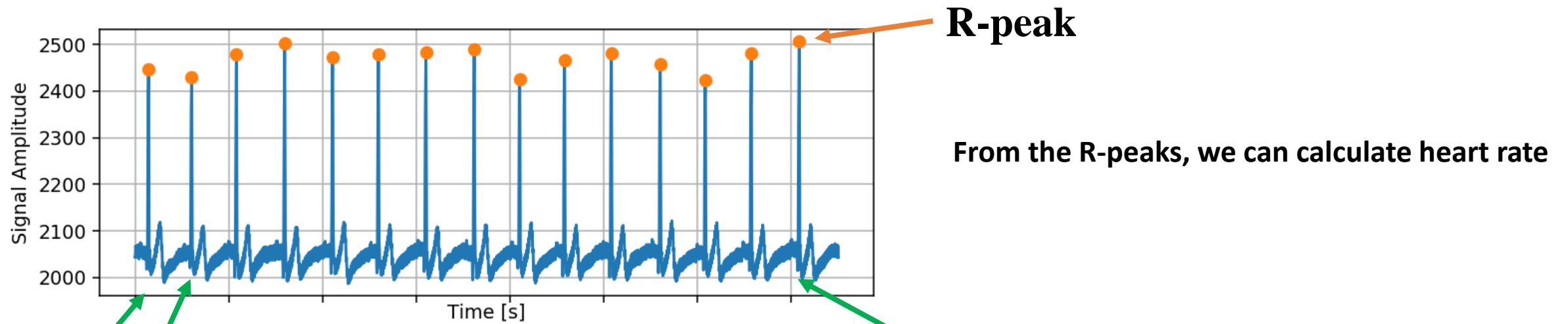


Normal heart?
Disease?

Step-1: detect the R-peaks



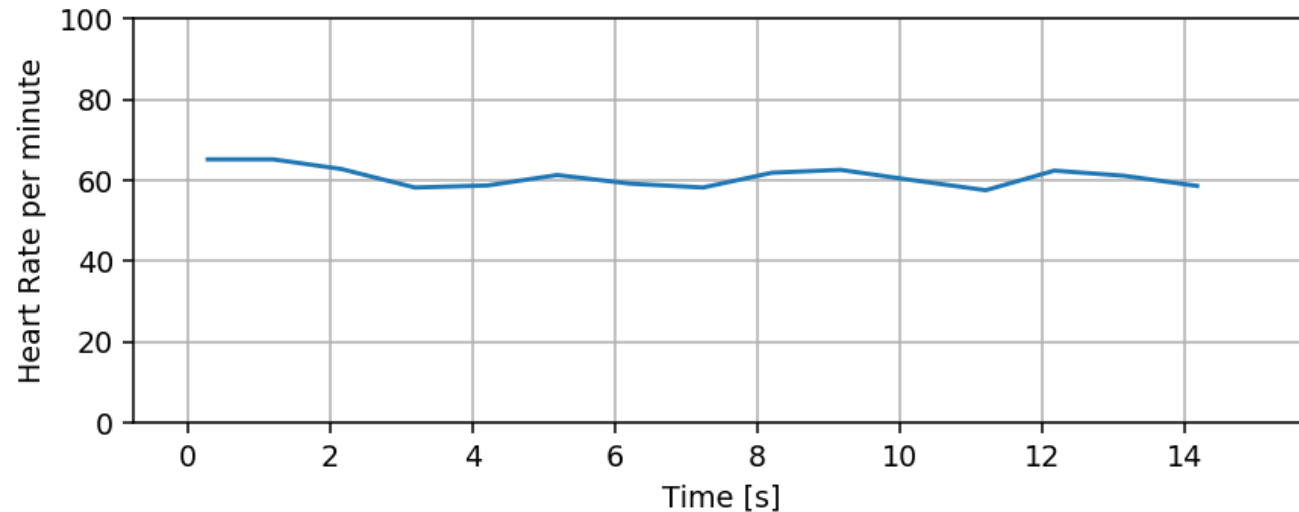
a R-peak indicates the left ventricle starts to contract



$[t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}]$ a list of time points

$$\text{Heart Rate Per minute } (t_n) = 60 / (t_n - t_{n-1})$$

use a numpy array $\{Rate[n], n=0,2,\dots\}$ to store *Heart Rate* (t_n) , set $Rate[0]=Rate[1]$



