

CARDIAC ARRHYTHMIA DETECTION AND CLASSIFICATION USING ECG SIGNALS

CREATIVE AND INNOVATIVE PROJECT REPORT

Submitted by

AADITHYA NARESH(2018103501)

SIBI AKASH (2018103594)

VIVEK RAMKUMAR(2018103082)

in partial fulfillment of the requirements for the award

of the degree of

BACHELOR OF ENGINEERING

in

**COMPUTER SCIENCE AND
ENGINEERING**



COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2021

ANNA UNIVERSITY: CHENNAI 600 025**BONAFIDE CERTIFICATE**

Certificate that this project request titled **Cardiac Arrhythmia Detection using ECG signals** the bonafide work of Aadithya Naresh(2018103501), Sibi Akash (2018103594), Vivek Ramkumar(2018103082) who carried out the project work under my supervision, for the fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering. Certified further that to the best of my knowledge, the work reported herein does not form part of any other thesis on the basis of which a degree or an award was conferred on an earlier occasion on these or any other candidates.

Place: Chennai

Mrs. Lalitha Devi K

Date:25-05-2021

Teaching Fellow,

Department of Computer Science
and Engineering,

Anna University, Chennai-

25.

COUNTERSIGNED

Dr. Valli S,

Head of the Department,

Department of Computer Science and

Engineering, Anna University Chennai,

Chennai - 600 025.

ACKNOWLEDGEMENT

Foremost, we would like to express our sincere gratitude to our project guide, **Mrs. Lalitha Devi K**, Teaching Fellow, Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for her constant source of inspiration. We thank her for the continuous support and guidance which was instrumental in taking the project to successful completion.

We are grateful to **Dr.S.Valli** , Professor and Head, Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for her support and for providing necessary facilities to carry out for our project.

We would also like to thank our friends and family for their encouragement and continued support. We would also like to thank the Almighty for giving us the moral strength to accomplish our task.



Aadithya Naresh-

Scanned with CamScanner



Vivek Ramkumar-



Sibi Akash -

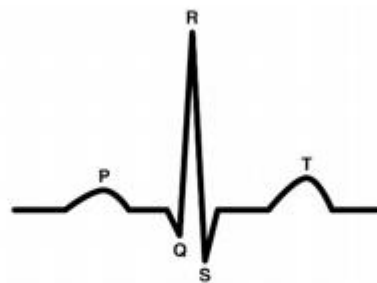
Table of Contents

1. Introduction.....	6
1.1 What is Arrhythmia?	6
1.2 Normal ECG signal	6,7
1.2 Types of Arrhythmia	7
1.3 Problem Statement	7
2. Literature Survey.....	7,8
3. Block Diagram	8
4. Module wise pseudocode with input/output	9
4.1 Module 1 - Preprocessing	9
4.2 Module 2 - Feature Extraction – Detection of R waves	9
4.3 Module 3 - Feature Extraction – Detection of QRS complex	9
4.4 Module 4 - Feature Extraction – Detection of P waves	9
4.5 Module 5 - Classifier models:	10
4.6 Module 6 - Accuracy and Validation	10
5. Screenshot of each module/Intermediate Results.....	10
5.1 Module 1 Preprocessing.....	10,11
5.1.1 Code Screenshot	11
5.1.2 Outputs	
5.1.2.1 Record 1	11,12
5.1.2.2 Record 2	12
5.1.2.3 Record 3	12,13
5.1.2.4 Record 4	13
5.1.2.5 Record 5	14
5.2 Module 2 - Feature Extraction – Detection of R waves	14
5.2.1 Code Screenshot	14
5.2.2 Outputs	
5.2.2.1 Record 100	15
5.2.2.2 Record 101	15
5.2.2.3 Record 102	16
5.2.2.4 Record 103	16
5.2.3 R peak point detection code	16

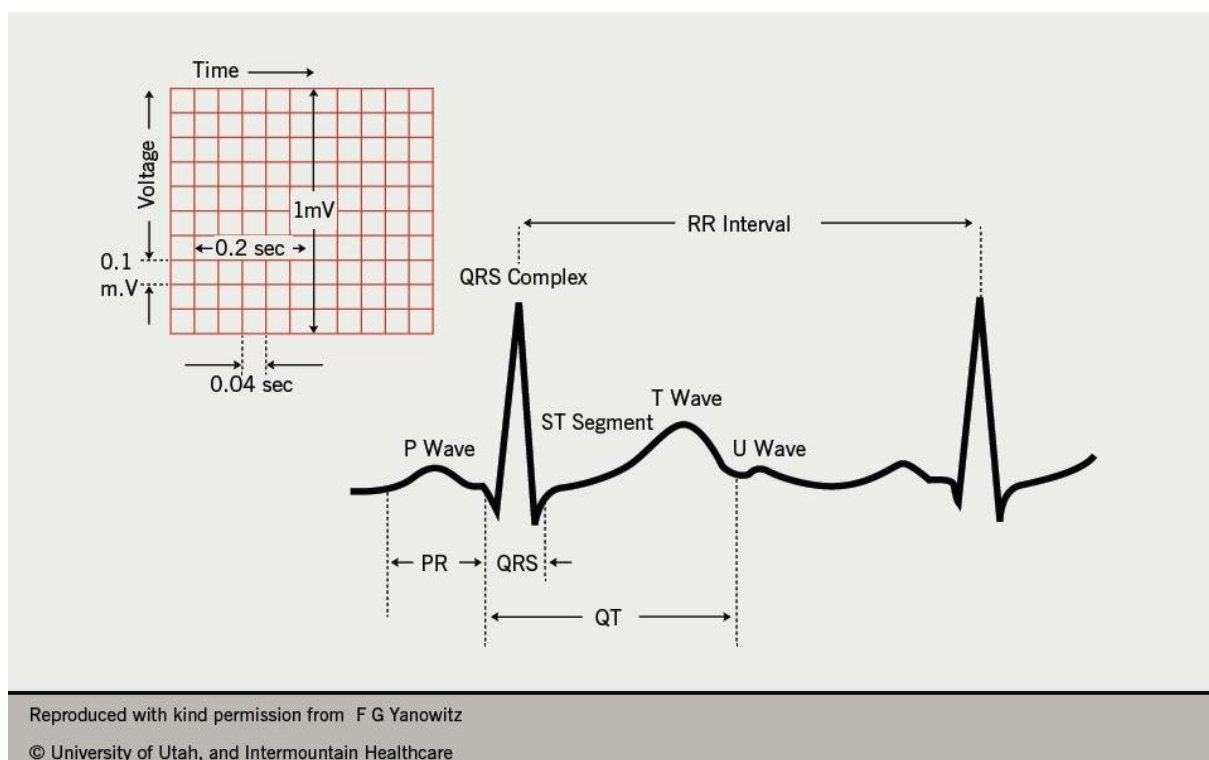
5.2.4 R peak point detection output	17
5.3 Module 3 - Feature Extraction – Detection of QRS complex	18
5.3.1 Code Screenshot	18
5.3.2 Outputs	
5.3.2.1 Record 101	18
5.3.2.2 Record 118	19
5.4 Module 4 - Feature Extraction – Detection of P waves	19
5.4.1 Code Screenshot	19
5.4.2 Outputs	
5.4.2.1 Record 101	20
5.4.2.2 Record 118	20
5.5 Module 5 - Classifier models:	21
5.5.1 Logistic regression and Decision Tree Classifier.....	21
5.5.2 Random Forest Classifier and Naïve Bayes Classifier...	21
5.5.3 SVM and Neural Network Classifier	22
5.6 Module 6 - Accuracy and Validation	22
5.7 Module 7 - Accuracy Comparison Chart	23
6. Innovation	23
7. Conclusion	23,24
8. References	24

Introduction

Cardiac arrhythmia is a group of conditions in which the heartbeat is irregular, too fast, or too slow. Arrhythmias occur due to problems with the electrical conduction system of the heart. The proposed system is going to discuss and explain the prediction of abnormal beats in a patient and represent it using an ECG graph. Then after detecting the abnormal beats, we will train two deep learning models based on the abnormal beats to determine which of the two models will provide us with better performance. The performance can be measured with a number of factors like accuracy, precision etc.



Normal/healthy ECG waveform. Where, P-wave, QRS-complex and T-wave represent the contraction/depolarization of atria, contraction/depolarization of ventricles and repolarization of ventricles respectively.



Some types of arrhythmia:

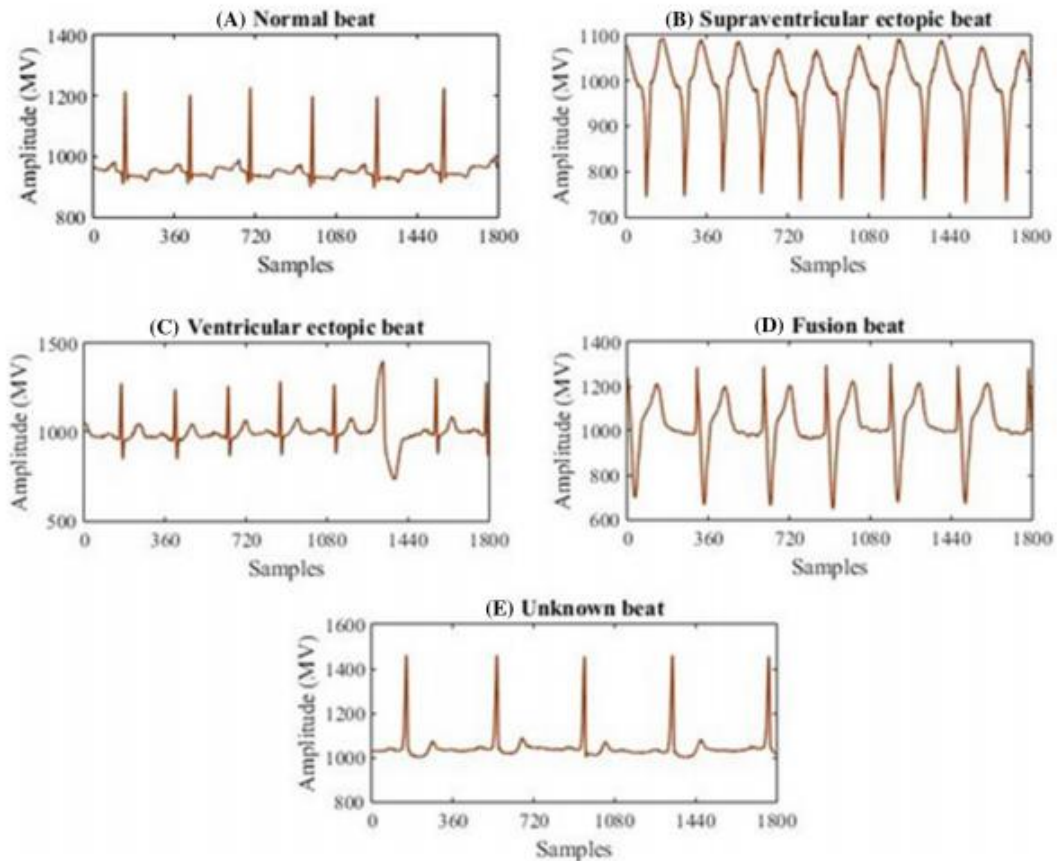


Fig. 7 Five types of heartbeat signal wave representation according to the ANSI/AAMI standards

Problem Statement:

The task we have on hand is to do an on-line cardiac arrhythmia detection. However, many ECG analysis results reported are confined to traditional, off-line, PC-based operation.

