Group 5: Heart Rate Prediction using Electrocardiography (ECG) Project 1

Suraj Shah	Sita Rama Nikitha Pabolu	Satya Srinija Kanteti	Akhila Muthyala
1211235204	1213420986	1213202430	1213175494
ssshah22@asu.edu	spabolu1@asu.edu	skantet1@asu.edu	amuthyal@asu.edu

ABSTRACT

Mobile ambulatory cardiac monitoring is an important application in cardiac health monitoring and maintenance. Heart rate is one of the most important factors in determining several disorders such as stroke. Bradycardia is one very significant problem in several individuals that is a precursor to future heart related problems. Our aim is to predict Bradycardia, which is "*a condition in which the resting heart rate falls below 60 beats per minute. In this condition, enough oxygen is not pumped into the heart. The characteristics of this are regular rhythm, less than 60 heart rate, normal QRS duration, visible P-wave before each QRS complex, and a normal P-R interval*". In this project, we will first collect ECG data from all four group members for a period of four days, each using a wearable sensor that was provided by the Impact Lab. Using this, we develop a heart rate prediction methodology.

INTRODUCTION

A few of the currently existing attempts to predict Bradycardia, developed by various organizations and researchers has a little probability for failure. Also, during the course of this project, it has been observed that certain heart rate determination algorithms and Bradycardia detection algorithms have failed to detect the heart rate accurately and have shown poor performance. The main goal of this project was to collect the ECG data from four patients (group members), clean and process it for usage. From this pre-processed data, in order to detect the heart rate, the peaks in the ECG signal had to be detected. After detecting the heart rate from the peaks, a Bradycardia detection algorithms has been implemented after it has occurred. "*If the computed heart rate falls below 60 beats per minute, then it is detected as Bradycardia."* Then, an Android application was developed to show the execution of this algorithm and measure the performance of it in terms of false positives and false negatives. Next considering the variance of heart rate as a precursor for Bradycardia, a prediction algorithm has been developed and this has been repeated by using a deep belief network as well. The performance for a k-fold cross validation has been reported.

IMPLEMENTATION

This project was implemented as an Android application using Java and Android Studio, and Matlab and Python for cleaning the data and peak detection. Python was used to implement a Bradycardia detection algorithm.

The workflow was split according to the four different parts present in the project description as follows:

Data Collection

By using the heart rate signal monitoring device (Faros ECG sensor) provided by the Impact Lab, each group member collected their ECG data for a period of 4 days. The sensor was wearable and each person recorded the data for a period of 8 hours during the day, which resulted in around **7,200,000 sampled data (250Hz sampling rate x 8 hours x 60 mins x 60 secs)**. This was for one member and there were 4 such files, which were in EDF format.

Data Preprocessing

This collected data had to be preprocessed to use it for peak detection and heart rate determination. It had to be cleaned for baseline correction so that

the baseline wandering is solved and also bandpass filtering so that noise is removed.

The above figure shows how the raw data before preprocessing looks like.

All the 4 EDF files were imported in Matlab using the '*edfread.m*' and its '*ReadEDF'* function. After setting the parameters like frequency, several filters like Normalized Passband, Stopband, Passband Ripple, Stopband Ripple, Chebyshev Type II Order were applied. and only the first 7,200,00 samples were considered to maintain consistency.

```
Fs = 250;  % Sampling Frequency (Hz)
Fn = Fs/2;  % Nyquist Frequency
Ts = 1/Fs;
T = [0:size(data,1)-1]*Ts;
Wp = [1  100]/Fn;  % Passband (Normalised)
Ws = [0.5  110]/Fn;  % Stopband (Normalised)
Rp = 10;  % Passband Ripple (dB)
Rs = 30;   % Stopband Ripple (dB)
[n,Ws] = cheb2ord(Wp, Ws, Rp, Rs); % Chebyshev Type II Order
[b,a] = cheby2(n, Rs, Ws); % Transfer Function Coefficients
EKG_F = filtfilt(sos,g,data);
```

These result in the filtered data in ***EKGf*** and the data is stored in a .dat file so that it can be read using Matlab's '*dlmread'* function.

