# MACHINE LEARNING FOR EVENT DETECTION IN PHYSIOLOGICAL SIGNALS

**COURSE:** ACM40730: MATHEMATICA FOR RESEARCH

**SUBMITTED BY:** ARLENE JOHN

**SUBMISSION DATE:** 02/12/2018

## INTRODUCTION

### AIM

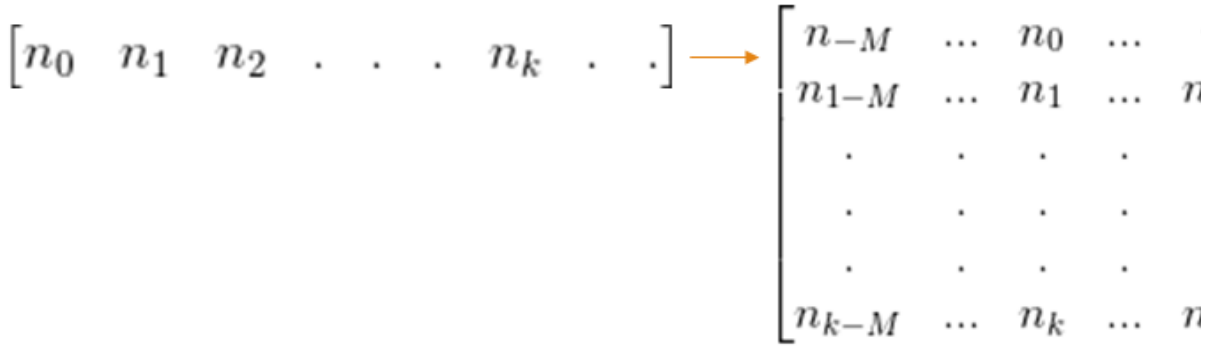
The project aims to check the feasibility of transforming event detection problems to classification problems that can be solved using supervised machine learning algorithms. The detection of events in physiological signals is traditionally carried out using signal processing methods such as filtering, differentiation, peak detection, etc. Event detection is of high importance in scenarios where the exact time instant at which an event occurs needs to be recorded for further inference. For example, in the case of heart rate detection using Electrocardiogram (ECG are electrical signals recording cardiac activity and contains 5 waves namely P, Q, R, S, and T) the exact time instant of occurrence of two consecutive R peaks needs to be detected for calculating heart rate.

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a list of predictors) and the desired output value (called the supervisory signal, classes, or labels).

Usually, signals on which event detection methods are carried out are 1 Dimensional such as physiological signals Electrocardiogram, Electromyogram, Speech, etc. Traditionally these signals are windowed and certain features are extracted from these signals to act as predictors for supervised machine learning algorithms. But, it is a common observation that extracting features from a window of signals would reduce time resolution of the predicted output leading to information loss (exact time of event occurrence is not available), difficulty to decide the most suitable features and loss in the spatial-temporal structure of the signal. In this project, I intend to propose a novel method of choosing features based on sample points from the neighboring samples. Data preparation is discussed in the next section.

## DATA PREPARATION

Data is prepared by using data frames centered around the sample point of interest $n_k$ and stacking them. When $n_0$ is the sample point of interest, zeros are prepended to it. The Figure below explains the data preparation process of which the frame size is 2M+1:



All sample points within a frame act as the vectors/predictors describing the sample point of interest in the 2 *M*+1 dimensional space.

The function for stacking data and labels are called *stackdata* in this notebook and the code is provided below (Please execute now for smooth execution later on).

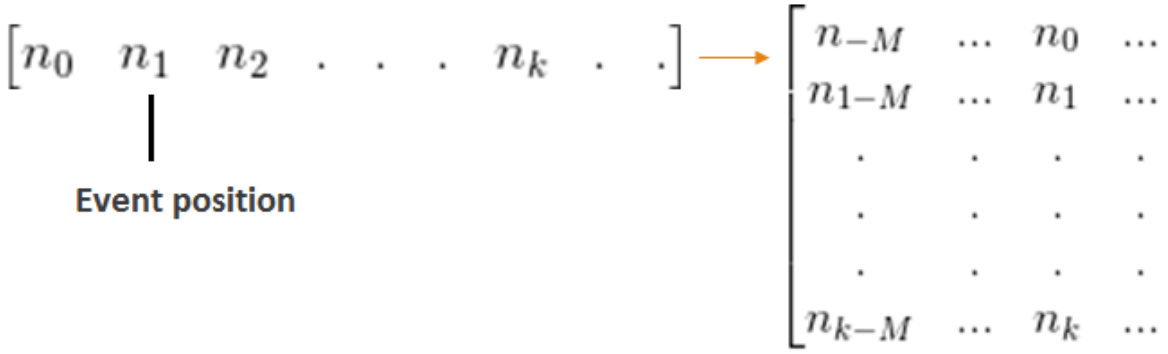
In[1]:=
```
stackdata[signal_, stacklength_] := Module[{stackedsigdata, i, signalappended, setval},
  (*signal is the signal and stacklength is length of data frame desired*)
  stackedsigdata = ConstantArray[0, {Dimensions[signal][[1]] + 1, stacklength}];
  (*creating a constant array*)
  signalappended = ArrayPad[signal, Floor[stacklength / 2]];
  (*prepending and appending zeros to the datastack so that the first frame
   will have 0s prepended to the first element in middle position and last
   frame will have 0s appended to the last element in middle position*)
  For[i = Floor[stacklength / 2] + 1;
   setval = 0, i ≤ Length[signalappended] - Floor[stacklength / 2], i++,
   stackedsigdata[[i - Floor[stacklength / 2] + 1]] =
    signalappended[[i - Floor[stacklength / 2] ;; i + Floor[stacklength / 2]]]];
  (*maintains first row as frame of 0s, starts assigning values from second row*)
  Return[stackedsigdata];];
```

Once the data is stacked, frames with irrelevant information (more zeros than actual values) need to be deleted. For this, a function called *deleting* is defined (Please execute now for smooth execution later on).