Hence, In this work, we are concerned with two problems. First, we want to classify a record into binary classes (normal or abnormal). Second, we want to classify a record into multiple classes, depending on the specific case of arrhythmia present (multi-class classification). We construct a neural network topology using significant hyperparameter tuning to achieve the desired results.

Literature Survey

Albert Haque [7] proposes the following system,

Identifying patterns in arrhythmia has been studied for several years and many statistical approaches have been attempted. These approaches can be grouped into two categories: (i) statistical learning based on explicit feature extraction and (ii) recognizing patterns from the raw time series data. Most attempts fall into the first category of extracting features using human intuition. Many studies use classical

machine learning algorithms such as support vector machines. The second category of approaches are centered around time series analysis. Time series approaches use wavelet transforms and attempt to minimize the noise present in the data. Some models note the periodic interval between the QRS complex and PR/QT intervals. Autoregressive neural networks have also been proposed for forecasting time series data.

The results obtained by the model proposed by him are as follows,

In the multi-class case, the neural network still outperforms SVMs and logistic regression with a test accuracy of 75.7%.

Table 1. Classification Accuracy

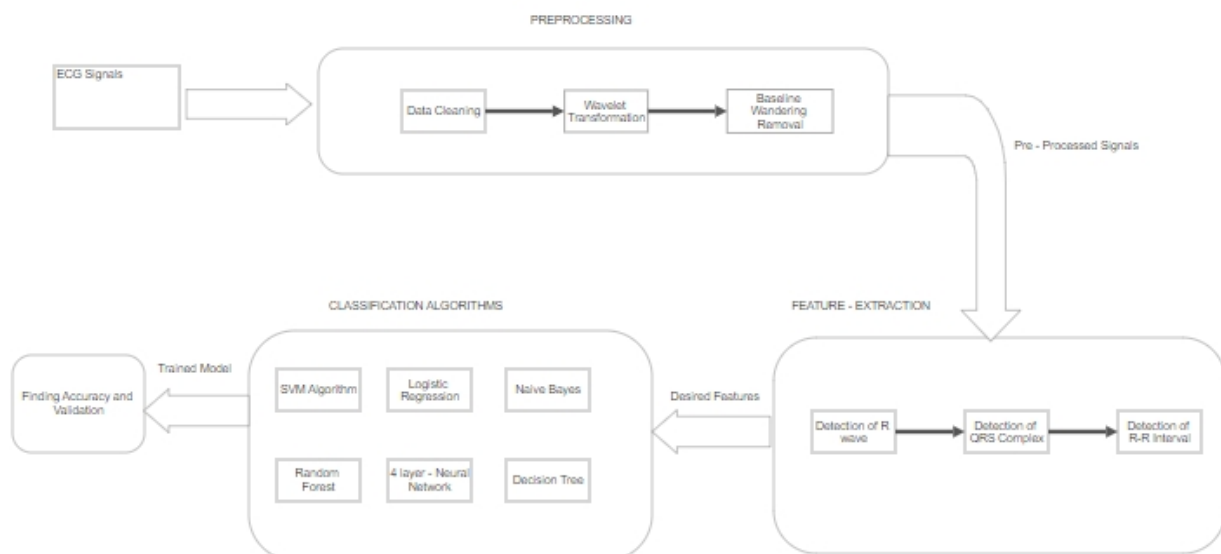
Model	Binary		Multi-Class	
	Training	Test	Training	Test
Neural Network	100.0%	91.9%	100.0%	75.7%
SVM	92.4%	87.5%	96.6%	65.1%
Random Forest	99.7%	72.0%	97.0%	76.0%
Logistic Regr.	100.0%	77.6%	100.0%	69.0%

Pranav Rajpurkar,Awni Y. Hannun,Masoumeh Haghpanahi,Codie Bourn,Andrew Y. Ng [8] proposes a different model architecture as shown below.

We use a convolutional neural network for the sequence-to-sequence learning task. The high-level architecture of the network is shown in Figure 2. The network takes as input a time-series of raw ECG signal, and outputs a sequence of label predictions. The 30 second long ECG signal is sampled at 200Hz, and the model outputs a new prediction once every second. We arrive at an architecture which is 33 layers of convolution followed by a fully connected layer and a softmax.

The results obtained by them are satisfactory and on par with human level analysis.

Block Diagram



Modules with input and output

Module 1: Preprocessing:

Input: Ecg signal dataset

Output: Baseline wander frequency normalized Ecg signal dataset

Tools used: Matlab

The MIT-BIH Arrhythmia dataset is processed using Discrete Wave Transformation (DWT) to remove the baseline wander frequency. First, the baseline of the signal is subtracted. Additionally, some noise removal can be done. Two median filters are applied for this purpose, of 200-ms and 600-ms. These values depend on the frequency sampling of the signal. The signal resulting from the second filter operation contains the baseline wanderings and can be subtracted from the original signal. Now, we have baseline wander frequency normalized ecg signals.

Module 2: Feature Extraction – Detection of R waves:

Input: Normalized Ecg signal dataset

Output: R-R interval for each beat (in s), Position of R referenced to beginning of the segment.

Tools used: Matlab

The R peak can be located from the pair maximum-minimum zero-crossings. R-R interval is the time interval between successive R-peaks in a heartbeat. This is an important feature in classifying an ecg signal.

Module 3: Feature Extraction – Detection of QRS complex:

Input: Normalized Ecg signal dataset

Output: QRS duration for each beat (in s)

Tools used: Matlab

The beginning of a QRS complex is the first local maximum before an R peak and the end is the first local minimum after an R-peak. In the case where there is not an S wave component detectable, the search is extended until the extreme on R peak is reached.

Module 4: Feature Extraction – Detection of P waves:

Input: Normalized Ecg signal dataset

Output: PR interval for each beat (in s), Number of P waves for each beat.

Tools used: Matlab

Identify the energy content of the P and T waves in the frequency range from 2 Hz to 13 Hz. P and T waves are sought in the fourth scale in the range between two QRS complexes. Once the QRS complex has been identified, in a similar manner to that

presented, the procedure uses a search window defined by the end of a QRS complex and the beginning of the next QRS complex.

Module 5: Classifier models:

Input: Extracted features annotated with arrhythmia class for each data segment.

Output: Predicted classes for test data.

Tools used: Python, Jupyter Notebook.

Once individual waves have been located, a set of features for each heartbeat is obtained, and the results for the data segment are stored in an array. We use classifier models such as Neural Networks, Svm and Logistic regression. Using these models we predict arrhythmia for the split test data and also classify what kind of Arrhythmia is present.

Module 6: Accuracy and Validation:

Input: Predicted and actual arrhythmia classes.

Output: Accuracy of the classifier model

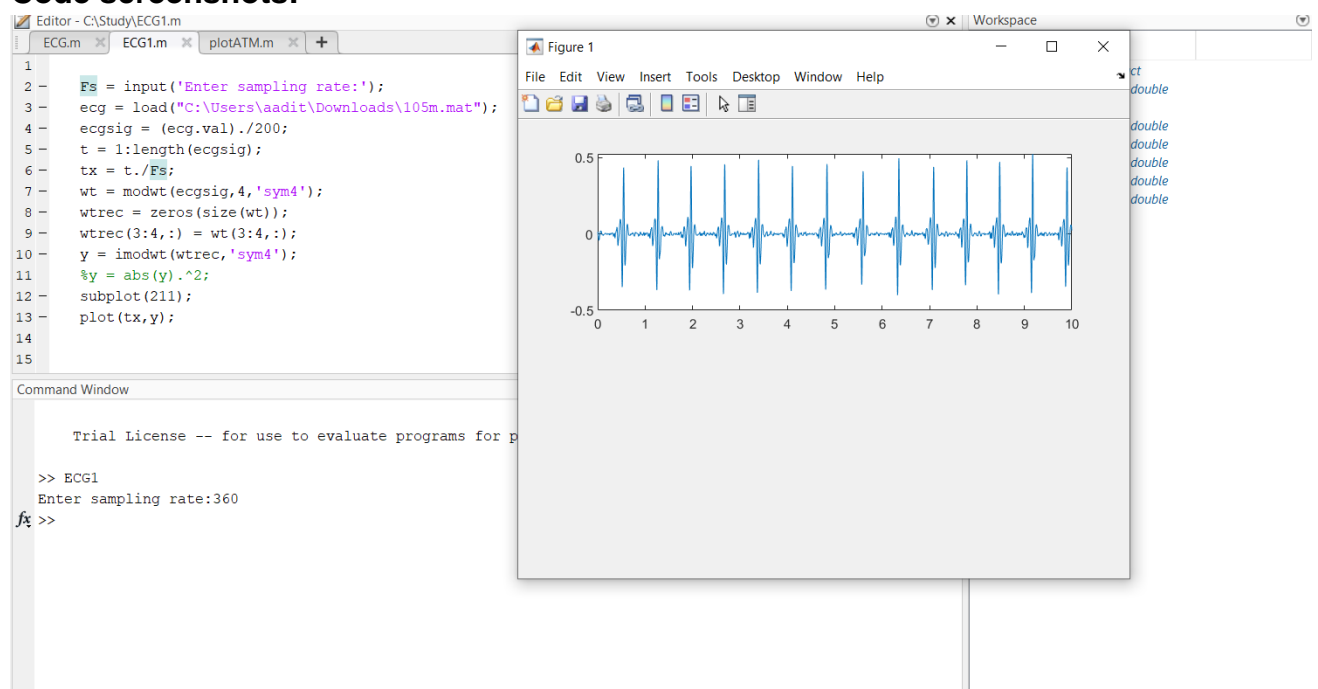
Tools used: Python, Jupyter Notebook.