One observation is that the graph we get for this filtered data gets inverted. So, to find the peaks, instead of finding the max peaks, the min peaks were calculated. Another observation was that the '*findpeaks'* function and the '*RPeak Detection algorithm'* obtained from Impact lab were not performing good for either the sample data that was given or with our own data because the baseline wander and noise still existed even after applying those functions. So, we wrote our own peak detection algorithm in Python, which gives a fairly good number of peaks (about **30,000 peaks** detected per patient in 8 hours of period).

Note: All the cleaned dat files can be found in the path - '*/Phase 2/2.1 and 2.2 with cleaned data'*

Peak Detection

The cleaned data files in .dat format are imported to the Python function '*find_peaks()'*, which gives the R-peaks. This function makes certain assumptions like the maximum achievable heart rate for a person is 210 and the minimum is 40 beats per minute. Based on the frequency, it defines the minimum distance between two R-peaks that we have to see, which will also reduce the errors even in the presence of some baseline correction. The algorithm also defines a maximum window in which the R-peaks can be detected. We found that if the minimum heart rate is 40, so 40 R-peaks in 60 secs, gives 1 R-peak in 1.5 secs and above. So, given the **window size is 1.5**, the minimum value was the current R-peak and its index is stored. Similarly, over all the 7,200,000 samples, we store the indices of all the R-peaks. A few of those indices are as follows:

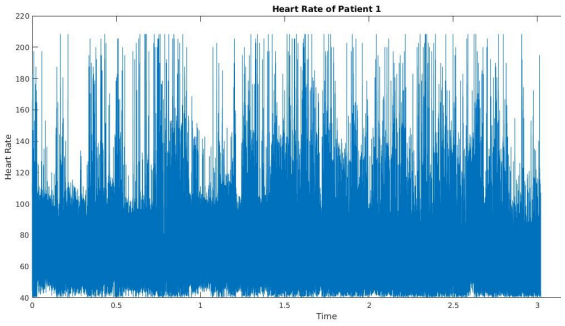
165
306
569
708
829
1024
1361
1507
1821
1927
2017...and so on.

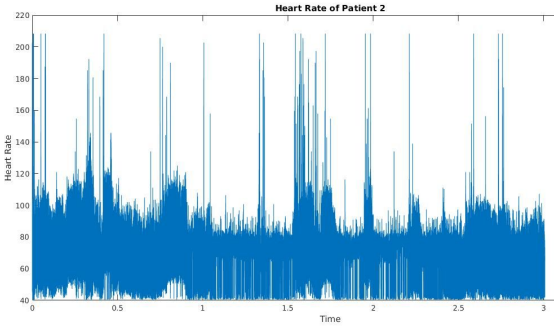
Note: The entire list of R-peak indices can be found in the respective *'_rpeaks.dat'* file in the path - '*/Phase 2/2.3'*

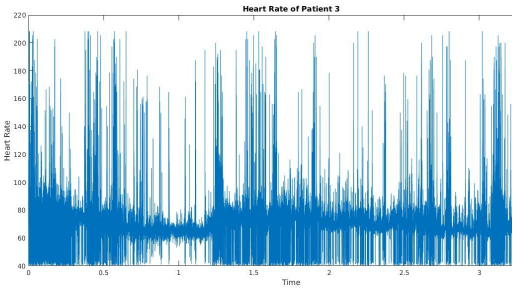
About 30,000 peaks were detected which is consistent with an actual human being's heart rate. Using these indices, the heart rate was determined, using the equation:

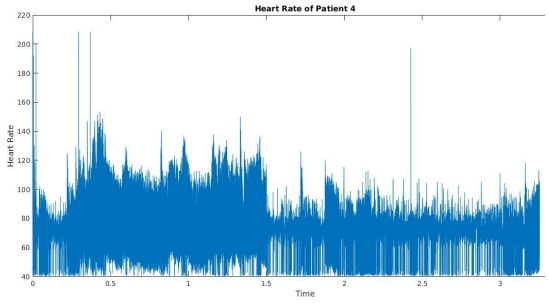
Hr = 60 x 250 / (RpeakIndex(2:end) - RpeakIndex(1:end - 1))

These heart rate values were plotted with respect to time and the plots for each patient are as follows:









It can be seen that the heart rate is within the normal range for a human.