In[2]:=
```
deleting[Datalongstack1_, label1_] :=
  Module[{j = 2, i, Datalongstack = Datalongstack1, label = label1, Data, count},
   (*Datalongstack1 is the data frame stack obtained from stackdata and
    labels are the class labels which will be discussed in next section*)
   Data = ConstantArray[0, {Dimensions[Datalongstack][[1]],
      Dimensions[Datalongstack][[2]] + 1}];
   For[ i = 2, i <= Length[Datalongstack], i++,
    count = Count[Datalongstack[[i]][[All]], x_ /; x == 0];
    If[count < 36, Data[[j]][[1 ;; -2]] = Datalongstack[[i]][[All]];
     Data[[j]][[-1]] = label[[i]];
     j++, Null];];
   Data = Drop[Data, {j, -1}];
   Return[Data];];
```

## CLASSES/LABELS PREPARATION

The list of classes are made by assigning a value of 1 to the frame position $k$ if event occurs at point $n_k$ at which that corresponding frame is centered. This is discussed in the figure below. Raw data labels have a label value of 1 at the time instant of occurrence of event.



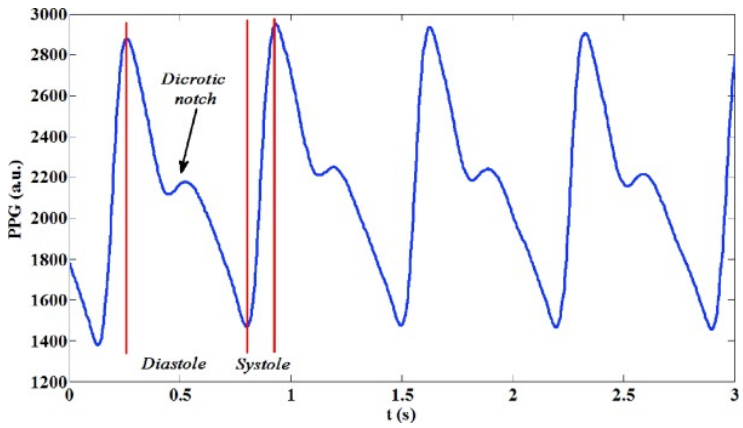

## PROBLEM SELECTED

The problem chosen for checking the feasibility of the proposed method of solving event detection problems by transforming them into classification problems is that of calculating heart beat rate (a.k.a heartrate) from cardiac activity signals such as ECG and PPG signals. The simplest method for detecting heartrate is to detect a repetitive event that occurs per heartbeat. For this problem, I have taken data from the MIMIC III Clinical database (https://physionet.org/physiobank/database/mimic3cdb/) made available by Physiobank. As all data cannot be shared, relevant data has been extracted and converted to .mat files for demonstration in this project.

# HEART RATE DETECTION FROM PHOTOPLETHYSMOGRAM (PPG) SIGNALS

**The steps that follow involve importing data from your PC. The required data is provided in the data.zip folder file, Make changes in the file location accordingly for smooth execution.**

Photoplethysmogram (PPG) is an optically obtained plethysmogram, a volumetric measurement of an organ. A PPG is often obtained by using a pulse oximeter which illuminates the skin and measures changes in light absorption. As heart pumps blood to the periphery per cardiac cycle, the arteries and arterioles in the subcutaneous tissue distend. If the pulse oximeter is attached without compressing the skin, a pressure pulse can be seen as shown in figure below. From the occurrence of various peaks in the PPG signal, we can exactly match the time of systole and diastole.




PPG signal data is imported

In[3]:=
```
Datalong = Import["C:\\Users\\Arlene John\\Desktop\\data\\ppg_continuous.mat"];
(*load ppg_continuous.mat file*)
```

Data imported is of a 3 dimensionsal form and needs to be converted to 2 dimensions.

In[4]:=
```
Datalong = Datalong[[1]][[All]][[All]];
```

In[5]:=
```
Datalong = Table[Datalong[[i]][[1]], {i, 1, Length[Datalong]}];
```

Calling the stacking function

In[6]:=
```
Datalong = stackdata[Datalong, 71];
```

Importing the class labels and converting it to a vector form

In[7]:=
```
labels = Import["C:\\Users\\Arlene John\\Desktop\\data\\label_continuous2.mat"];
```

In[8]:=
```
labels = labels[[1]][[All]][[All]];
```

In[9]:=
```
labels = Table[labels[[i]][[1]], {i, 1, Length[labels]}];
```

As the *stackdata* function had the first row just full of zeros, prepending a class label of 0.

In[10]:=
```
labels = Prepend[labels, 0.];
```

Frames with irrelevant information (more zeros than actual values, except the first row ) need to be deleted. Calling the *deleting* function

In[11]:=
```
T = deleting[Datalong, labels];
```

Extracting data and labels from the variable T.