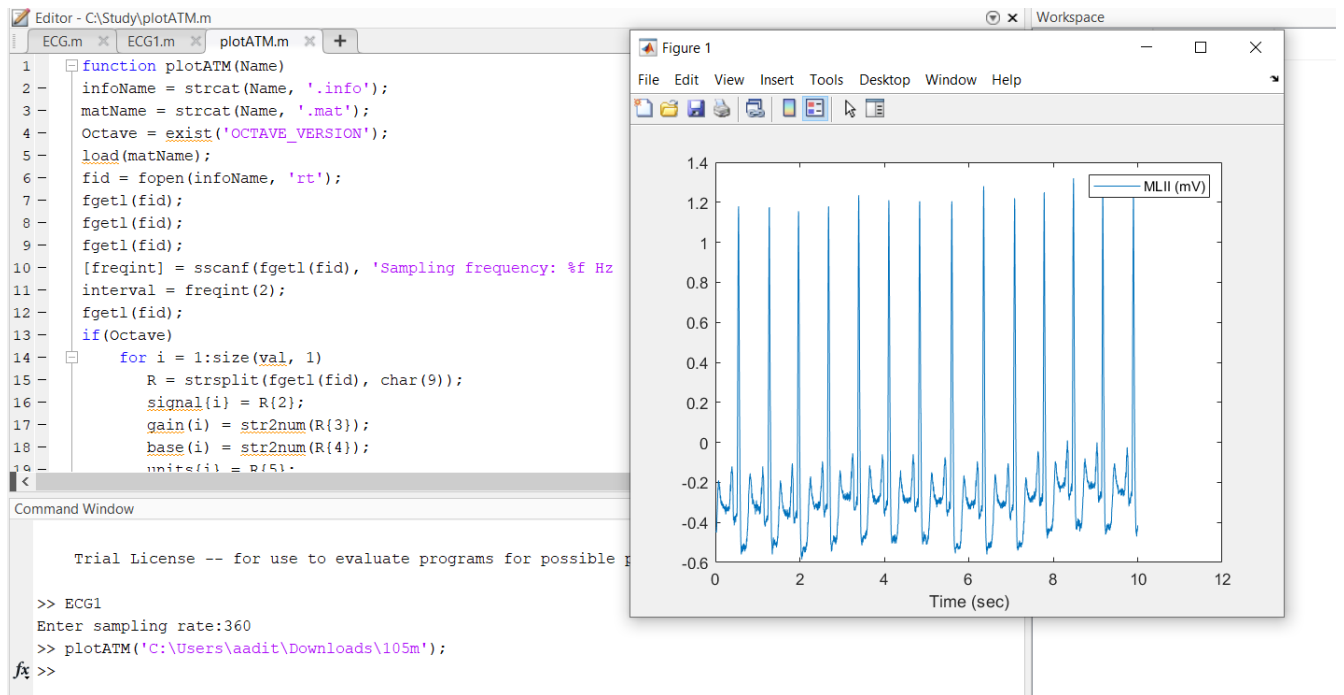
Classified ecg signals from the classifier models are validated and then accuracy of the model is determined. The accuracy of these models are compared and the best model is determined.

Screenshot of each module:

Module 1 - Preprocessing:

Code screenshots:

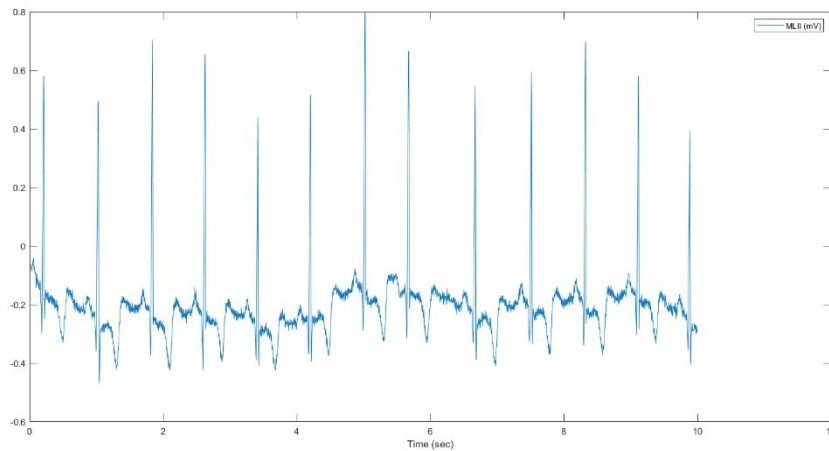




Outputs:

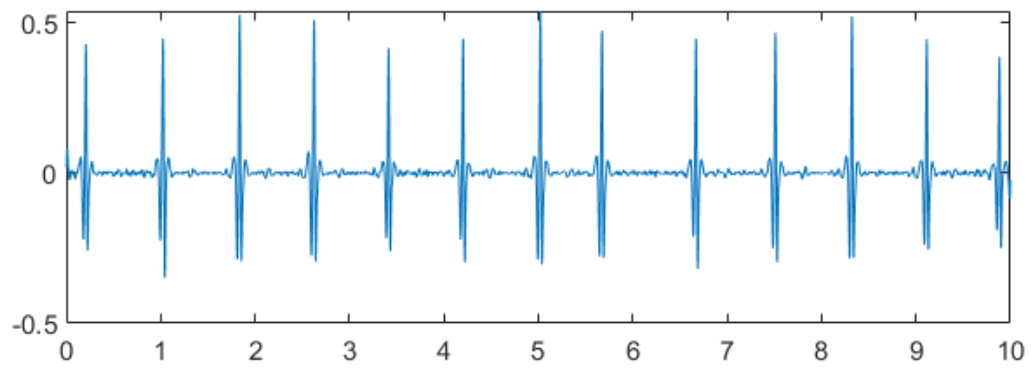
Record 1:

With baseline wandering:



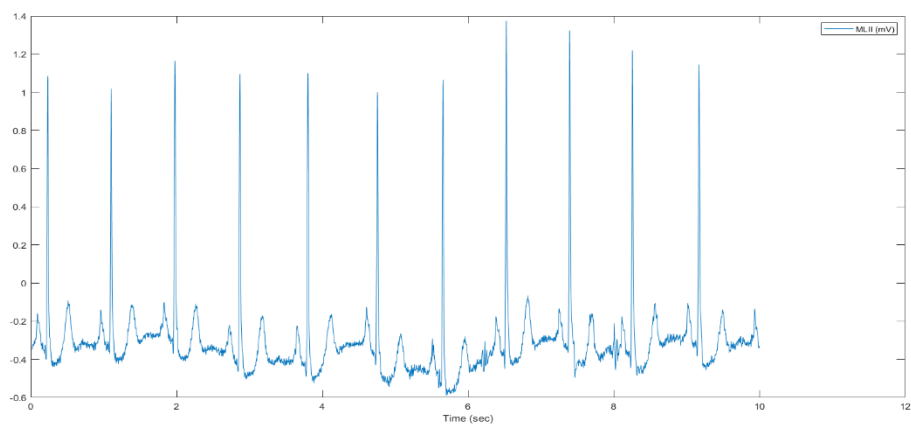
Without Baseline wandering:

P.T.O

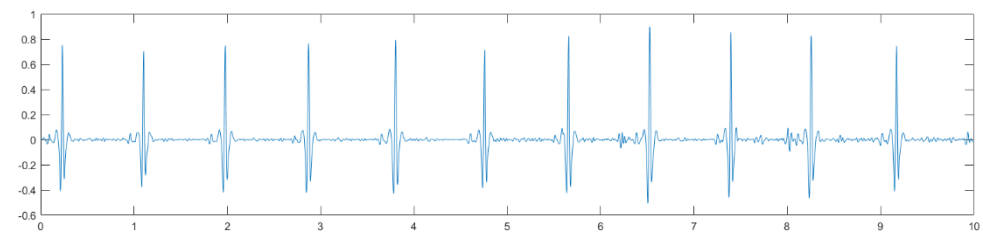


Record 2:

With baseline wandering:

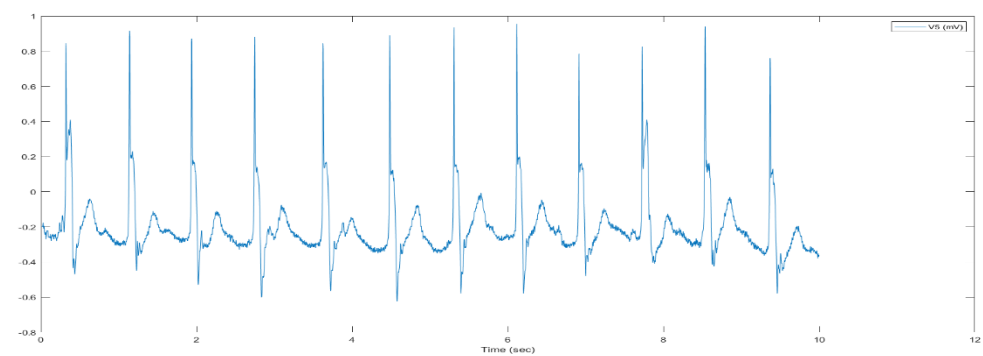


Without baseline wandering:

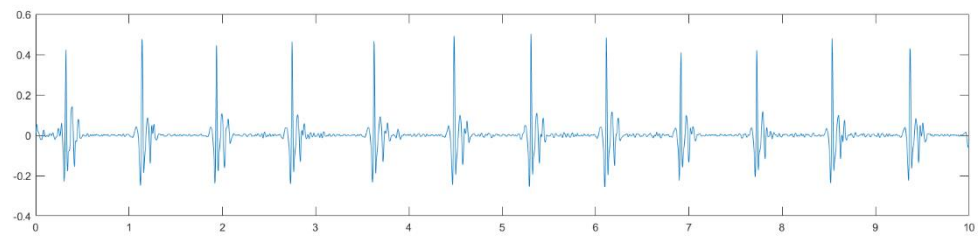


Record 3:

With baseline wandering:

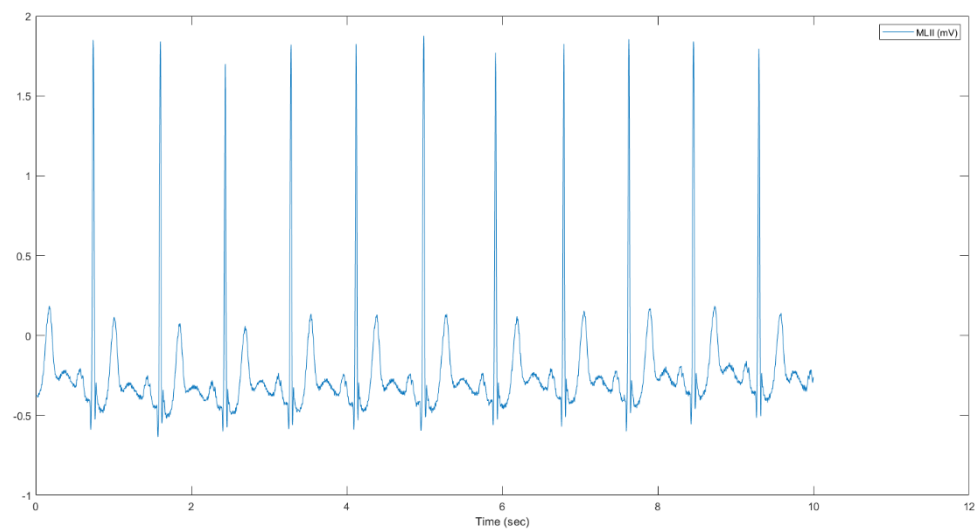


Without baseline wandering:

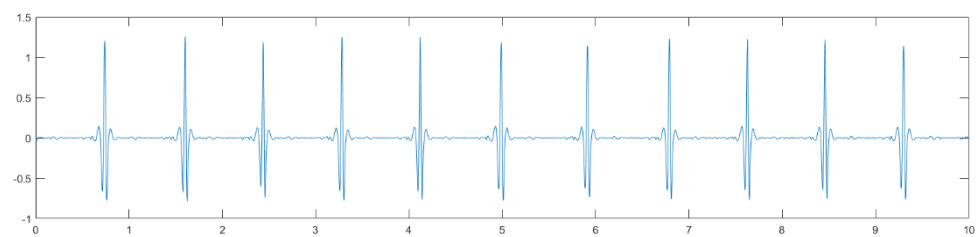


Record 4:

With baseline wandering:

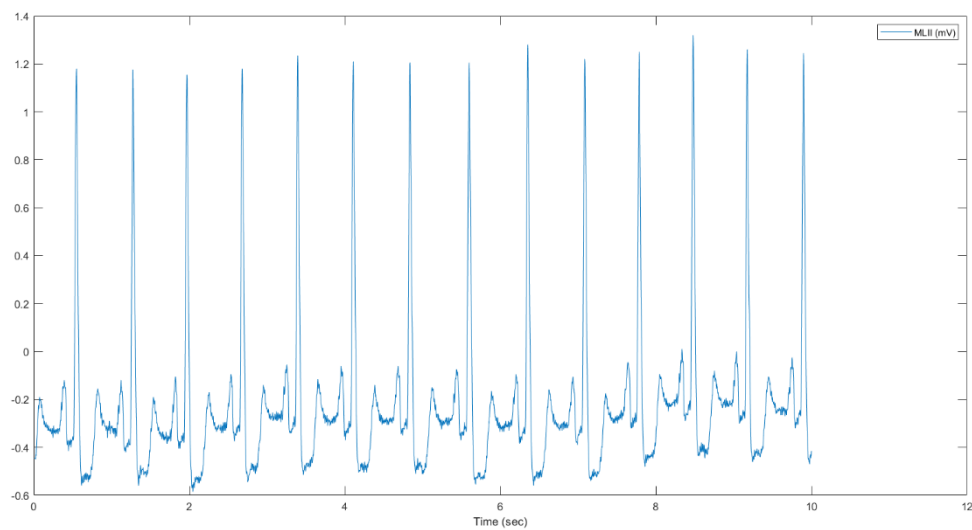


Without baseline wandering:

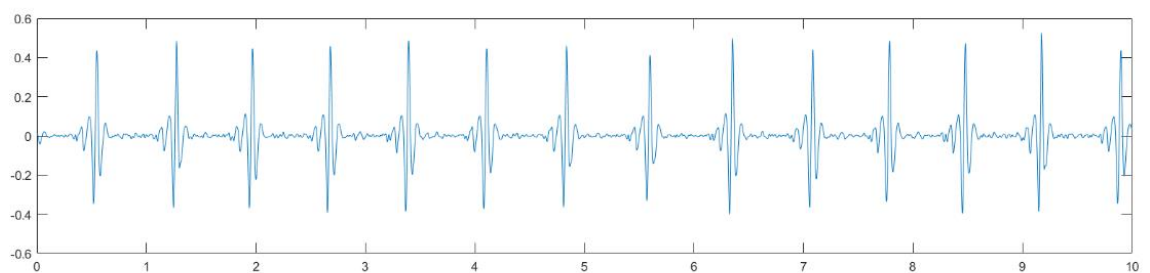


Record 5:

With baseline wandering:



Without baseline wandering:

**Module 2 - Detection of R waves****Code screenshot**

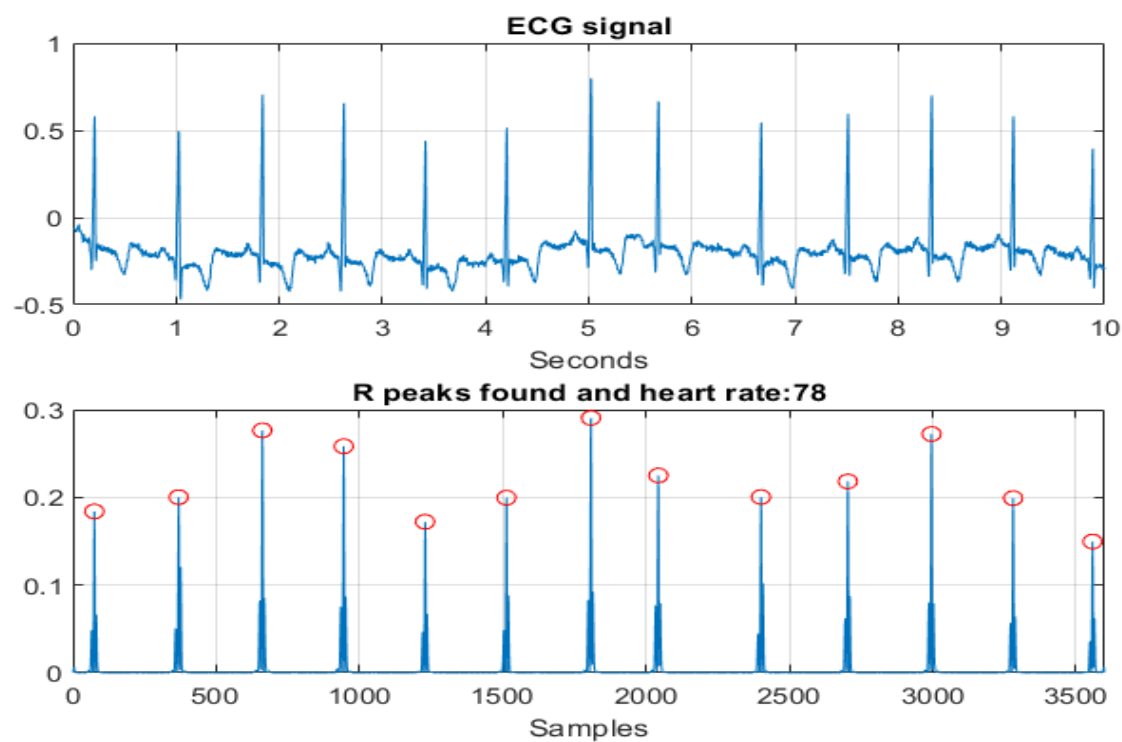
```

1 load mit200;
2 ecgsig1 = ecgsig;
3 tml = tm;
4 disp(strcat("first",num2str((ecgsig))));
5 %disp(num2str((tml)));
6
7 ecg = load("C:\Users\aadit\Downloads\200m.mat");
8 ecgsig = (ecg.val)./200;
9 t = 1:length(ecgsig);
10
11 disp(strcat("Second",num2str((ecgsig))));
12 %disp(num2str((tml)));
13
14
15 ecgsig = (ecg.val)./200;
16 t = 1:length(ecgsig);
17
18
19 disp(num2str(length(ecgsig)));
20 disp(num2str(length(t)));
21
22 wt = modwt(ecgsig,1);
23 wtrec = zeros(size(wt));
24 wtrec(4:5,:) = wt(4:5,:);
25 y = imodwt(wtrec,'sym4');
26
27 y = abs(y).^2; avg = mean(y);
28 [qrspeaks,locs] = findpeaks(y,t,'MinPeakHeight',8*avg,...
29 'MinPeakDistance',50);
30 disp(strcat('R peak points = ',num2str(locs)));
31
32
33 figure
34 plot(t,y)
35 hold on
36 plot(locs,qrspeaks,'ro')
37 xlabel('Seconds')
38 title('R Peaks localized by wavelet Transform with Automatic Annotations')

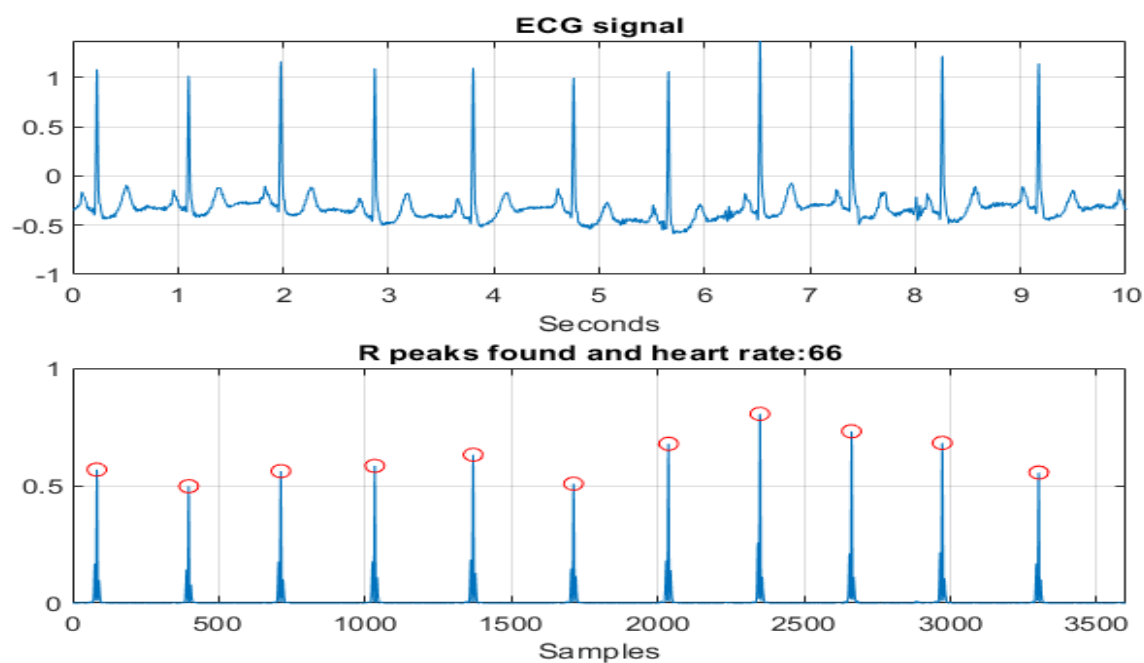
```