Bradycardia Detection Algorithm

The basic working of the algorithm is that if the heart rate is less than a threshold, the person has Bradycardia, else, he does not. The chosen threshold here was **60 beats per min.**

Due to the errors present in the data, there is a large variation in the heart rate (Ex: 50 to 100, 100 to 40, to 125 and so on). To avoid this variation, the algorithm implemented by us uses time as a parameter and the heart rate is averaged over a time. How this is done is that the first heart rate is taken and the heart rate after 2 mins is taken and the values between these two heart rates are averaged. Likewise, for all the other values of heart rate, they are averaged over a period of 2 mins. This gives less varying heart rate values that are close to each other like 65, 67, 74, 60 and so on unlike before. Now, this averaged heart rate is checked and if it is less than 60, we report that it is Bradycardic, otherwise, not.

In this algorithm, the count of Bradycardic peaks, and the maximum number of Bradycardic peaks is also calculated to understand what is the maximum time for which the heart rate was continuously less than 60. This helped in choosing the appropriate time window.

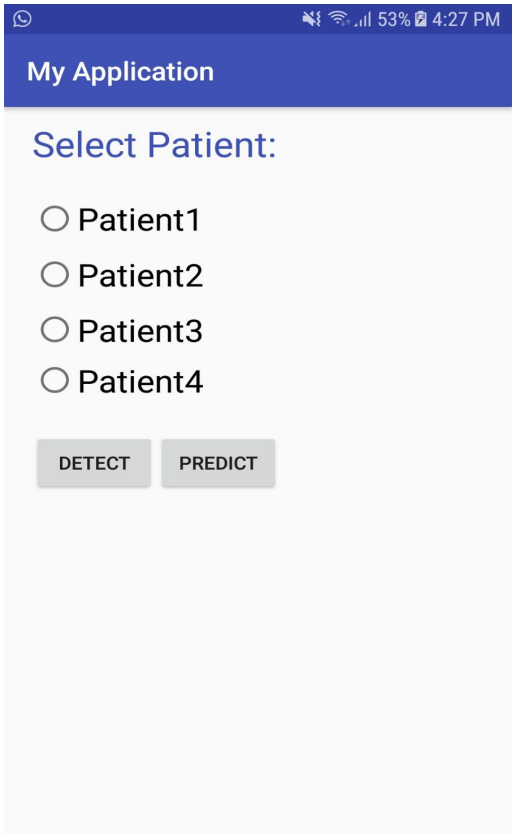
```
minute_samples = 2*60*fs
for i in range(count):
        j = i+1
        while j < count and Rpeak[j]-Rpeak[i] <
```

```
minute_samples:
            j+=1
    if j<count:
            Get the average hr
    if min(avg_hr) < 60.0:
            print("Bradycardia")
    else:
            print("No Bradycardia")
```

Android Implementation

User Interface

The UI is designed such that the user can select one of the four patients to run the bradycardia detection algorithm on him and detect whether he has Bradycardia or not.



Algorithm Execution

Once a patient is selected and the DETECT button is hit, it runs the Bradycardia detection algorithm in Java and show the results on the app. The results include - whether the person has Bradycardia or not, the execution time of the algorithm, the min avg 2-min heart rate and performance of the algorithm in terms of false positives and negatives etc.

Manual Annotation

For the purpose of computing the performance of our algorithm in relation with the actual values , manual annotation was necessary. The heart rates were manually annotated such that if the heart rate is below 60, then it is annotated as 1 (having Bradycardia) and if it is greater than 60, it is annotated as 0 (not having Bradycardia). A part of the annotations can be seen as follows:

A1

	A	B	C
1	106	0	
2	57	1	
3	107	0	
4	123	0	
5	76	0	
6	44	1	
7	102	0	
8	47	1	
9	141	0	
10	166	0	
11	44	1	
12	58	1	
13	74	0	
14	40	1	
15	54	1	
16	145	0	

Note: All the annotations can be found in the *'_heartrate.csv'* files in the path - *'Phase 2/2.4 and 2.5'*

Performance of the algorithm:

Based on the manual annotations, the performance of our detection algorithm has been evaluated. This is done by comparing the averaged 2-min values with the original values which have a lot of errors.