In[12]:=
```
Datalongstack = T[[All, 1 ;; -2]];
```

In[13]:=
```
label = T[[All, -1]];
```

In[14]:=
```
label[[1]] = 0.;
```

Separating training data, training labels, testing data and testing labels. 70%+20 samples are used for training while the rest is used for testing.

In[15]:=
```
trainingdata1 = Datalongstack[[1 ;; Ceiling[0.7 * Length[Datalongstack]] + 20]][[All]];
```

In[16]:=
```
testdata1 = Datalongstack[[Ceiling[0.7 * Length[Datalongstack]] + 21 ;; ]][[All]];
```

In[17]:=
```
traininglabel1 = label[[1 ;; Ceiling[0.7 * Length[Datalongstack]] + 20]][[All]];
```

In[18]:=
```
testlabel1 = label[[Ceiling[0.7 * Length[Datalongstack]] + 21 ;; ]][[All]];
```

For classification , the machine learning algorithm chosen was Support Vector Machine as this is a binary classification problem and SVM is fast with regard to fit time, very fast with regard to prediction time with moderate memory overhead. The KernelType chosen was Polynomial with Polynomial Degree of 8 over multiple rounds of trial and error.

In[19]:=
```
classifier1 = Classify[trainingdata1 → trainingdata1, Method →
    {"SupportVectorMachine", "KernelType" → "Polynomial", "PolynomialDegree" → 18}];
(*generating a classifer model*)
```

In[20]:=
```
results1 = classifier1[testdata1];
(*obtaining classification results using the generated model on the testdata*)
```

When checking for the performance of event detection problems, the point of event occurrence is allowed some level of tolerance. To correct for this tolerance, the function *beatbybeat* is defined with a tolerance band of 0.15 seconds. The sampling rate for the PPG signals were 125 Hz. The function returns a matrix with the test classes and obtained classes which have been tweaked to benefit from the tolerance.

In[21]:=
```
beatbybeat[ref1_, compare1_, freq_] :=
  Module[{Fs = freq, ref = ref1, compare = compare1, signal1, signal2, window, windowpos,
    S, FPsetpos, k, pos, joined, joinedmatrix}, (*ref1 is the test class,
   compare1 is result class and freq is the signal sampling frequency*)
   If[Length[ref] > 3 && Length[compare] > 3,
                                        If[Length[ref] >= Length[compare],
     compare = N[ArrayPad[compare, {0, Length[ref] - Length[compare]}]],

     ref = N[ArrayPad[ref, {0, Length[compare] - Length[ref]}]]];
    signal1 = ref;
    signal2 = compare;
    For[i = 1, i ≤ Length[signal1], i++,
     If[signal1[[i]] == 0, Continue[], window = signal2[[
        Max[(i - Round[0.15 * Fs]), 1] ;; Min[Length[signal2], (i + Round[0.15 * Fs])]]];
                                       windowpos = Range[Max[(i - Round[0.15 * Fs]), 1],
        Min[Length[signal2], (i + Round[0.15 * Fs])], 1];
                                       S = Total[window];
                                       If[S == 1, signal2[[Max[(i - Round[0.15 * Fs]), 1]
          ;; Min[Length[signal2], (i + Round[0.15 * Fs])]]] = 0;
       signal2[[i]] = 1,];
      If[S > 1, FPsetpos = {0}; For[k = 1, k ≤ Length[window], k++,
        If[window[[k]] == 1, FPsetpos = Join[FPsetpos, {windowpos[[k]]},]];
       FPsetpos = FPsetpos[[2 ;;]];
       pos = Ordering[FPsetpos - k, 1];
       signal2[[FPsetpos[[pos]]]] = 0;
       signal2[[i]] = 1,],];
    joined = Join[signal1, signal2];
    joinedmatrix = Multicolumn[joined, 2],
    If[Length[ref] >= Length[compare],
     compare = N[ArrayPad[compare, {0, Length[ref] - Length[compare]}]],

     ref = N[ArrayPad[ref, {0, Length[compare] - Length[ref]}]]];
    signal1 = ref;
    signal2 = compare;
    joined = Join[signal1, signal2];
    joinedmatrix = Multicolumn[joined, 2],];
   Return[joinedmatrix];];
```

In[22]:=
```
joinmat = beatbybeat[testlabel1, results1, 125];
```

Once the corrected result classes are obtained, the confusion matrix needs to be calculated. For this a function *confusionmatrix* was defined which returns the confusion matrix as {{True Negatives, False Positives},{False Negatives, True Positives}}

In[23]:=
```
confusionmatrix[joinedmatrix_, length_] :=
 Module[{t, patt1, patt2, patt3, patt4, Cval, TP, FP, TN, FN},
  patt1 = {0., 0.}; patt2 = {0., 1.};
  patt3 = {1., 0.};
  patt4 = {1., 1.};
  TP = 0;
  TN = 0;
  FP = 0;
  FN = 0;
  For[t = 1, t ≤ length, t++,
   If[joinedmatrix[[1]][[t]] == patt1, TN++,];
   If[joinedmatrix[[1]][[t]] == patt2, FP++,];
   If[joinedmatrix[[1]][[t]] == patt3, FN++,];
   If[joinedmatrix[[1]][[t]] == patt4, TP++,];];
  Cval = {{TN, FP}, {FN, TP}};
  Return[Cval]]
```