The Algorithm and An Example

- Read 1D_Signal_Processing_Peak_Detection.ipynb

Algorithm for 1D Signal Peak Detection

Input Signal \mathbf{x}

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

x_1 is a local peak if $x_1 - x_0 > 0$ and $x_1 - x_2 > 0$

two kernels

kernel-1

-1	1	0
----	---	---

kernel-2

0	1	-1
---	---	----

$$y_1 = x_1 - x_0$$

Processed Signal \mathbf{y}
using kernel-1

y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

$$z_1 = x_1 - x_2$$

Processed Signal \mathbf{z}
using kernel-2

z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

if $y_1 > 0$ and $z_1 > 0$, then x_1 is a peak

Write the Peak Detection Algorithm in Python

```
# x is a signal that has many peaks
h1 = [-1, 1, 0]    # kernel-1
h2 = [0, 1, -1]    # kernel-2
y = scipy.ndimage.correlate(x, h1, mode='nearest')
z = scipy.ndimage.correlate(x, h2, mode='nearest')
```

```
IndexArray1= np.where(y>0) find the indexes of the positive elements in y
IndexArray2= np.where(z>0) find the indexes of the positive elements in z
```

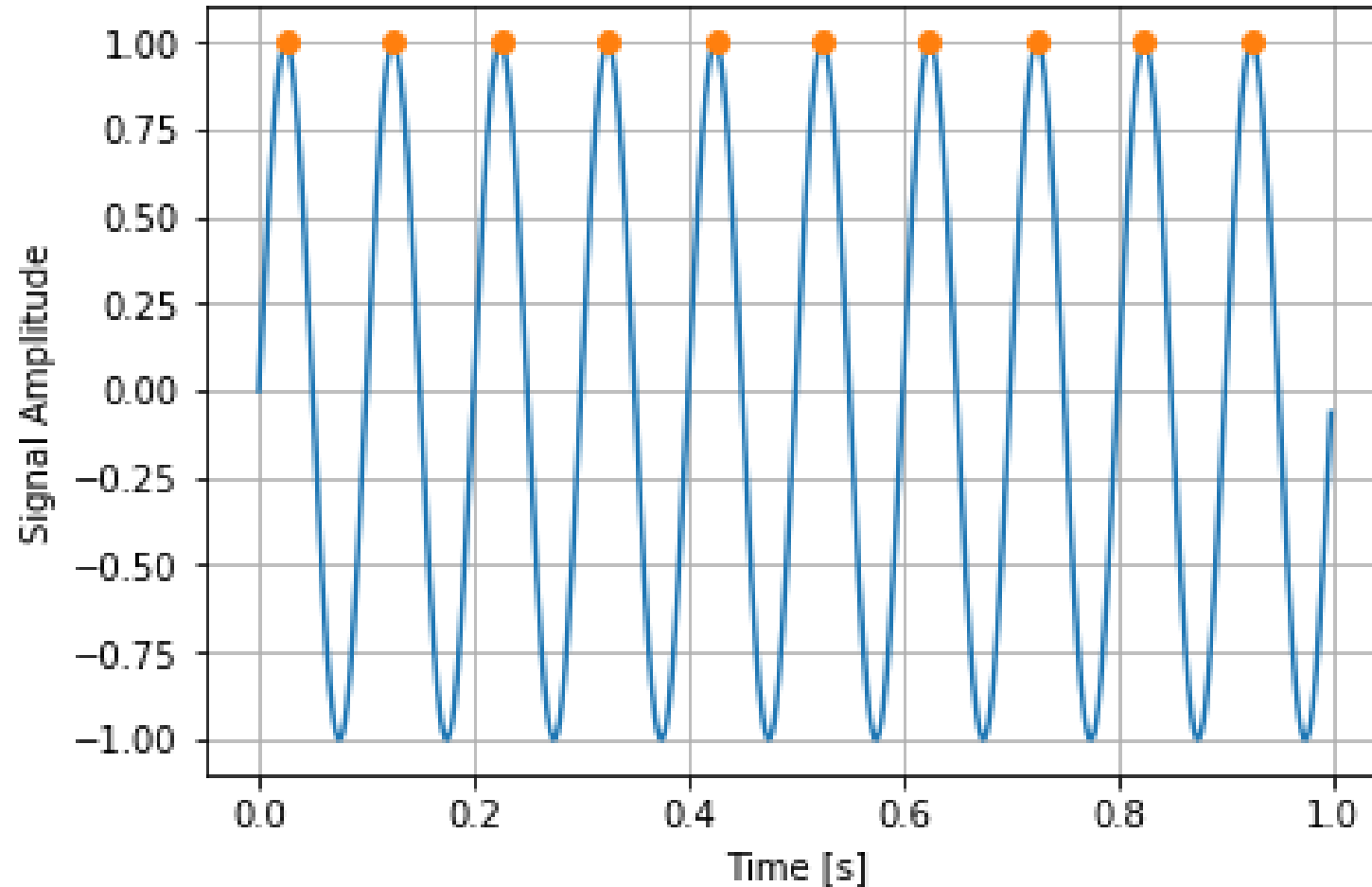
find the intersection of IndexArray1 and IndexArray2 (where $y > 0$ and $z > 0$)

```
PeakIndexArray = np.intersect1d(IndexArray1, IndexArray2)
```