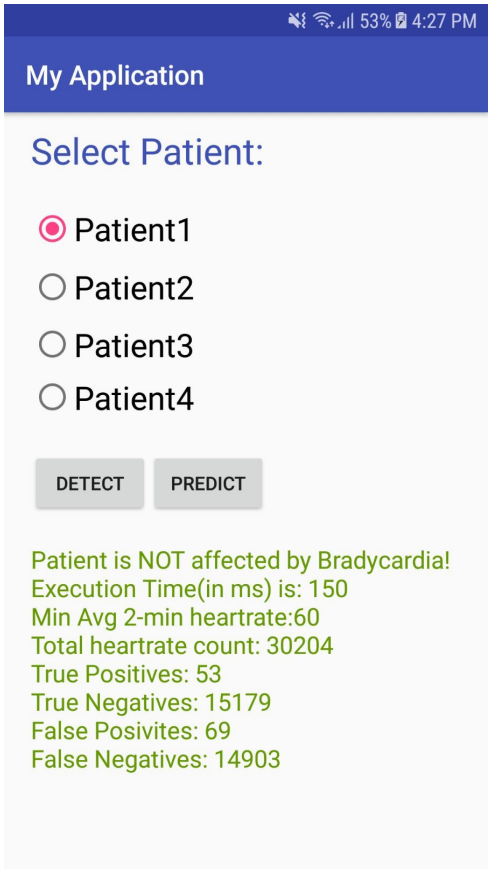
if both of them are <60, then
    True positive
if original > 60 and avg > 60, then
    True negative

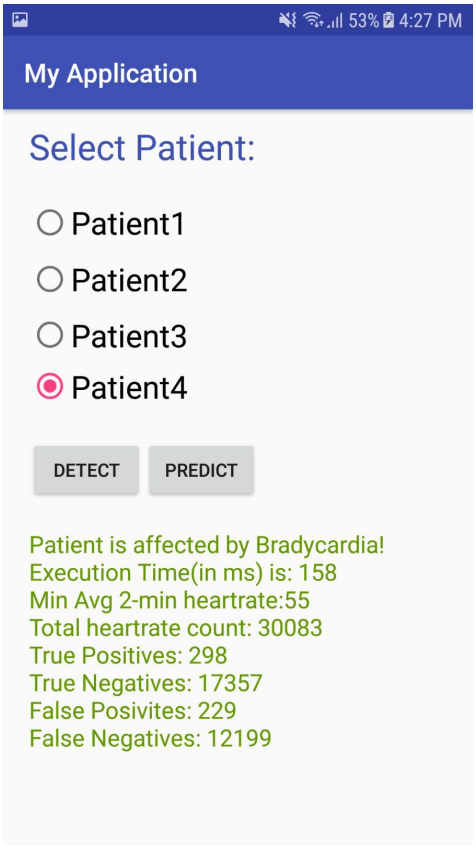
if original < 60 and avg > 60, then
False negative
If original > 60 and avg <60, then
False positive

These are displayed on the app. Since the original values have a lot of variations as discussed before, it is going to give higher number of false negatives because the averaged value will be more than the original value.

Results

Following all the above procedure, it has been detected that out of all the 4 team members, Patient 1 and Patient 2 do not have Bradycardia, with average heart rate above 60 and Patient 3 and Patient 4 have Bradycardia, with average heart rate below 60.





Note: The complete Android project can be found in '*/Phase 3/3.4 and 3.5'*

Variance of Heart Rate

We consider the variance as a precursor for Bradycardia. We found the variance of heart rate in the following steps:

We took data from "*11-10-30.dat", "11-44-37.dat", "13-12-05.dat", "13-45-41.dat"* files.