In[24]:=
```
Ck1 = confusionmatrix[joinmat, Max[Length[testlabel1], Length[results1]]]
```

Out[24]=
```
{{10614, 108}, {10, 143}}
```

From the confusion matrix, we observer that the classifier exhibited a Sensitivity of 94.1% and specificity of 99.5%. Sensitivity is not very good and in case of physiological signals, specificity needs to be 100% as a larger number of false positives can leads to Alarm Fatigue. Also the PolynomialDegree chosen was 18, which is high leading to more time consuming computations and therefore is not optimal. Therefore the use of ECG signals with a much more vibrant spatio-temporal structure is discussed in the next section. The classifier performance is checked and animated to simulate real-time peak detection conditions next

In[25]:=
```
signal1 = Import["C:\\Users\\Arlene John\\Desktop\\data\\PPG_signal.mat"];
(*importing ppg signal and converting to a signal vector with zero padding*)
```

In[26]:=
```
signal1 = signal1[[1]][[All]][[All]];
```

In[27]:=
```
signal1 = Transpose[signal1];
```

In[28]:=
```
signalpadded1 = ArrayPad[signal1, {{0, 0}, {70, 0}}];

(*Preparing data for performance check and heart rate calculation*)
```

In[29]:=
```
signalwindow1 = ConstantArray[0, {Length[Transpose[signalpadded1]], 71}];
(*pre-assigning dimensions to signal window such that data frame length is 71*)
```

In[30]:=
```
RR1 = ConstantArray[0, {1, Length[Transpose[signalpadded1]]}];
(*For storing heartrate data*)
```

In[31]:=
```
j1 = ConstantArray[0, {1, Length[Transpose[signalpadded1]]}];(*loop variable*)
```

In[32]:=
```
plotposition1 = ConstantArray[0, {1, Length[Transpose[signalpadded1]]}];
(*plotting position variable for animating*)
```

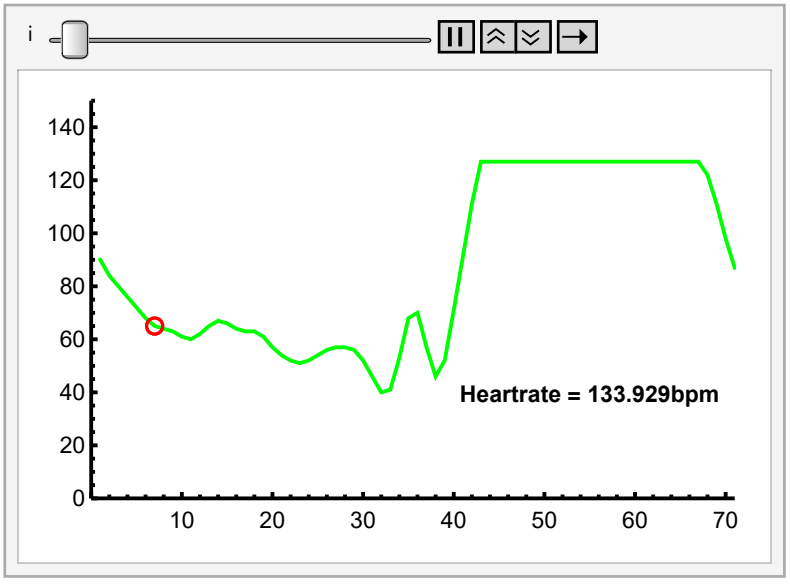
Warning-The code that follows  has a lengthy evaluation time

In[33]:=
```
For[i = 1;
  setval = 0, i ≤ Length[Transpose[signalpadded1]] - 71,
  i++, signalwindow1[[i]][[All]] = signalpadded1[[1]][[i ;; i + 70]];
  If[classifier1[signalwindow1[[i]][[All]]] == 1.,
   RR1[[1]][[i]] = 60 / ((i - setval) / 125);
   setval = i;
   plotposition1[[1]][[i]] = 36;
   j1[[1]][[i]] = 1,
   j1[[1]][[i]] = 0;
   If[plotposition1[[1]][[i - 1]] ≠ 0,
    plotposition1[[1]][[i]] = plotposition1[[1]][[i - 1]] - 1,];
   If[i ≠ 1, RR1[[1]][[i]] = RR1[[1]][[i - 1]],]];
```

The results obtained from the previous step are plotted and animated. The green signal is the PPG signal while the red circle indicates the position and time instant of event occurrence.

In[34]:=
```
Animate[If[
  plotposition1[[1]][[i]] ≠ 0, Show[ListPlot[signalwindow1[[i]][[All]], Joined → True,
    PlotRange → {{0, 71}, {0, 150}}, PlotStyle → {Directive[Green, Thick]},
    AxesStyle → Directive[Black, 12, Thick]], ListPlot[
    {{plotposition1[[1]][[i]], signalwindow1[[i]][[plotposition1[[1]][[i]]]]}},
    PlotMarkers → Graphics[{Red, Circle[]}, ImageSize → 10]],
   Graphics[Text["Heartrate = " <> ToString@N[RR1[[1]][[i]]] <> "bpm",
     {55, 40}, BaseStyle → {FontWeight → "Bold", FontSize → 12}]]], Show[
   ListPlot[signalwindow1[[i]][[All]], Joined → True, PlotRange → {{0, 71}, {0, 150}},
    PlotStyle → {Directive[Green, Thick]}, AxesStyle → Directive[Black, 12, Thick]],
   Graphics[Text["Heartrate = " <> ToString@N[RR1[[1]][[i]]] <> "bpm",
     {55, 40}, BaseStyle → {FontWeight → "Bold", FontSize → 12}]]]],
 {i, 1, Length[signalwindow1] - 71, 1, AnimationRate → 35, Appearance → "Open"}]
```