Output for 4 records

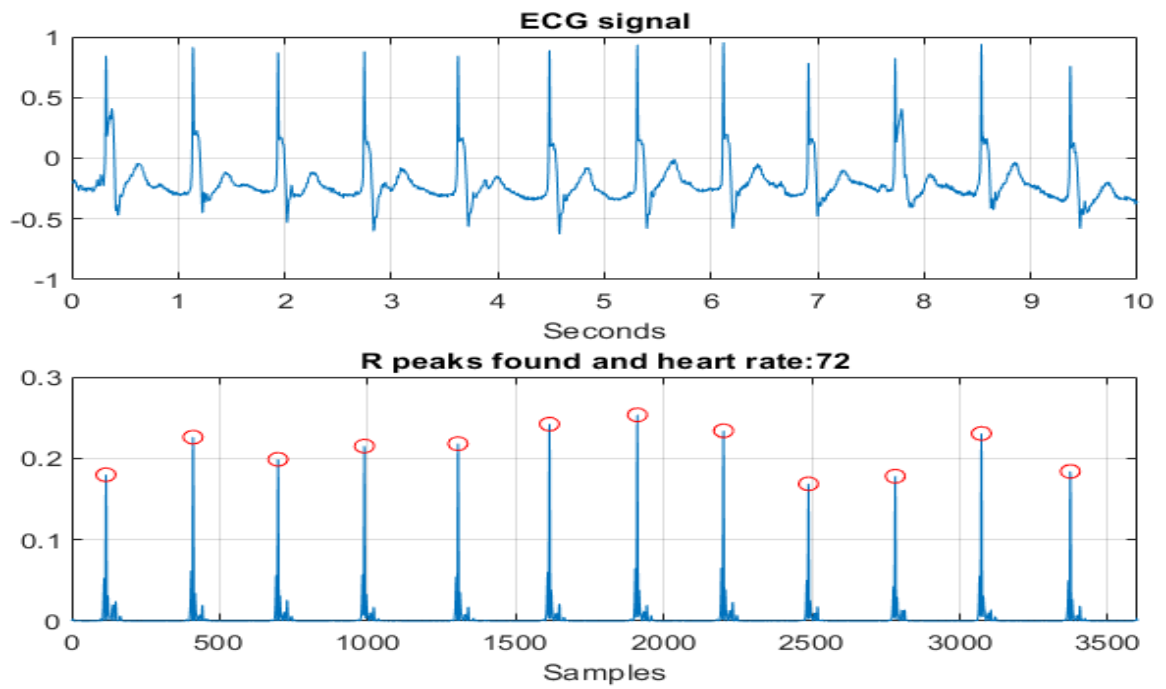
Record 100-



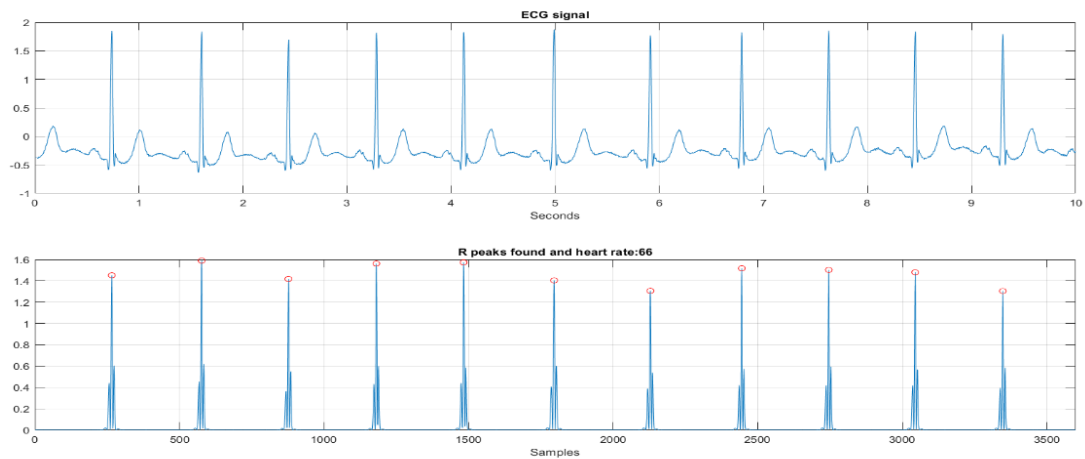
Record 101-



Record 102-



Record 103-



R peak detection:

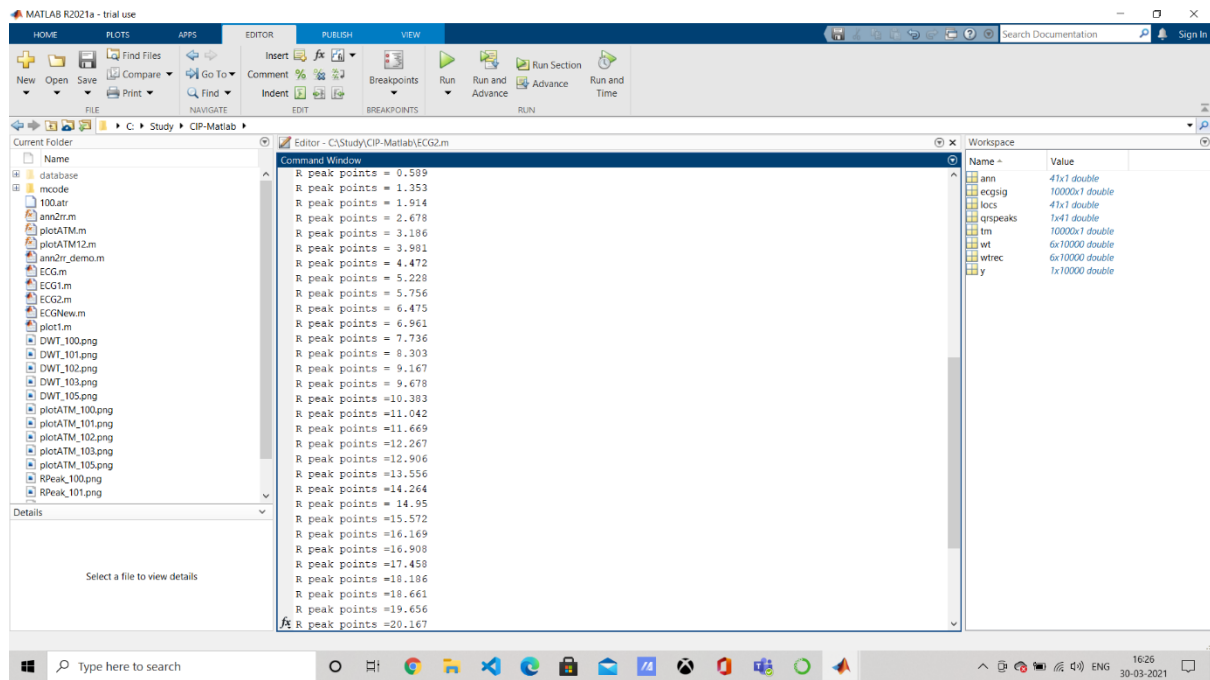
Code screenshot

```

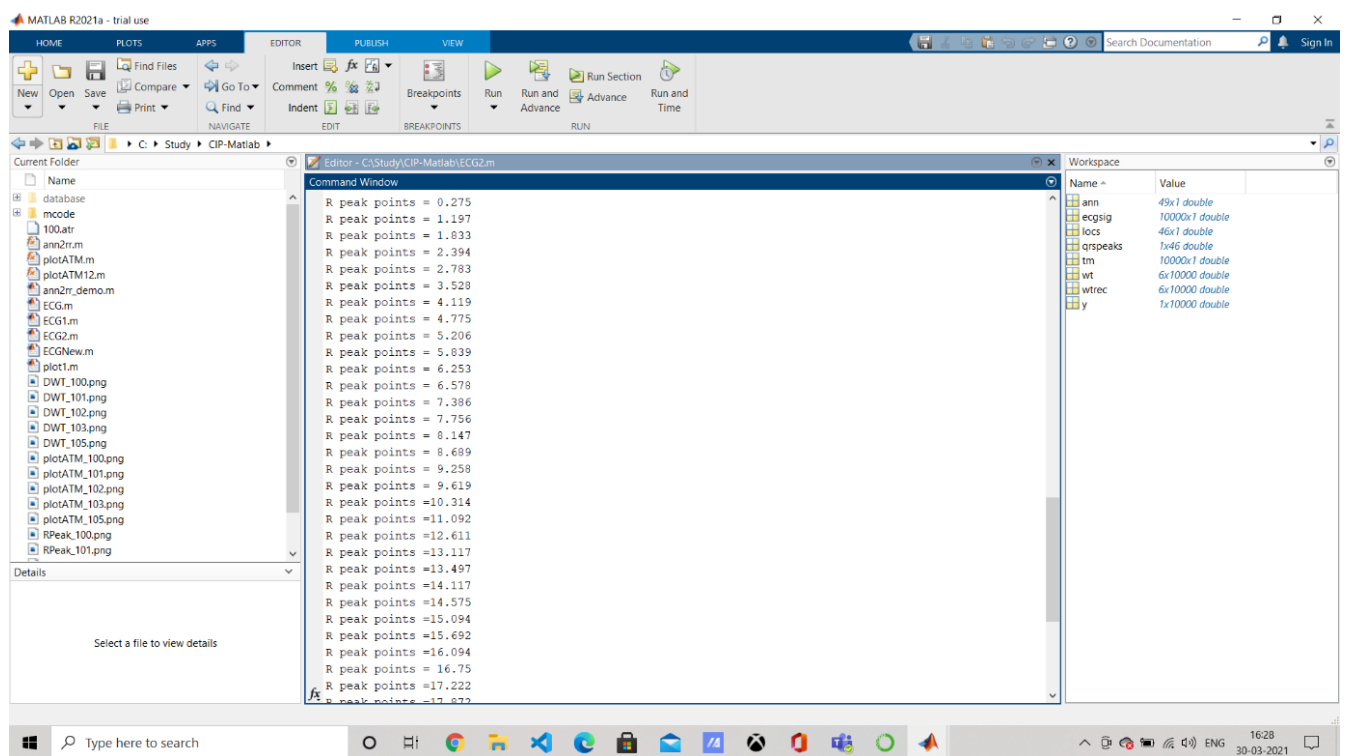
1 [signal,Fs,tm]=rdsamp('mitdb/109',1,10000);
2 [ann,type] = rdann('mitdb/109', 'atr',1,10000,[],'L');
3
4 disp(strcat('ANN-',num2str(ann/360)));
5
6
7 figure
8 disp(num2str(length(signal)));
9 disp(num2str(length(tm)));
10 plot(tm,signal)
11 hold on
12 plot(tm(ann),signal(ann),'ro')
13 xlabel('Seconds')
14
15 ylabel('Amplitude')
16 title('Subject - MIT-BIH 100')
17 wt = modwt(signal((19/360):(3587/360)),5);
18 wtrec = zeros(size(wt));
19 wtrec(4:5,:) = wt(4:5,:);
20 y = imodwt(wtrec,'sym4');
21 y = abs(y).^2;
22 [qrspeaks,locs] = findpeaks(y,tm,'MinPeakHeight',0.30,...
23 'MinPeakDistance',0.150);
24 [rrlocs] = zeros(34,1);
25 for f=2:length(locs)
26     rrlocs(f) = locs(f) - locs(f-1);
27
28 end
29 %disp(strcat('R location = ',num2str(locs)));
30 %disp(strcat('RR interval = ',num2str(rrlocs)));
31 figure
32 plot(tm,y)
33 hold on
34 plot(locs,qrspeaks,'ro')
35 xlabel('Seconds')
36 title('R Peaks Localized by Wavelet Transform with Automatic Annotations')