First we calculated the heart rate and average heart rate using the following code:

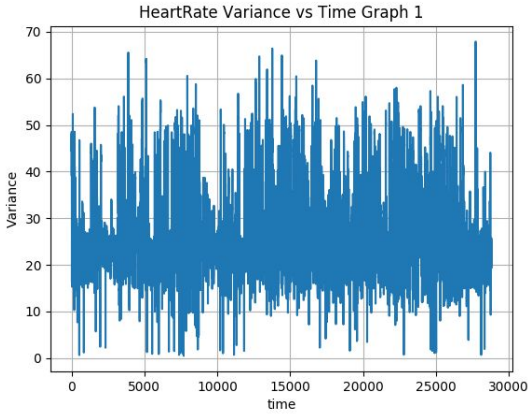
```
heartrate = [int(multiplier/(lines[i+1]-lines[i]))
for i in range(len(lines)-1)]
```

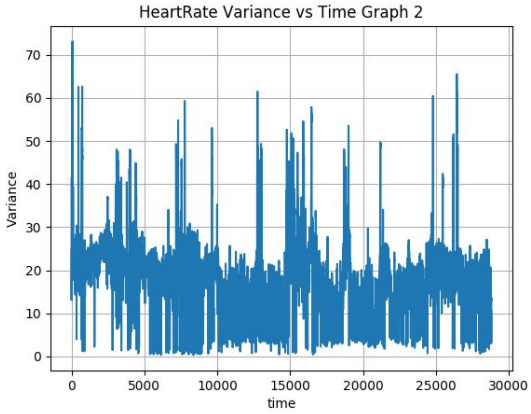
We calculated variance values in the sample window

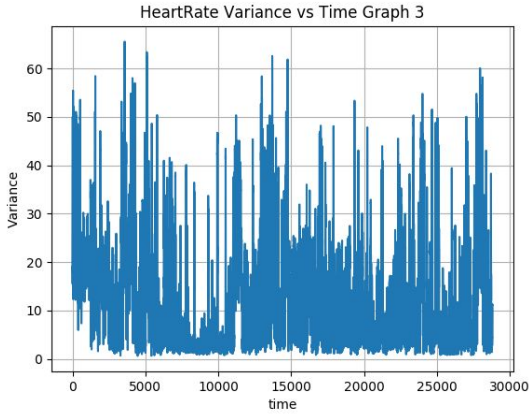
```
minute_samples = 1*10*fs
for i in range (count):
        j=i+1
```

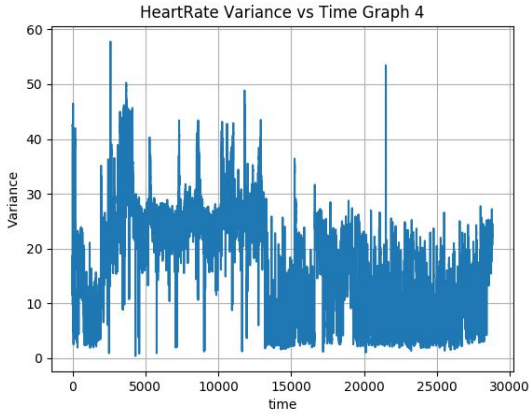
```
        while j < count and lines[j]-lines[i] <
minute_samples:
        j+=1
if j<count:
hr = sum(heartrate[i:j-1])/len(heartrate[i:j-1])
avg_hr.append(hr)
var =  math.sqrt((sum([heartrate[i]**2 for i in
range(i,j-1)])/len(heartrate[i:j-1]) -
math.pow(hr,2)))
avg_var.append(var)
```

This variance of heart rate has been plotted with respect to time.









Manual Annotation of pre-Bradycardia

During bradycardia detection, the bradycardia events were manually annotated. The events exhibiting bradycardia are given “1” and “0” otherwise. During bradycardia prediction using variance as precursor, the events are divided into 3 categories, pre-bradycardia, bradycardia and rest of the events. The pre-bradycardia events are given “2” , bradycardia events “1” and rest of them are given “0”.

A1 : HeartRate

	A	B	C	D	E
1	HeartRate	HRClass	AverageH	AvgVarian	AVHRClass
2	70	0	81.63636	42.52544	0
3	208	0	81.36364	42.60873	0
4	57	1	68.7	15.26467	0
5	43	1	69.27273	14.15557	0
6	41	1	71.81818	11.46392	0
7	83	0	71.72727	11.70929	0
8	81	0	72	12	0
9	79	0	71.1	12.22661	0
10	78	0	67.2	14.992	0
11	80	0	67.1	14.9228	0
12	78	0	65.66667	15.06283	0
13	67	0	61.55556	16.14594	2
14	66	0	59.11111	16.78918	1