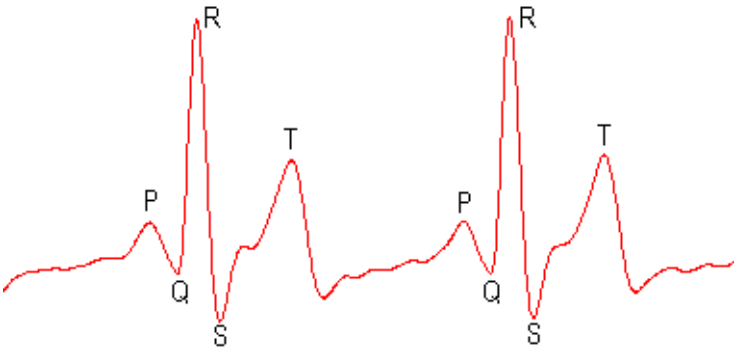
Out[34]=




# HEART RATE DETECTION FROM ELECTROCARDIOGRAM

# (ECG) SIGNALS

Electrocardiography (ECG) is the process of recording the electrical activity of the heart over a period of time using electrodes placed over the skin. These electrodes detect the tiny electrical changes on the skin that arise from the heart muscle's electrophysiologic pattern of depolarizing and repolarizing during each heartbeat. There are three main components to an ECG: the P wave, which represents the depolarization of the atria; the QRS complex, which represents the depolarization of the ventricles; and the T wave, which represents the repolarization of the ventricles. Detecting any one of these waves is sufficient to calculate heart rate but most commonly the R peak is preferred for detection due to its high energy content.




Import ECG Data

In[37]:=
```
Datalong2 = Import["C:\\Users\\Arlene John\\Desktop\\data\\ecg_continuous.mat"];
```

In[38]:=
```
Datalong2 = Datalong2[[1]][[All]][[All]];
```

In[39]:=
```
Datalong2 = Table[Datalong2[[i]][[1]], {i, 1, Length[Datalong2]}];
```

In[40]:=
```
Datalong2 = stackdata[Datalong2, 71];
```

The labels are the same as that imported for PPG signals as the data was obtained from a single patient simultaneously.

In[41]:=
```
labels2 = labels;
```

Preparing data as discussed in previous section

In[42]:=
```
T = deleting[Datalong2, labels2];
```

In[43]:=
```
Datalongstack1 = T[[All, 1 ;; -2]];
```

In[44]:=
```
label1 = T[[All, -1]];
```

In[45]:=
```
label1[[1]] = 0.;
```

In[46]:=
```
trainingdata2 = Datalongstack1[[1 ;; Ceiling[0.7 * Length[Datalongstack]] + 20]][[All]];
```

In[47]:=
```
testdata2 = Datalongstack1[[Ceiling[0.7 * Length[Datalongstack]] + 21 ;; ]][[All]];
```

In[48]:=
```
traininglabel2 = label1[[1 ;; Ceiling[0.7 * Length[Datalongstack]] + 20]][[All]];
```

In[49]:=
```
testlabel2 = label1[[Ceiling[0.7 * Length[Datalongstack]] + 21 ;; ]][[All]];
```

Classifier chosen as discussed in previous section with PolynomialDegree 3.

In[50]:=
```
classifier2 = Classify[trainingdata2 → traininglabel2, Method →
    {"SupportVectorMachine", "KernelType" → "Polynomial", "PolynomialDegree" → 3}];
```

In[51]:=
```
results2 = classifier2[testdata2];
```

In[52]:=
```
joinmat = beatbybeat[testlabel2, results2, 125];
```

In[53]:=
```
Ck2 = confusionmatrix[joinmat, Max[Length[testlabel2], Length[results2]]]
```

Out[53]=
```
{{10721, 1}, {8, 145}}
```

From the confusion matrix, we observer that the classifier exhibited a Sensitivity of 88.8% and specificity of 100%. Sensitivity is lower than that discussed in the section with PPG signals while specificity has reached 100%, indicating that there is scope for improvement. The classifier performance is checked and animated to simulate real-time peak detection conditions next

In[55]:=
```
signal2 = Import["C:\\Users\\Arlene John\\Desktop\\data\\ECG_signal.mat"];
(*importing ecg signal and converting to a signal vector with zero padding*)
```

In[56]:=
```
signal2 = signal2[[1]][[All]][[All]];
```

In[57]:=
```
signal2 = Transpose[signal2];
```

In[58]:=
```
signalpadded2 = ArrayPad[signal2, {{0, 0}, {70, 0}}];
```

```
(*Preparing data for performance check and heart rate calculation*)
```

In[59]:=
```
signalwindow2 = ConstantArray[0, {Length[Transpose[signalpadded2]], 71}];
```

```
(*pre-assigning dimensions to signal window such that data frame length is 71*)
```

In[60]:=
```
RR2 = ConstantArray[0, {1, Length[Transpose[signalpadded2]]}];
(*to store hearrtrate information*)
```