$m = \text{PeakIndexArray}[n]$, and m is an element-index of the signal/array x

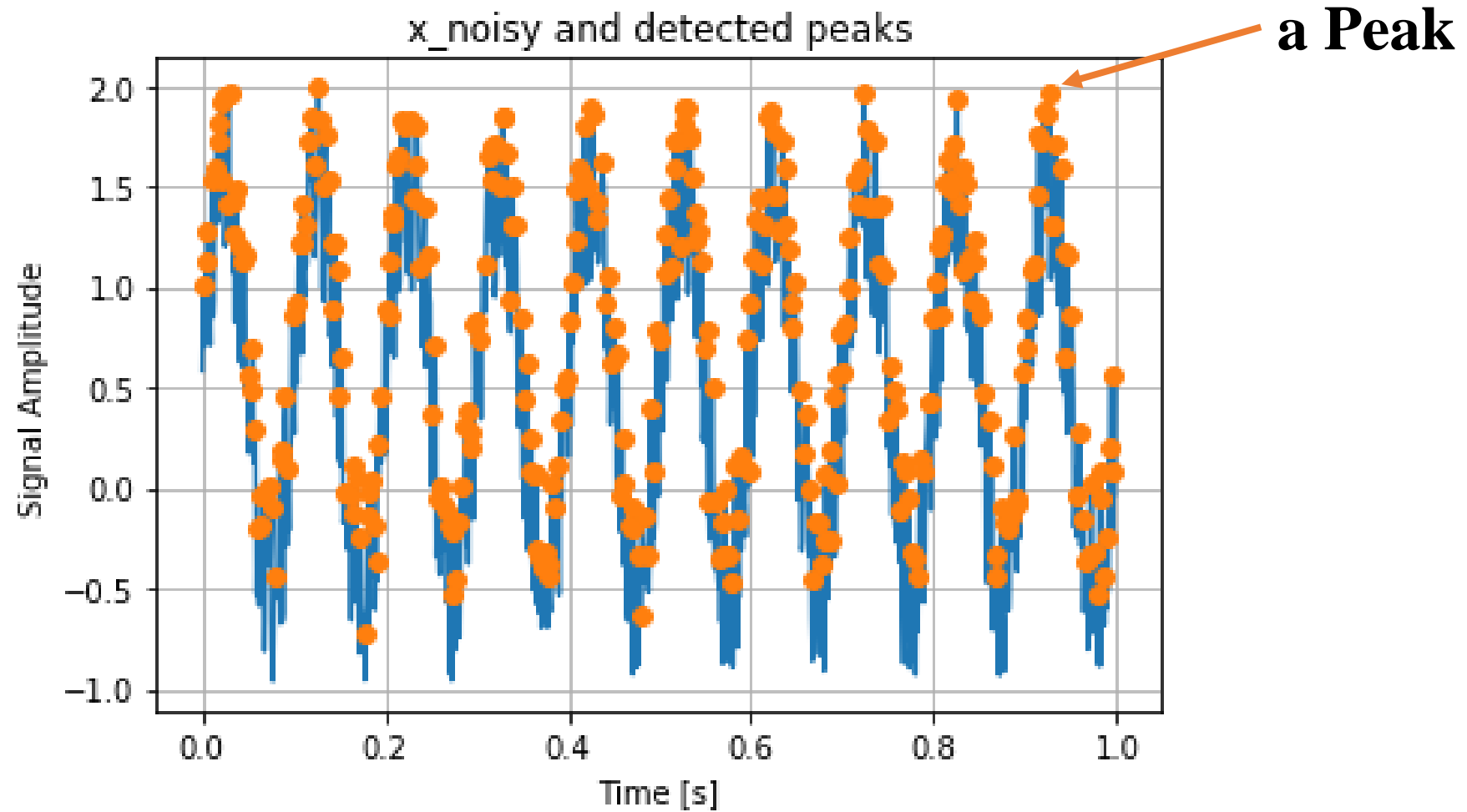
$x[m]$ is a peak

Apply the Algorithm to detect peaks of a clean signal



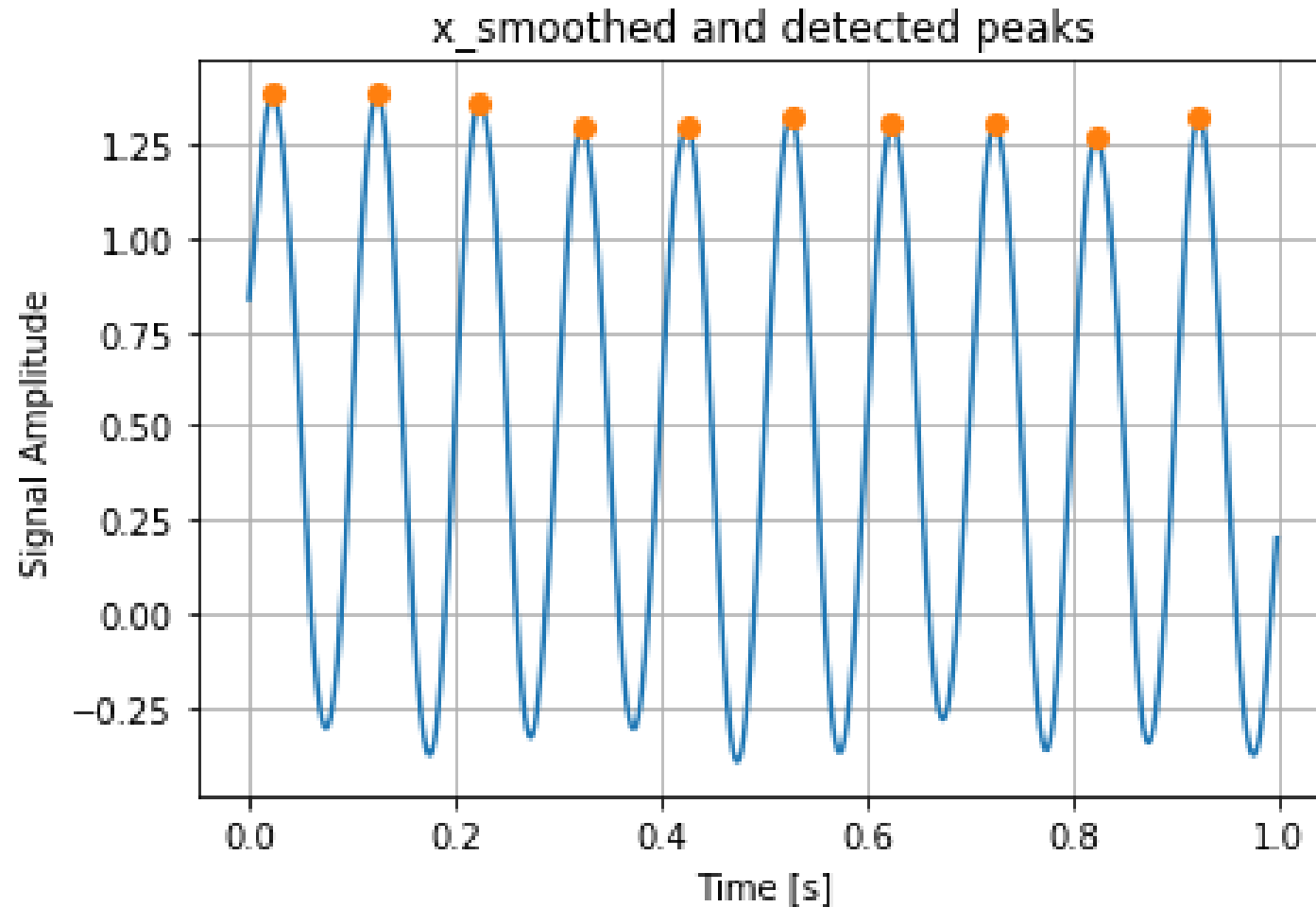
Read 1D_Signal_Processing_Peak_Detection.ipynb

Apply the Algorithm to detect peaks of a noisy signal



The algorithm does not work on noisy signals

Smooth the noisy signal and then detect the peaks



Your Task

- complete your task in `project_ecg_step_1_template.ipynb`