Bradycardia Prediction

Variance of heart rate has been used for predicting bradycardia. The algorithm If the variance of heart rate of each patient is in the range of (average variance - 2) and (average variance + 3), then it is predicted as pre-Bradycardia, otherwise, it is not pre-Bradycardia.
Result of thresholding algorithm are as follows:

Patient 1:
True Positive: 1041
False Positive: 10789
True Negative: 14538
False Negative: 1263
Precision: 8.799661876584954%
Recall: 45.182291666666664%

Patient 2:
True Positive: 3197
False Positive: 6041
True Negative: 18125
False Negative: 894
Precision: 34.60705780471964%
Recall: 78.14715228550476%

Patient 3:
True Positive: 373
False Positive: 4332
True Negative: 26224
False Negative: 252
Precision: 7.927736450584485%
Recall: 59.68%
Patient 4:
True Positive: 518
False Positive: 4547
True Negative: 26053
False Negative: 239
Precision: 10.227048371174728%
Recall: 68.42800528401585%

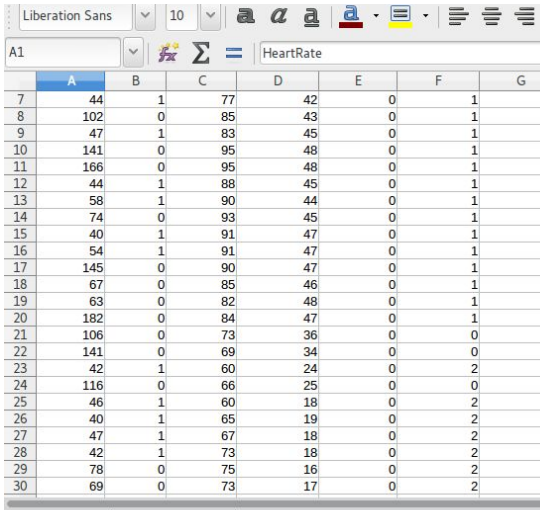
Using the above approach, the prediction algorithm can predict Bradycardia 5-10 seconds before it happens.

Note: All the average variances and pre-Bradycardia annotations can be found in *'_heartrate.csv'* files in path - *'/Phase 4/4.1 4.2 and 4.3'*

Machine learning to predict Bradycardia

Using the annotations made for pre-Bradycardia and Bradycardia, a machine learning algorithm has been used to predict Bradycardia. K-means clustering has been used with variance as a precursor for pre-Bradycardia.

The way the algorithm works is just as the basic k-means algorithm. The algorithm separates the heart rate values into three distinct and well-separated clusters based on variance of heart rate. Six iterations of k-means have been run to obtain constant clusters and respective centroids. Also, the "_kmeans.csv" files have been annotated with the k-means classes, like all the previous annotations. It can be seen as follows.
'ClusterLabel' represents the label of the cluster which the heart rate belongs to. 0 indicates no bradycardia, 1 indicates bradycardia and 2 indicates pre-Bradycardia.



	A	B	C	D	E	F	G
7	44	1	77	42	0	1	
8	102	0	85	43	0	1	
9	47	1	83	45	0	1	
10	141	0	95	48	0	1	
11	166	0	95	48	0	1	
12	44	1	88	45	0	1	
13	58	1	90	44	0	1	
14	74	0	93	45	0	1	
15	40	1	91	47	0	1	
16	54	1	91	47	0	1	
17	145	0	90	47	0	1	
18	67	0	85	46	0	1	
19	63	0	82	48	0	1	
20	182	0	84	47	0	1	
21	106	0	73	36	0	0	
22	141	0	69	34	0	0	
23	42	1	60	24	0	2	
24	116	0	66	25	0	0	
25	46	1	60	18	0	2	
26	40	1	65	19	0	2	
27	47	1	67	18	0	2	
28	42	1	73	18	0	2	
29	78	0	75	16	0	2	
30	69	0	73	17	0	2	

The annotated files for all the four patients can be found in "_kmeans.csv" files
The performance for clustering has been reported in terms of true positives, true negatives, false positives and false negatives, precision, recall and accuracy. The results are as follows:

Patient 1
True positives: 2304
False positives: 7907
True negatives: 2970
False negatives: 0
Precision: 22.5639016747%
Recall: 100.0%
Accuracy: 40.0121386845%