In[61]:=
```
j2 = ConstantArray[0, {1, Length[Transpose[signalpadded2]]}];
```

In[62]:=
```
plotposition2 = ConstantArray[0, {1, Length[Transpose[signalpadded2]]}];
```

Warning-The code that follows has a lengthy evaluation time

In[63]:=
```
For[i = 1;
  setval = 0, i ≤ Length[Transpose[signalpadded2]] - 71,
  i++, signalwindow2[[i]][[All]] = signalpadded2[[1]][[i ;; i + 70]];
  If[classifier2[signalwindow2[[i]][[All]]] == 1.,
   RR2[[1]][[i]] = 60 / ((i - setval) / 125);
   setval = i;
   plotposition2[[1]][[i]] = 36;
   j2[[1]][[i]] = 1,
   j2[[1]][[i]] = 0;
   If[plotposition2[[1]][[i - 1]] ≠ 0,
    plotposition2[[1]][[i]] = plotposition2[[1]][[i - 1]] - 1,];
   If[i ≠ 1, RR2[[1]][[i]] = RR2[[1]][[i - 1]],]]];
```

The results obtained from the previous step are plotted and animated. The green signal is the ECG signal while the red circle indicates the position and time instant of event occurrence.

In[64]:=
```
Animate[If[
  plotposition2[[1]][[i]] ≠ 0, Show[ListPlot[signalwindow2[[i]][[All]], Joined → True,
    PlotRange → {{0, 71}, {-150, 40}}, PlotStyle → {Directive[Green, Thick]},
    AxesStyle → Directive[Black, 12, Thick]], ListPlot[
    {{plotposition2[[1]][[i]], signalwindow2[[i]][[plotposition2[[1]][[i]]]]}},
    PlotMarkers → Graphics[{Red, Circle[]}, ImageSize → 10]],
   Graphics[Text["Heartrate = " <> ToString@N[RR2[[1]][[i]]] <> "bpm",
     {55, -70}, BaseStyle → {FontWeight → "Bold", FontSize → 12}]]],
  Show[ListPlot[signalwindow2[[i]][[All]], Joined → True,
    PlotRange → {{0, 71}, {-150, 40}}, PlotStyle → {Directive[Green, Thick]},
    AxesStyle → Directive[Black, 12, Thick]],
   Graphics[Text["Heartrate = " <> ToString@N[RR2[[1]][[i]]] <> "bpm",
     {55, -70}, BaseStyle → {FontWeight → "Bold", FontSize → 12}]]]],
 {i, 1, Length[signalwindow2] - 71, 1, AnimationRate → 35, Appearance → "Open"}]
```

Out[64]=


## REDUCING NUMBER OF PREDICTORS

Sometimes having a large number of predictors often can lead to bad classification models. Therefore, the scope reduction of predictor numbers by downsampling data frames is discussed in this section.

In[65]:=
```
trainingdata3 =
  Table[Downsample[trainingdata2[[i]][[All]], 2], {i, 1, Length[trainingdata2]}];

(*downsampling trainingdata*)
```

In[66]:=
```
classifier3 =
  Classify[trainingdata3 → traininglabel2, Method → {"SupportVectorMachine",
     "KernelType" → "Polynomial", "PolynomialDegree" → 2}];(*classifier*)
```

In[67]:=
```
testdata3 = Table[Downsample[testdata2[[i]][[All]], 2], {i, 1, Length[testdata2]}];
(*downsampling testdata*)
```

In[68]:=
```
results3 = classifier3[testdata3]; (*obtaining testdata results*)
```

In[69]:=
```
joinmat = beatbybeat[testlabel2, results3, 125];
```

In[70]:=
```
Ck3 = confusionmatrix[joinmat, Max[Length[testlabel2], Length[results3]]]
```

Out[70]=
```
{{10720, 2}, {16, 137}}
```

From the confusion matrix, we observer that the classifier exhibited a Sensitivity of 91.5% (higher than previously) and specificity of 99.9% (lower). The tradeoff between sensitivity ans specificity needs to be

discussed. For this application, I prefer a specificity of 100%, therefore I would choose classifier 2 with data frame length of 71 without downsampling. The classifier performance is checked and animated to simulate real-time peak detection conditions next using the ECG signal imported in the previous part.

In[71]:=
```
signalwindow3 = ConstantArray[0, {Length[Transpose[signalpadded2]], 36}];
```

In[72]:=
```
j3 = ConstantArray[0, {1, Length[Transpose[signalpadded2]]}];
```

In[73]:=
```
plotposition3 = ConstantArray[0, {1, Length[Transpose[signalpadded2]]}];
```

In[74]:=
```
RR3 = ConstantArray[0, {1, Length[Transpose[signalpadded2]]}];
```

Warning-The code that follows has a lengthy evaluation time

In[75]:=
```
For[i = 1;
  setval = 0, i ≤ Length[Transpose[signalpadded2]] - 70, i++,
  signalwindow3[[i]][[All]] = Downsample[signalpadded2[[1]][[i ;; i + 70]], 2];
  If[classifier3[signalwindow3[[i]][[All]]] == 1.,
   RR3[[1]][[i]] = 60 / ((i - setval) / 125);
   setval = i;
   plotposition3[[1]][[i]] = 19;
   plotposition3[[1]][[i + 1]] = 0;
   j3[[1]][[i]] = 1,
   j3[[1]][[i]] = 0;
   If[plotposition3[[1]][[i - 1]] == 0 && plotposition3[[1]][[i - 2]] ≠ 0,
    plotposition3[[1]][[i]] = plotposition3[[1]][[i - 2]] - 1;
    plotposition3[[1]][[i + 1]] = 0,];
   If[i ≠ 1, RR3[[1]][[i]] = RR3[[1]][[i - 1]],]]];
```