```


R peaks for record 200 in the dataset



R peaks for record 203 in the dataset



Module 3 - Detection of QRS complex:

Code Screenshot

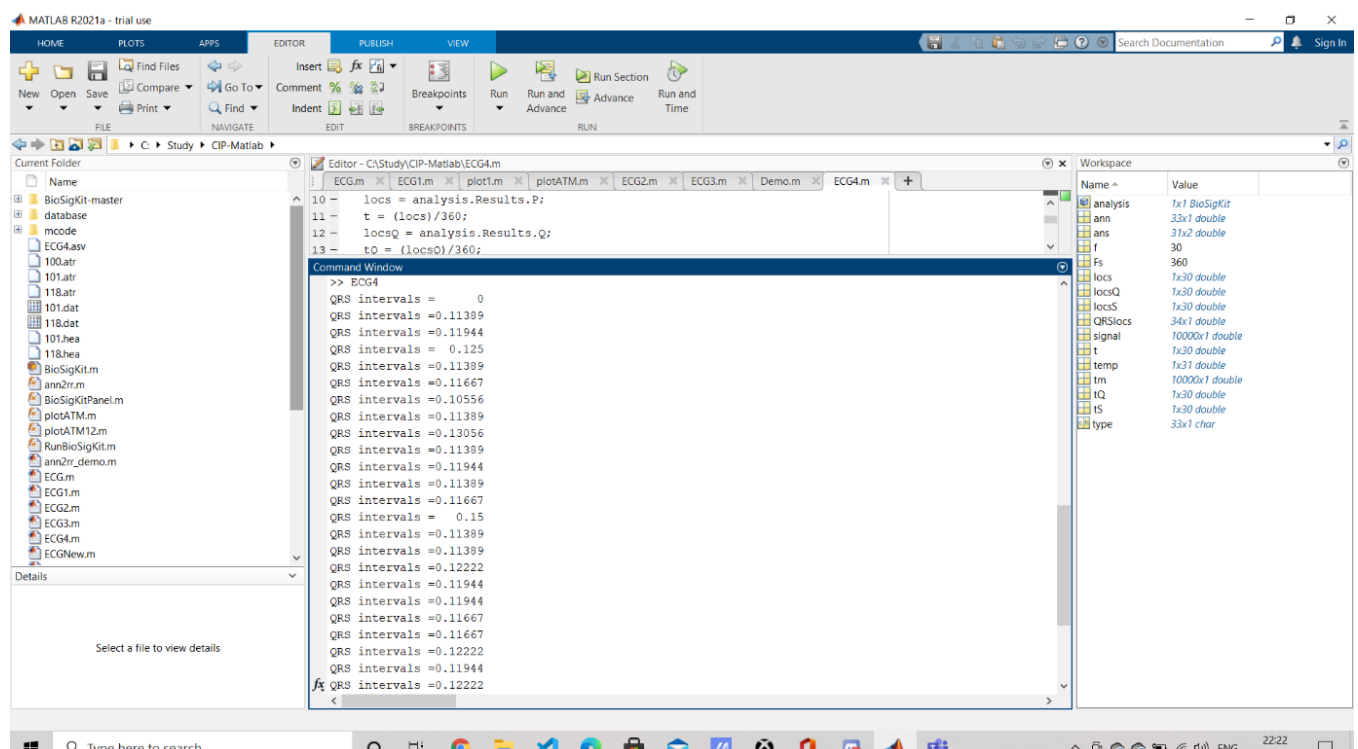
```

1  [signal,Fs,tm]=rdsamp('118',1,10000) ;
2  [ann,type] = rdann('118', 'atr',1,10000 ) ;
3  analysis = RunBioSigKit(signal,Fs,0);
4  analysis.MTEO_qrstAlg();
5
6  tR = analysis.Results.R/360;
7  locs = analysis.Results.P;
8  tP = (locs)/360;
9  locsQ = analysis.Results.Q;
10 tQ = (locsQ)/360;
11 locsS = analysis.Results.S;
12 tS = (locsS)/360;
13
14 [QRSlocs] = zeros(34,1);
15 for f=2:length(locs)
16     QRSlocs(f) = tS(f) - tQ(f);
17 end
18
19
20 disp(strcat('QRS intervals = ',num2str(QRSlocs)));

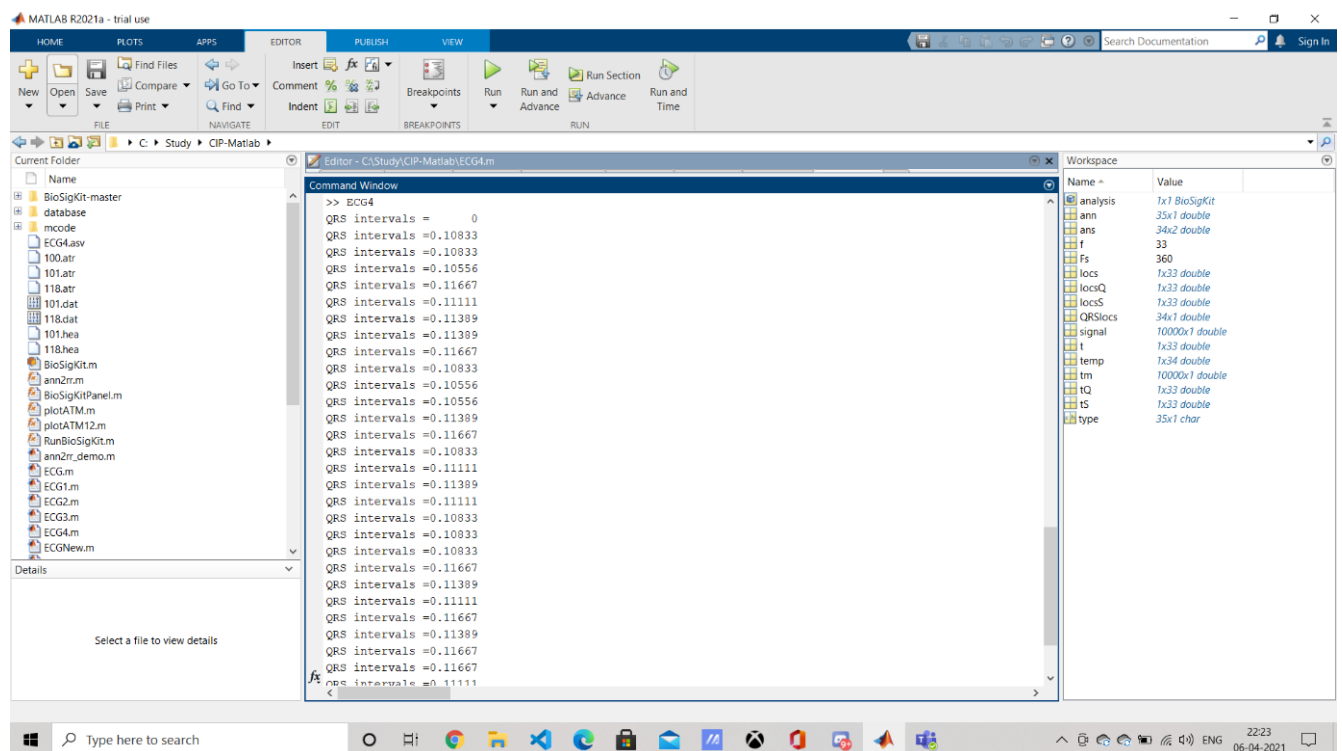
```

Sample Output

Record 101 :



Record 118:



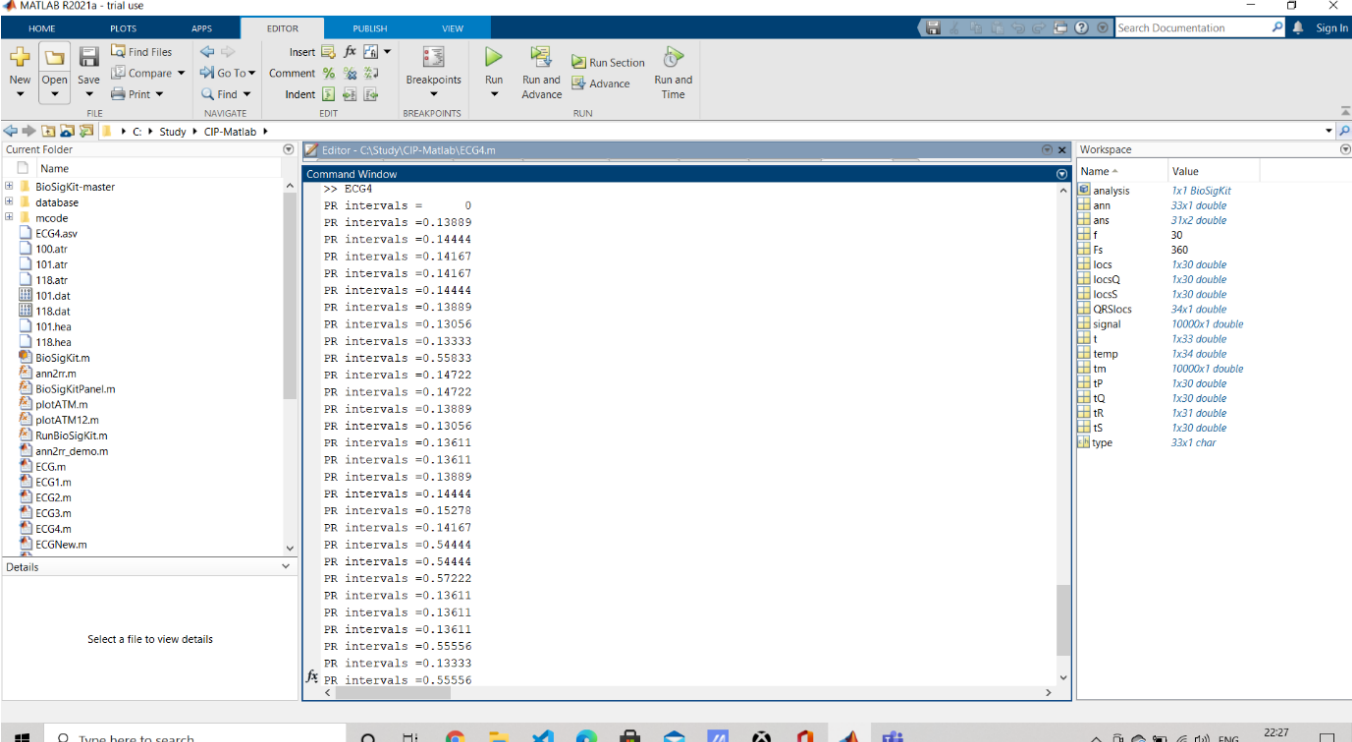
Module 4 - P wave detection and PR intervals:

Code Screenshot

```
1 [signal,Fs,tm]=rdsamp('mitdb/105',1,20000);
2
3 %[ann,type] = rdann('mitdb/118', 'atr',1,10000,[],'L');
4
5 %disp(strcat('ANN=',num2str(ann(1)/360)));
6 x = 13;
7 y = 3978;
8 analysis = RunBioSigKit(signal,Fs,0);
9 analysis.MTEO_qrstAlg();
10
11 timelimit = (y-x)/360;
12 tR = analysis.Results.R/360;
13 disp(strcat('Heart rate=',num2str(length(tR)*60/timelimit)));
14 locs = analysis.Results.P;
15 tP = (locs)/360;
16 locsQ = analysis.Results.Q;
17 tQ = (locsQ)/360;
18 locsS = analysis.Results.S;
19 ts = (locsS)/360;
20
21 [PRlocs] = zeros(length(tP),1);
22 for f=1:length(tP)
23     PRlocs(f) = tR(f) - tP(f);
24 end
25
26 %writematrix(PRlocs,'M.csv')
27
28 disp(strcat('R locs = ',num2str(tR)));
29 disp(strcat('R locs = ',num2str(tP)));
30
31 disp(strcat('PR intervals = ',num2str(PRlocs)));
```

Sample outputs:

Record 101:



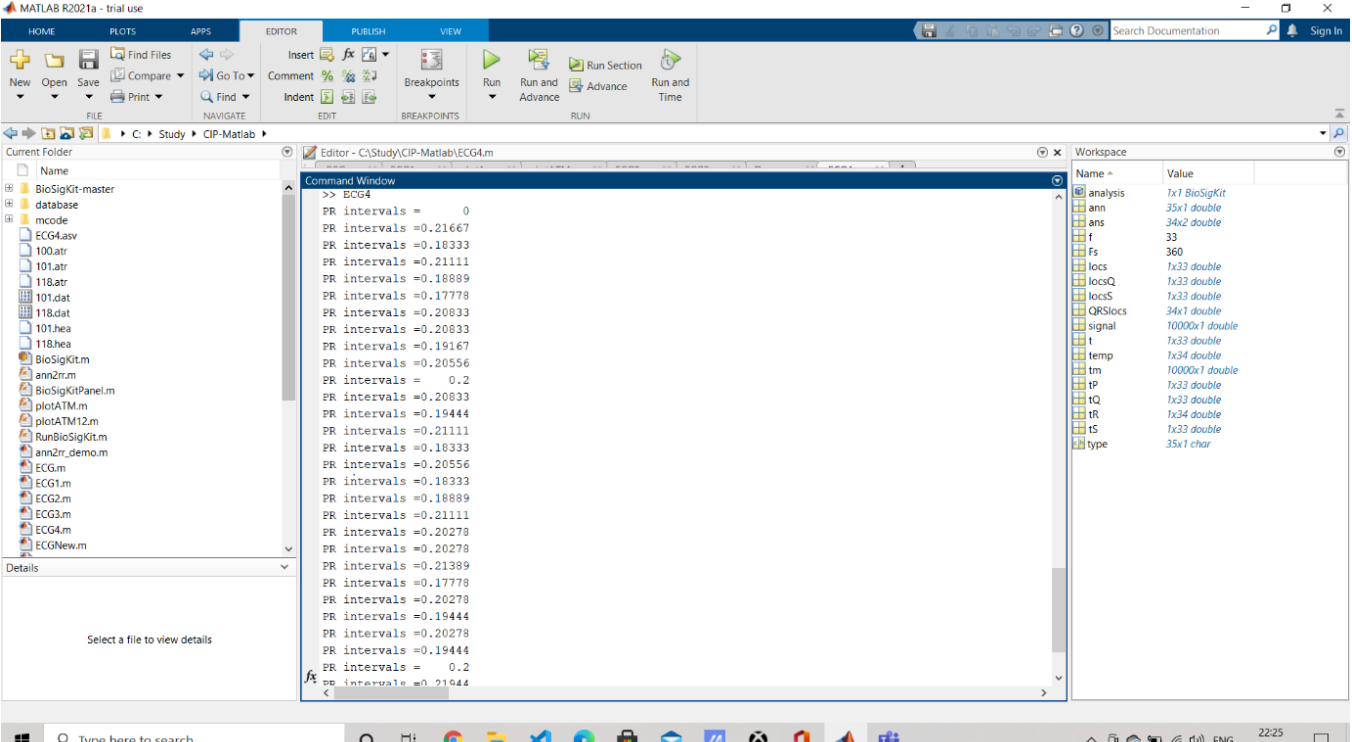
Command Window Output:

```
>> ECG4
PR intervals = 0
PR intervals = 0.13889
PR intervals = 0.14444
PR intervals = 0.14167
PR intervals = 0.14167
PR intervals = 0.14444
PR intervals = 0.13889
PR intervals = 0.13056
PR intervals = 0.13333
PR intervals = 0.55833
PR intervals = 0.14722
PR intervals = 0.14722
PR intervals = 0.13889
PR intervals = 0.13056
PR intervals = 0.13611
PR intervals = 0.13611
PR intervals = 0.13889
PR intervals = 0.14444
PR intervals = 0.15278
PR intervals = 0.14167
PR intervals = 0.54444
PR intervals = 0.54444
PR intervals = 0.57222
PR intervals = 0.13611
PR intervals = 0.13611
PR intervals = 0.13611
PR intervals = 0.55556
PR intervals = 0.13333
PR intervals = 0.55556
```

Workspace Variables:

Name	Value
analysis	1x1 BioSigKit
ann	33x1 double
ans	31x2 double
f	30
fs	360
locs	1x30 double
locsQ	1x30 double
locsS	1x30 double
QRSlocs	34x1 double
signal	10000x1 double
t	1x33 double
temp	1x34 double
tm	10000x1 double
tP	1x30 double
tQ	1x30 double
tR	1x31 double
tS	1x30 double
type	33x1 char

Record 118:



Command Window Output:

```
>> ECG4
PR intervals = 0
PR intervals = 0.21667
PR intervals = 0.18333
PR intervals = 0.21111
PR intervals = 0.18889
PR intervals = 0.17778
PR intervals = 0.20833
PR intervals = 0.20833
PR intervals = 0.19167
PR intervals = 0.20556
PR intervals = 0.2
PR intervals = 0.20833
PR intervals = 0.19444
PR intervals = 0.21111
PR intervals = 0.18333
PR intervals = 0.20556
PR intervals = 0.18333
PR intervals = 0.18889
PR intervals = 0.21111
PR intervals = 0.20278
PR intervals = 0.20278
PR intervals = 0.20278
PR intervals = 0.21389
PR intervals = 0.17778
PR intervals = 0.20278
PR intervals = 0.20278
PR intervals = 0.19444
PR intervals = 0.20278
PR intervals = 0.19444
PR intervals = 0.2
PR intervals = 0.21944
```

Workspace Variables:

Name	Value
analysis	1x1 BioSigKit
ann	35x1 double
ans	34x2 double
f	33
fs	360
locs	1x33 double
locsQ	1x33 double
locsS	1x33 double
QRSlocs	34x1 double
signal	10000x1 double
t	1x33 double
temp	1x34 double
tm	10000x1 double
tP	1x33 double
tQ	1x33 double
tR	1x34 double
tS	1x33 double
type	35x1 char

MODULE 5: CLASSIFIER MODELS

Logistic regression and Decision Tree Classifier :

```
In [24]: from sklearn import svm
         clf = svm.SVC()
         clf.fit(X_train, y_train)
         y_pred = clf.predict(X_test)
         confusion_matrix(y_test, y_pred)
         svm_accuracy = accuracy_score(y_test, y_pred)
```

```
In [43]: from sklearn.linear_model import LogisticRegression
         pcar = LogisticRegression(random_state=0).fit(X_train, y_train)
         y_pred = pcar.predict(X_test)
         confusion_matrix(y_test, y_pred)
         logistic_accuracy = accuracy_score(y_test, y_pred)
         logistic_accuracy
```

C:\Users\aadit\conda\envs\tf-gpu\lib\site-packages\sklearn\linear_model_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
<https://scikit-learn.org/stable/modules/preprocessing.html>
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(

Out[43]: 0.803030303030303

```
In [27]: from sklearn.tree import DecisionTreeClassifier
         clf = DecisionTreeClassifier(random_state=0).fit(X_train, y_train)
         y_pred = clf.predict(X_test)
         confusion_matrix(y_test, y_pred)
         decisionTree_accuracy = accuracy_score(y_test, y_pred)
         decisionTree_accuracy
```