Patient 2
True positives: 1370
False positives: 675
True negatives: 12054
False negatives: 2721
Precision: 66.9926650367%
Recall: 33.4881447079%
Accuracy: 79.8097502973%

.
Patient 3
True positives: 423
False positives: 9292
True negatives: 19845
False negatives: 202
Precision: 4.35409161091%
Recall: 67.68%
Accuracy: 68.1002620792%

Patient 4
True positives: 757
False positives: 14448
True negatives: 1816
False negatives: 0
Precision: 4.97862545215%
Recall: 100.0%
Accuracy: 15.1166206451%

The reason for such inefficient accuracy could be because a machine learning approach is not suitable for predicting a heart condition. Several machine learning approaches including K-means clustering have predefined steps that are unrelated with the data that is being used to learn the model. In this case, the data being ECG data, it requires a separately customized algorithm for it so that it suits the data in context. Due to this reason, machine learning approaches could not give a good accuracy for prediction.

Note: All the cluster labels for k-means can be found in '*_kmeans.csv'* files in path - *'/Phase 4/4.4 4.5 and 4.6'.*

TEAM RESPONSIBILITIES

The team roles were not specifically set towards the beginning of the project and were attributed to the specific sections and subtasks that each member contributed to. Figure shows all the tasks, subtasks and the individual tasks done by each member.

Task Number	Task Description	Contributor
1	Collecting data using Impact lab sensor for 8 hours	Suraj Shah

2	Collecting data using Impact lab sensor for 8 hours	Satya Srinija Kanteti
3	Collecting data using Impact lab sensor for 8 hours	Sita Rama Nikitha Pabolu
4	Collecting data using Impact lab sensor for 8 hours	Akhila Muthala
5	Clean and denoise ECG data using filters and delete data	Suraj Shah
6	Implement peak detection algorithm in Python	Suraj Shah
7	Deriving heart rate from the detected peaks in Matlab	Satya Srinija Kanteti
8	Plotting the heart rate with respect to time in Matlab	Satya Srinija Kanteti
9	Developing a signal processing algorithm in Python for bradycardia detection	Suraj Shah
10	Manually annotating bradycardia from the obtained heart rates	Satya Srinija Kanteti
11	Evaluating the performance of bradycardia using false positives/negatives	Satya Srinija Kanteti
12	Implementing an Android application to detect the bradycardia	Sita Rama Nikitha Pabolu
13	Computing the execution time of the algorithm	Sita Rama Nikitha Pabolu
14	Computing variance of heart rate and plotting it against time	Suraj Shah Akhila Muthala
15	Implementing threshold based algorithm for bradycardia prediction	Suraj Shah
16	Annotating pre-Bradycardia	Suraj Shah
17	How far ahead can bradycardia be predicted?	Suraj Shah
18	Implementing machine learning algorithm to predict bradycardia	Satya Srinija Kanteti
19	Evaluating the performance of this algorithm	Satya Srinija Kanteti
20	Reporting how far we can predict Bradycardia using machine learning	Sita Rama Nikitha Pabolu

REFERENCES

1. Patel, Gakare, Cheeran "Real Time ECG Feature Extraction and Arrhythmia Detection on a Mobile Platform", *International Journal of Computer Applications (0975–8887) Volume 44 – No.23, April 2012*
2. Matlab Implementation of Pan Tompkins ECG QRS detector -https://www.researchgate.net/publication/313673153_Matlab_Implementation_of_Pan_Tompkins_ECG_QRS_detector
3. Pan, Tompkins, "A Real-Time QRS Detection Algorithm" *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, VOL. BME-32, NO. 3, MARCH 1985*
4. Sedghamiz, "Matlab Implementation of Pan Tompkins ECG QRS Detector",
5. Complete Pan Tompkins Implementation ECG QRS detector - https://www.mathworks.com/matlabcentral/fileexchange/45840-complete-pan-tompkins-implementation-ecg-qrs-detector?focused=9112156&tab=function
6. Detection of ECG signal using Pan Tompkins Algorithm - https://onedrive.live.com/view.aspx?resid=74D305AD22809D85!3770&app=Word
7. Chebyshev filer II Order - https://www.mathworks.com/help/signal/ref/cheby2.html