In[76]:=
```
Animate[If[
  plotposition3[[1]][[i]] ≠ 0, Show[ListPlot[signalwindow3[[i]][[All]], Joined → True,
    PlotRange → {{0, 36}, {-150, 40}}, PlotStyle → {Directive[Green, Thick]},
    AxesStyle → Directive[Black, 12, Thick]], ListPlot[
    {{plotposition3[[1]][[i]], signalwindow3[[i]][[plotposition3[[1]][[i]]]]}},
    PlotMarkers → Graphics[{Red, Circle[]}, ImageSize → 10]],
   Graphics[Text["Heartrate = " <> ToString@N[RR3[[1]][[i]]] <> "bpm",
     {25, -70}, BaseStyle → {FontWeight → "Bold", FontSize → 12}]]],
  Show[ListPlot[signalwindow3[[i]][[All]], Joined → True,
    PlotRange → {{0, 36}, {-150, 40}}, PlotStyle → {Directive[Green, Thick]},
    AxesStyle → Directive[Black, 12, Thick]],
   Graphics[Text["Heartrate = " <> ToString@N[RR3[[1]][[i]]] <> "bpm",
     {25, -70}, BaseStyle → {FontWeight → "Bold", FontSize → 12}]]],
 {i, 1, Length[signalwindow3] - 35, 1, AnimationRate → 35, Appearance → "Open"}]
```

Out[76]=


# DATA FUSION BASED ON SIGNAL QUALITY

Sometimes the heart rate obtained can be unreliable due to noisy signals or flat lines. Therefore for improving reliability of heartrate obtained, signal quality indicators of both PPG and ECG signals are used to selecting the most reliable heartrate. If the signal quality indicator of PPG signal is higher than that of ECG signal, the heartrate obtained from PPG signal is displayed and vice versa. The calculation of Signal Quality Indicators is outside the scope of this project and therefore already available signal quality indicator is imported and used.

In[77]:=
```
ecgsqis = Import["C:\\Users\\Arlene John\\Desktop\\data\\ecgsqi.mat"];
(*Importing ECG SQI and converting to vector
 with array padding for initial few zeros in signal*)
```

In[78]:=
```
ecgsqis = ecgsqis[[1]][[All]][[All]];
```

In[79]:=
```
ecgsqis = ArrayPad[ecgsqis, {{0, 0}, {70, 0}}];
```

In[80]:=
```
ppgsqis = Import["C:\\Users\\Arlene John\\Desktop\\data\\ppgsqi.mat"];
(*Importing ECG SQI and converting to vector with array padding*)
```

In[81]:=
```
ppgsqis = ppgsqis[[1]][[All]][[All]];
```

In[82]:=
```
ppgsqis = ArrayPad[ppgsqis, {{0, 0}, {70, 0}}];
```

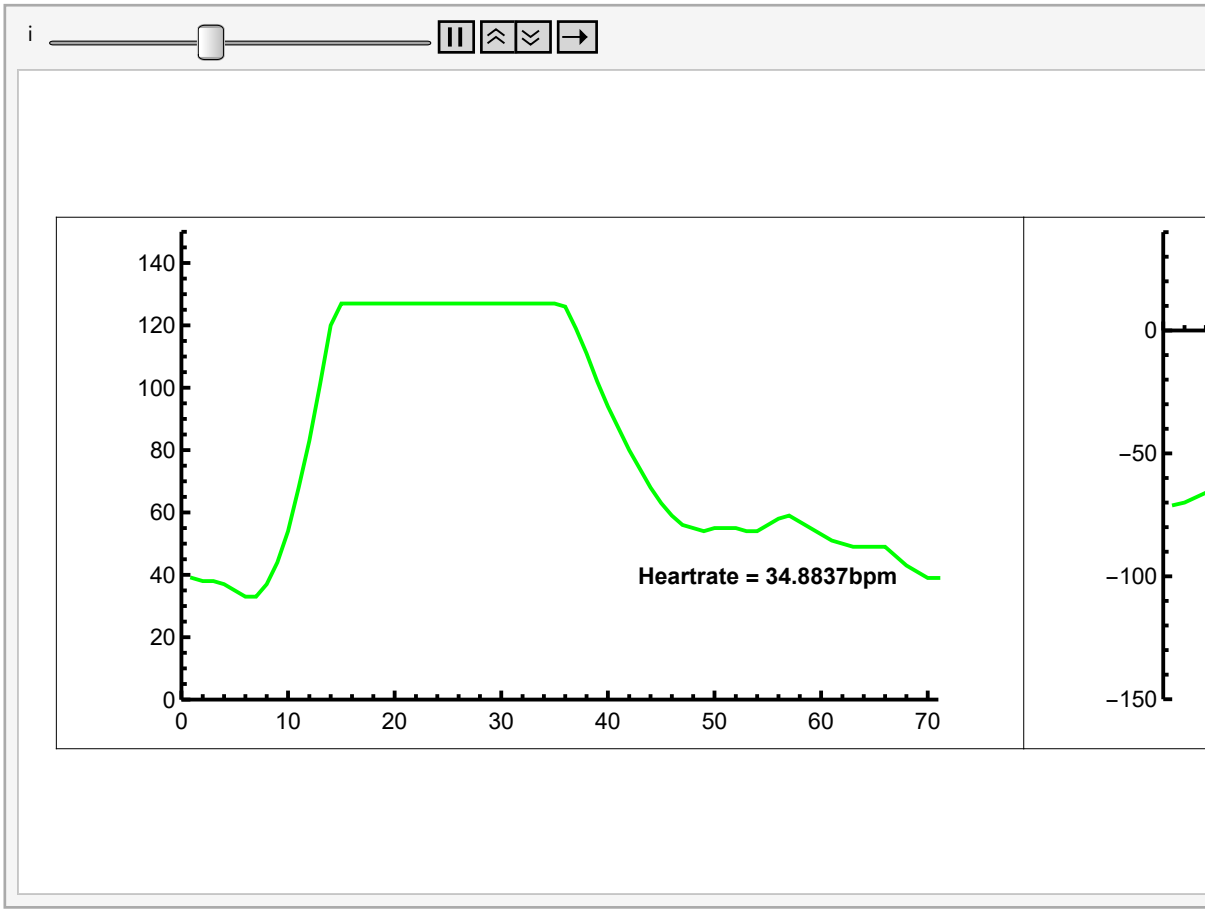
In[83]:=
```
RRfinal = ConstantArray[0, {1, Length[Transpose[RR1]]}];
```

In[84]:=
```
For[i = 1, i ≤ (Length[Transpose[RR1]] - 71), i = i + 71,
  If[Total[ecgsqis[[1]][[i ;; i + 70]]] >= Total[ppgsqis[[1]][[i ;; i + 70]]],
   RRfinal[[1]][[i ;; i + 70]] = N[RR2[[1]][[i ;; i + 70]]],];
  If[Total[ecgsqis[[1]][[i ;; i + 70]]] < Total[ppgsqis[[1]][[i ;; i + 70]]],
   RRfinal[[1]][[i ;; i + 70]] = N[RR1[[1]][[i ;; i + 70]]],];]
(*storing previously calculated heart rates to final heart
 rate depending on signal quality*)


(*Animating*)
```

In[85]:=
```
Animate[GraphicsRow[{If[
    plotposition1[[1]][[i]] ≠ 0, Show[ListPlot[signalwindow1[[i]][[All]], Joined → True,
      PlotRange → {{0, 71}, {0, 150}}, PlotStyle → {Directive[Green, Thick]},
      AxesStyle → Directive[Black, 12, Thick]], ListPlot[
      {{plotposition1[[1]][[i]], signalwindow1[[i]][[plotposition2[[1]][[i]]]]}},
      PlotMarkers → Graphics[{Red, Circle[]}, ImageSize → 10]],
     Graphics[Text["Heartrate = " <> ToString@N[RR1[[1]][[i]]] <> "bpm",
       {55, 40}, BaseStyle → {FontWeight → "Bold", FontSize → 12}]]],
    Show[ListPlot[signalwindow1[[i]][[All]], Joined → True,
      PlotRange → {{0, 71}, {0, 150}}, PlotStyle → {Directive[Green, Thick]},
      AxesStyle → Directive[Black, 12, Thick]],
     Graphics[Text["Heartrate = " <> ToString@N[RR1[[1]][[i]]] <> "bpm",
       {55, 40}, BaseStyle → {FontWeight → "Bold", FontSize → 12}]]]], If[
    plotposition2[[1]][[i]] ≠ 0, Show[ListPlot[signalwindow2[[i]][[All]],
      Joined → True, PlotRange → {{0, 71}, {-150, 40}}, PlotStyle →
       {Directive[Green, Thick]}, AxesStyle → Directive[Black, 12, Thick]], ListPlot[
      {{plotposition2[[1]][[i]], signalwindow2[[i]][[plotposition2[[1]][[i]]]]}},
      PlotMarkers → Graphics[{Red, Circle[]}, ImageSize → 10]],
     Graphics[Text["Heartrate = " <> ToString@N[RR2[[1]][[i]]] <> " bpm",
       {55, -70}, BaseStyle → {FontWeight → "Bold", FontSize → 12}]]],
    Show[ListPlot[signalwindow2[[i]][[All]], Joined → True,
      PlotRange → {{0, 71}, {-150, 40}}, PlotStyle → {Directive[Green, Thick]},
      AxesStyle → Directive[Black, 12, Thick]],
     Graphics[Text["Heartrate = " <> ToString@N[RR2[[1]][[i]]] <> " bpm",
       {55, -70}, BaseStyle → {FontWeight → "Bold", FontSize → 12}]]]],
   Graphics[Text["Heartrate = " <> ToString@N[RRfinal[[1]][[i]]] <> " bpm",
     {55, -70}, BaseStyle → {FontWeight → "Bold", FontSize → 20}]]},
  100, Frame → All, ImageSize → {1300, 400}], {i,
  1,
  Length[signalwindow1] - 71,
  1,
  AnimationRate → 35,
  Appearance → "Open"}]
```

Out[85]=




To conclude, even detection using SVM works quite well but the challenge of finding an optimal classifier with the perfect balance between sensitivity and specificity is yet to be explored.