Out[27]: 0.9545454545454546

Random Forest Classifier and Naïve Bayes Classifier:

```
In [28]: from sklearn.ensemble import RandomForestClassifier
         clf = RandomForestClassifier(max_depth=2, random_state=0).fit(X_train, y_train)
         y_pred = clf.predict(X_test)
         confusion_matrix(y_test, y_pred)
         RF_accuracy = accuracy_score(y_test, y_pred)
         RF_accuracy
```

Out[28]: 0.6818181818181818

```
In [29]: from sklearn.naive_bayes import GaussianNB
         y_pred = GaussianNB().fit(X_train, y_train).predict(X_test)
         confusion_matrix(y_test, y_pred)
         NB_accuracy = accuracy_score(y_test, y_pred)
         NB_accuracy
```

Out[29]: 0.9393939393939394

SVM and Neural Network Classifier:

```
In [35]: optimizer = tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07, amsgrad=False)
```

```
In [ ]: total_batch = int(len(y_train) / batch_size)
max_acc = 0
for epoch in range(epochs):
    avg_loss = 0
    for i in range(total_batch):
        batch_x, batch_y = get_batch(X_train, y_train, batch_size=batch_size)
        # create tensors
        batch_x = tf.Variable(batch_x)
        batch_y = tf.Variable(batch_y)
        # create a one hot vector
        batch_y = tf.one_hot(batch_y, 6)
        with tf.GradientTape() as tape:
            logits = nn_model(batch_x, W1, b1, W2, b2)
            loss = loss_fn(logits, batch_y)
            gradients = tape.gradient(loss, [W1, b1, W2, b2])
            optimizer.apply_gradients(zip(gradients, [W1, b1, W2, b2]))
        #optimizer.minimize(loss)
        avg_loss += loss / total_batch
    #optimizer.minimize(avg_loss)
    #test_logits = nn_model(X_test, W1, b1, W2, b2)
    test_logits = nn_model(X_test, W1, b1, W2, b2)

    max_idx = tf.argmax(test_logits, axis=1)
    #test_acc = np.sum(max_idx.numpy() == y_test) / len(y_test)

    test_acc = np.sum(max_idx.numpy() == y_test) / len(y_test)
    max_acc = max(max_acc, test_acc)
    if(epoch%100==0):
        print(f"Epoch: {epoch + 1}, loss={avg_loss:.3f}, test set accuracy={test_acc*100:.3f}%")

print(f"Epoch: {epoch + 1}, loss={avg_loss:.3f}, test set accuracy={test_acc*100:.3f}%")
print("\nTraining complete!")
print(max_idx.numpy())
print(y_test)

NN_accuracy = max_acc
```

```
In [30]: epochs = 3000
batch_size = 10
# convert x_test to tensor to pass through model (train data will be converted to
# tensors on the fly)
x_test = tf.Variable(X_test)
```

```
In [31]: def get_batch(x_data, y_data, batch_size):
        idxs = np.random.randint(0, len(y_data), batch_size)
        return x_data[idxs,:], y_data[idxs]
```

```
In [32]: W1 = tf.Variable(tf.random.normal([10, 300]), name='W1')
b1 = tf.Variable(tf.random.normal([300]), name='b1')
# and the weights connecting the hidden layer to the output layer
W2 = tf.Variable(tf.random.normal([300, 100]), name='W2')
b2 = tf.Variable(tf.random.normal([100]), name='b2')
W3 = tf.Variable(tf.random.normal([100, 50]), name='W3')
b3 = tf.Variable(tf.random.normal([50]), name='b3')
W4 = tf.Variable(tf.random.normal([50, 6]), name='W4')
b4 = tf.Variable(tf.random.normal([6]), name='b4')
```

```
In [33]: def nn_model(x_input, W1, b1, W2, b2):
        x_input = tf.reshape(x_input, (x_input.shape[0], -1))
        x = tf.add(tf.matmul(x_input, tf.cast(W1, tf.float32)), b1)
        x = tf.nn.tanh(x)
        x = tf.add(tf.matmul(x, tf.cast(W2, tf.float32)), b2)
        x = tf.nn.relu(x)
        x = tf.add(tf.matmul(x, tf.cast(W3, tf.float32)), b3)
        x = tf.nn.relu(x)
        logits = tf.add(tf.matmul(x, W4), b4)
        return logits
```

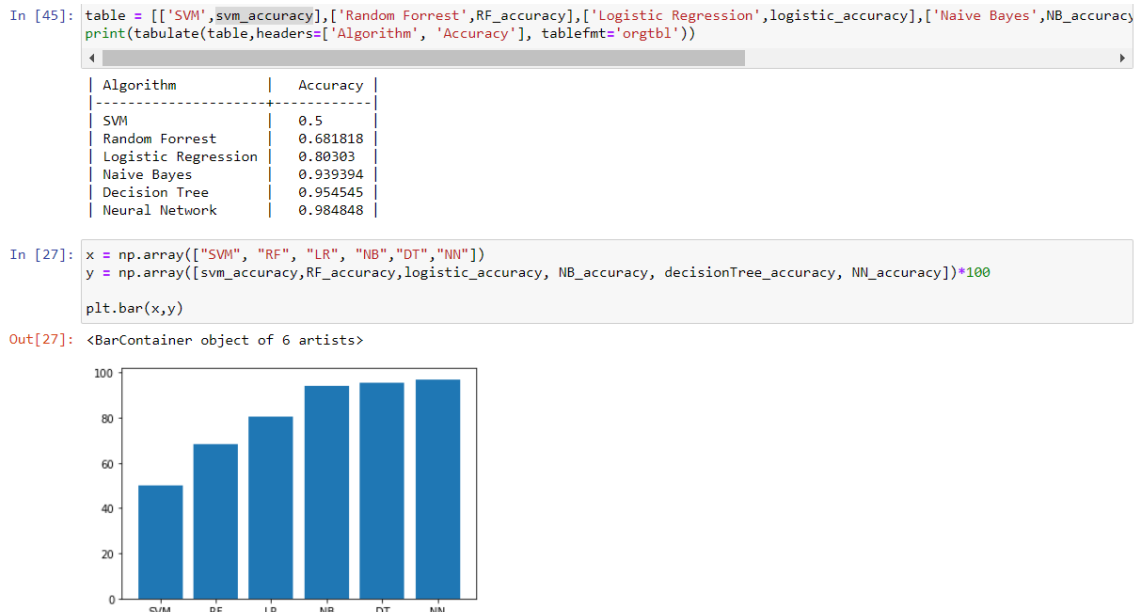
```
In [34]: def loss_fn(logits, labels):
        cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=labels,
                                                                              logits=logits))
        return cross_entropy
```

MODULE 6 : ACCURACY AND VALIDATION

```
In [45]: table = [['SVM', svm_accuracy], ['Random Forrest', RF_accuracy], ['Logistic Regression', logistic_accuracy], ['Naive Bayes', NB_accuracy]]
print(tabulate(table, headers=['Algorithm', 'Accuracy'], tablefmt='orgtbl'))
```

Algorithm	Accuracy
SVM	0.5
Random Forrest	0.681818
Logistic Regression	0.80303
Naive Bayes	0.939394
Decision Tree	0.954545
Neural Network	0.984848

MODULE 7 : Comparison chart



The abbreviations here are straightforward, NN is Neural Network, DT is decision tree and so on.. We can see that Decision tree and neural networks have high accuracy and that the neural network with 2 layers has slightly better accuracy than the decision tree algorithm. The deep learning model we have created seems to have fit the data better and also helps to vindicate us in our assumption of deep learning models producing higher accuracy.

Innovation:

1. Other papers/projects only identify whether the person has cardiac arrhythmia or not, but in this project, it not only does that but also classified into what type of arrhythmia.
2. Baseline wandering has been removed and hence we can work with multiple datasets irrespective of the distribution that the data is from.
3. The features we have chosen (PR intervals, QRS complex,RR interval) performs very well on the input dataset.

CONCLUSIONS:

In the clinical routine, computer aided diagnosis of heart arrhythmias can reduce the workload of cardiologists. As Machine Learning has evolved dramatically in recent years, it has reduced the workload of many things with the help of its algorithms.

In this project, a proper understanding of different abnormal beats in an ECG recording is simplified and those beats are graphically represented using a graph. The proposed System talks about different machine and deep learning algorithms which are used to measure the performance in terms of accuracy.

From the developed project we get an understanding that a shallow Neural Network provides a better performance (accuracy) as compared to the other machine learning algorithms represented . With the recent state-of-the-art performances of deep learning, biomedical scientists are coming one step closer to effective utilization of

deep learning techniques to be carried out to assist clinicians and patients alike in the near future.

References:

- [1] A.S. Adabag, G. Peterson, F.S. Apple, J. Titus, R. King, R.V. Luepker, Etiology of sudden death in the community: results of anatomic, metabolic, and genetic evaluation, *Am. Heart. J.* 159 (January (1)) (2010) 33–39.
- [2] J.J. Goldberger, A.E. Buxton, M. Cain, O. Costantini, D.V. Exner, B.P. Knight, D.Lloyd-Jones, A.H. Kadish, B. Lee, A. Moss, R. Myerburg, J. Olgin, R. Passman, D.Rosenbaum, W. Stevenson, W. Zareba, D.P. Zipes, Risk stratification for arrhythmic sudden cardiac death: identifying the roadblocks, *Circulation* 123(2011) 2423–2430.
- [3] E. Asensio, R. Narváez, J. Dorantes, J. Oseguera, T.A. Orea, R.P. Hernández, G.V.Rebollar, L. Mont, J. Brugada, Conceptos actuales sobre la muerte súbita, *Gac.Med. Mex.* 141 (March–April (2)) (2005) 89–98.
- [4] M. Velic, I. Padavic, S. Car, Computer aided ECG Analysis – State of the Art andUpcoming Challenges, 2013, arXiv:1306.5096v1 [cs.CV] 21 June.
- [5] M.E. Guevara-Valdivia, Monitoreo a distancia de los dispositivos automáticos implantables cardiovasculares (marcapasos, desfibriladores automáticos implantables y resincronizadores cardiacos), *Arch. Cardiol. Mex.* 79(July–September (3)) (2009) 221–225.
- [6] L. Gatzoulis, I. Iakovidis, Wearable and Portable eHealth Systems:Technological Issues and Opportunities for personalized care, *IEEE Eng. Med.Biol. Mag.* 26 (September-October (5)) (2007) 51–56
- [7]Albert Haque, Cardiac Dysrhythmia Detection with GPU-Accelerated Neural Networks.
- [8]Pranav Rajpurkar, Awni Y. Hannun,Masoumeh Haghpahani, Codie Bourn, Andrew Y. Ng,Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks.