Automatic Detection of Atrial Fibrillation Episodes

Data Science Project Report (Subject Code: 201400174)
Primary Topic: DM, Secondary Topic: DPV
Course: 2019/2020 – Quarter 2A – Group: 44 – Submission Date: 2020-Apr-19

Aishwarya Mala Gurusamy MuthuvelRabindran
University of Twente
a.m.gurusamymuthuvelrabindram@student.utwente.nl

Christopher Benjamin Leonard
University of Twente
c.benjaminleonard@studet.utwente.nl

ABSTRACT

This paper gives an outline of an attempt at automatically detecting Atrial Fibrillation (AF) episodes from the ECG data collected from the patient. Atrial Fibrillation is a condition of abnormal and non-uniform beating of the heart. It is caused due to irregularity in the beating of the atrial chambers of the heart at an abnormally increased pace. The algorithm proposed in this paper uses the length of the R-R intervals extracted from the ECG signal as input parameters and classifies it as AF or non-AF. Neural Networks (NN), Support Vector Machine (SVM) and Decision Tree algorithms were trained and used for classification. The performance of each of these algorithms was compared. Maximum accuracy of approximately 96% was obtained by using Neural Networks for classification. Dimensionality reduction on the feature set was attempted using the Gini impurity indexes obtained from the Decision tree algorithm. The feature set size was reduced significantly without much decrease in the accuracy.

KEYWORDS

Data Mining, Neural Networks, Support Vector Machine, Decision Tree, Automatic AF episode detection using R-R irregularity

1 INTRODUCTION

Atrial Fibrillation is a form of arrhythmia – irregularity in heart beat due to cardiac conditions. AF, as its name suggests, is caused due to disturbances in the atrial (upper) chambers of the heart [1].

Figure 1 below shows the different parts of the ECG wave of a normal heart beat [3]. The wave in an Electrocardiogram (ECG) is a graph that represents the electrical activity of the heart. The beating of the heart is caused due to an electrical impulse that travels through it and squeezes the muscles to pump blood. An ECG is nothing but a plot of the voltage of this electric impulse in the y-axis and the time in the x-axis. The ECG wave of a typical heart beat consists of three main components – P wave, QRS complex and T wave. The first "P wave" is created when the impulse travels through the atria, the upper chambers, of the heart. The second part, the "QRS complex" is created by the impulse travelling through the ventricles, the lower chambers, of the heart. The third part, the "T wave" is caused by the electrical recovery of the ventricular chambers to return to their resting state [4].

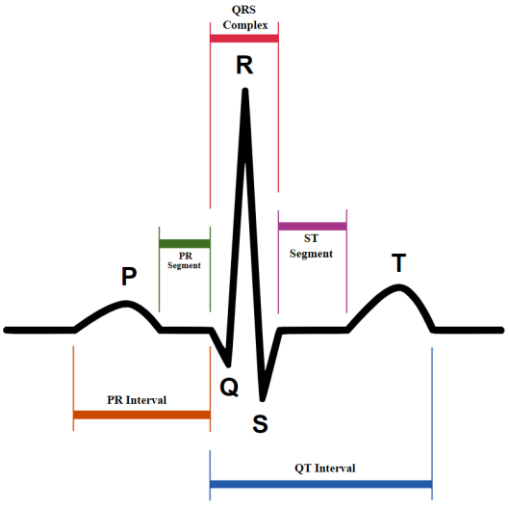
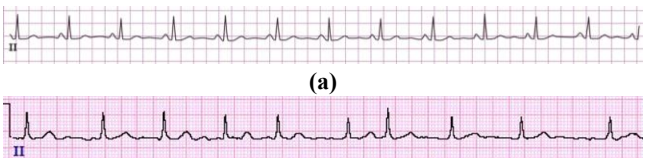


Figure 1 ECG wave of a normal heart beat

A normal episode of AF is generally identified by the absence of the P wave or by the unevenness and higher frequency of the ventricular rate for a period of at least 30 seconds. Thus, an AF episode could be identified from an ECG wave either by analyzing the Atrial activity, that is the P wave or by identifying irregularity in the ventricular rate by measuring the time interval between two consecutive R peaks – the R-R interval. The R peak is the most prominent characteristic within the ECG, making it relatively simple to detect. Therefore, algorithms that detect AF based on RR intervals are the most common [5].



(a)

(b)

Figure 2 A comparison between a normal ECG wave (a) and an ECG wave recorded during AF episode (b)

[Image Credits: https://geekymedics.com/understanding-an-ecg/ and www.healio.com/cardiology/learn-the-heart/cardiology-review/topic-reviews/atrial-fibrillation]

The work represented in this report also uses the R-R interval irregularity deduced from a patient's ECG record to automatically detect an AF episode in the patient. The algorithm uses ECG signals sampled using a window length of 30 seconds. The R-R intervals in the ECG samples are extracted and are used as input parameters to train classification algorithms to automatically detect AF episodes. The classification algorithms used are Neural Networks (NN), Support Vector Machine (SVM) and Decision tree. The performance of each of these algorithms on AF detection was compared. Further, feature set optimization was done using the Gini impurity indexes obtained from the Decision tree algorithm. The feature set size was reduced significantly without much decrease in the accuracy. The following sections of the report outline the algorithm and the results that were obtained.

2 RELATED WORKS

The aim of the project is to answer the question 'To what extent can one automatically detect episodes of AF'. From the literature, we were able to find AF detection algorithms based on R-R intervals that used different classification methods.

Moody et al. [6] used Markov models to train and detect AF episodes. They collaborated with the Beth Israel Hospital of Boston to create MIT-BIH Arrhythmia database that consisted of 12 recordings of each half-hour long. 2 of the recordings contained AF rhythms, 5 of the recordings contained a mix of both AF and normal rhythms and the remaining 5 were a variety of rhythms that were aimed to confuse the detector and help improve performance. They classified each of the R-R intervals as 'Short' (S), 'Regular' (R) and 'Long' (L) by comparing its length with the mean R-R interval. From their database, they calculated the emission probabilities for the states S, R and L and further the transition probabilities as per equation (1) below. In the equation, $R \in \{AF, other\}$, t_i & $t_j \in \{S, R, L\}$ and t_i is the current state and t_j is the previous state. $P_{i,j,R}$ gives the probability of the state t_i occurring given the previous state as t_j and class as R.

$$P_{i,j,R} = P(t_i \mid t_j, R) \quad (1)$$

Using this the conditional probability of the entire observation $T = \{t_1, t_2, \ldots, t_n\}$ given the classes 'AF' and 'other' are computed as per the below equation (2).

$$P(T \mid R) = \prod_{i=1}^{n} P_{i,\, i-1,\, R} \quad (2)$$

For a given observation T, the probabilities P (T | AF) and P(T | other) are calculated. The observation is classified as belonging to the class that had greater probability. Using this model along with first-order filtering and interpolation, they were able to achieve AF sensitivity of 96% and positive predictivity of 87%.

Logan et al. [7] used the variance of R-R intervals as a parameter to detect AF episodes. AF episodes are characterized by a significant change in the frequency of the ventricular rate and this, in turn, induces a detectable variance in the R-R interval. In addition to that, the R peak is the most prominent feature in an ECG and is least disturbed by noise and improper lead placements during the ECG recording. This makes it a very reliable parameter to detect AF episodes. Using these facts they created a very simple model. They obtained the R-R intervals and normalized it. They computed the variance of the normalized R-R intervals over a 10 second sliding window. They set a threshold on the variance to classify the current sample as AF or non-AF. They use a simple majority voting scheme over a 600 heart beats window to classify it as AF or non-AF. They claim this to be a very simple, reliable and morphology-independent algorithm to detect AF. On testing this algorithm on the MIT-BIH database, they were able to show a sensitivity 0.96 and specificity 0.89.

Tatento et al. [8] used RR intervals and ΔRR – the difference between two consecutive RR intervals to design an algorithm for the detection of AF episodes. They calculated the coefficients of variation (CV is the ratio of the standard deviation to the mean) of the RR intervals and ΔRR extracted from the test data and compared it with the standard CV values obtained from the training data and used this as a similarity score to classify the test data as AF or non-AF. Another approach was to use the Kolmogorov-Smirnov test to compare and obtain the similarity between the standard density histograms of the RR intervals and ΔRR and the density histograms of RR intervals and ΔRR obtained from the test data. On testing their algorithm, they were able to get the best performance on using the Kolmogorov-Smirnov test on the ΔRR intervals - a sensitivity of 94.4% and a specificity of 97.2%.

Mukherjee et al. [9] have proposed an algorithm that directly uses the ECG recordings for the purpose of detecting AF episodes. They create a two-layered framework for classification. The first layer comprises of three binary classifiers and the second layer from the output of the first layer as input gives the final classification output. PQRST points in the ECG recording are marked and is preprocessed to remove noise. This preprocessed ECG signal is used to extract the time, frequency, morphological and statistical features. Suitable feature subset is selected through feature ranking and dimensionality reduction processes. The selected features are fed into the binary classifiers for the classification task. This algorithm was able to perform consistently well even on hidden test data.

These literature works served as references to structure our approach to the problem statement. Further in this report, we explain the various aspects of our algorithm - dataset creation, the classification algorithms used and the results obtained, the feature dimensionality reduction and its results.

3 DATASET

The dataset provided with the project is the ECG data collected from the electrophysiology department of Erasmus Medical Centre in Rotterdam. All these data were obtained from patients within 10 days of Coronary artery bypass surgery because patients after cardiac surgery tend to have AF frequently. Each ECG from a patient is a 12 lead ECG where information is gathered from 12 different areas of the heart. Since R peaks are the most prominent

features in an ECG wave, the R-R interval was chosen as a parameter rather than using the P wave to detect AF episodes.

Each ECG recording was semi-automatically analyzed and R peaks were annotated, which was later manually verified by a physician who also labelled other irregularities. Each of these annotated ECG data was collected and stored in a text file as raw data with 3 columns - timestamp, R-R peak interval (ms) and type of heartbeat. This raw data was then preprocessed. As a first step of preprocessing, the ECG recordings were split into samples of length 30 seconds. Data cleaning was also a part of the preprocessing steps. The ECG samples with insufficient information or with information that cannot be used for analyzing were removed. All the R-R intervals that were obtained from each the ECG samples were placed into one of the 30 bins of each 50ms size and in the range 200 to 1700ms. This gives the frequency of R-R intervals present in a particular bin. The frequency obtained is normalized, shuffled and stored. The preprocessed data consists of 150,000 30 sec ECG records. As mentioned in preprocessing steps, R-R intervals were normalized and stored in the first 30 columns of a 31 column excel sheet. The 31st column is filled with either 1 or 0 where 1 represents AF and 0 represents non-AF as reported by a physician. Amongst 150000 records, 36537 are categorized as AF and 113463 are categorized as non-AF.

Once all the steps of preprocessing have been completed, data are classified into training, testing and validation datasets. Since the ratio of AF to non-AF is large it gives rise to an imbalanced dataset. To cope up with an imbalanced data set, we randomly chose 73200 records of non-AF. This maintained the ratio between the number of AF and non-AF records as 0.5 and balances the dataset.

Finally, 80% of the dataset was declared as training dataset and the remaining 20% was split equally and declared as validation and test dataset. Training data contained 87789 records and validation and testing data each contained 10974 records.

4 CLASSIFICATION

As classification algorithms, Neural networks, Support Vector Machine and Decision Tree were chosen. The normalized frequency of the R-R intervals (extracted from the 30 seconds long ECG sample) in 30 bins each of size 50 milliseconds and in the range 200 to 1700 milliseconds is taken as input for the classifiers. The output of these classifiers will be the class to which the input 30 second ECG sample belongs – AF or non-AF. We used Machine Learning models from Scikit-learn Python library for training and testing of all the classifiers [2]. In order to find the optimum training parameters of each algorithm, models were trained using the training data and then tested on the validation data and the model that had the best performance on the validation data was chosen.

4.1 Neural Networks (NN)

As parameters for the training of the Neural network model, we had to choose the best activation function, the optimum number of hidden layers and the number of neuron in each hidden layer. For this, we used the validation dataset.

The model provides ‘identity’, ‘logistic’, ‘relu’ and ‘tanh’ as activation functions for the hidden layers. On testing each of the activation function on our validation data, we obtained the results shown in figure 3. The model performed the best on using ‘relu’ (rectified linear unit function, returns f(x) = max(0, x)) as the activation function.

```
Activation Function: logistic
Error on Training data: 0.039367118887332124
Error on Validation data: 0.04182613449972663
Activation Function: tanh
Error on Training data: 0.03763569467700965
Error on Validation data: 0.041188263167486784
Activation Function: relu
Error on Training data: 0.035277768285320484
Error on Validation data: 0.040277018407144156
Lowest error is 0.040277018407144156 occurs at relu.
```

Figure 3 Performance of different activation functions

With RELU as activation function, we ran an algorithm to find the optimum number of neurons in the hidden layer. The results obtained are shown in figure 4. Although the least error rate occurs at hidden layer size 80, we choose 50 as the optimum hidden layer size as the decrease in error rate is insignificant when compared to the amount of complexity an additional 30 neurons will add to the model.

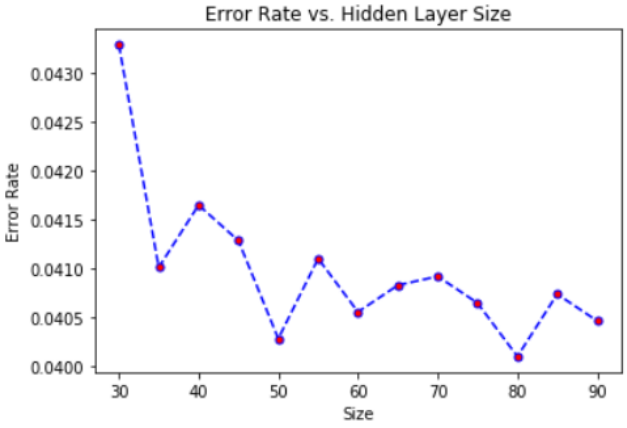


Figure 4 Choosing the optimum hidden layer size

The addition of hidden layers did not cause any significant decrease the error on the validation data and thus we stopped with one hidden layer. We let the weight optimization algorithm to be the default ‘adam’ – a Stochastic gradient-based optimizer. We set the maximum number of iterations as 20,000 and learning rate as 0.001 (default).

4.2 Support Vector Machine (SVM)

The SVM model provided by Sklearn provides 'linear', 'poly', 'rbf' and 'sigmoid' as kernels. We needed to choose a kernel that suits our dataset. The models trained using each of the kernels were tested against the validation data and the results are given in figure 5. The highest accuracy on the validation data was obtained while using the RBF (Radial Basis Function) kernel. We let the other parameters with their default values.

```
Kernel: rbf
Training data accuracy: 0.948376220255385
Validation data accuracy:  0.946965554948059

Kernel: poly
Training data accuracy: 0.6670539589242388
Validation data accuracy:  0.6724075086568252

Kernel: linear
Training data accuracy: 0.933750242057661
Validation data accuracy:  0.9360306178239475

Kernel: sigmoid
Training data accuracy: 0.9204797867614394
Validation data accuracy:  0.9243666848915618

Highest accuracy is 0.946965554948059 occurs at rbf kernel.
```

Figure 5 Choosing a suitable kernel for SVM training

4.3 Decision Tree

We used a decision tree based on the Gini criterion. We needed to choose the optimum depth of the tree. For this, we trained trees of different depths and calculated their performances on the validation data. The tree with the best performance was chosen as the final decision tree model. Figure 6 shows the results obtained for decision trees of different depths. The maximum accuracy on the validation data was obtained at a depth of 14. Other parameters were left with their default values during training.

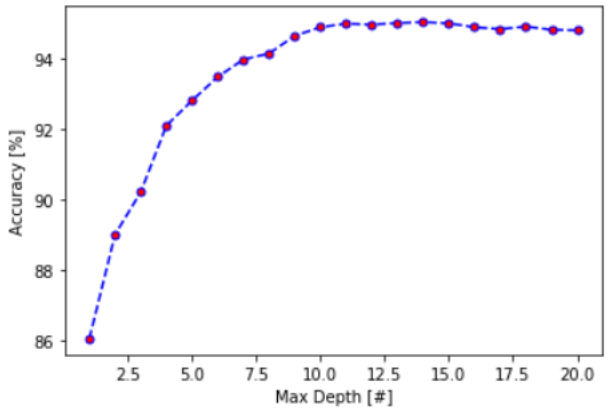


Figure 6 Accuracy vs Max tree depth

5 FEATURE DIMENSIONALITY REDUCTION

From the decision tree model, the Gini index of each parameter could be computed. These were used as a measure of the importance of the input feature. Figure 7 shows the input parameters sorted according to their Gini indexes.

```
Features in descending order of Gini index:
[ 3  2  4  1 12  7  6 11  5  8  9 10 13 15 14 16 17  0 18 24 20 19 25 22
 21 27 23 26 28 29]

Corresponding Gini Indexes:
[5.60562990e-01 1.31988349e-01 8.40688008e-02 4.13101487e-02
 2.90557705e-02 2.30806569e-02 2.00948426e-02 1.97566098e-02
 1.89815251e-02 1.66882620e-02 1.35305537e-02 1.25150250e-02
 7.30298969e-03 5.63136661e-03 5.00342946e-03 3.78622536e-03
 2.43634325e-03 2.20413958e-03 4.15275822e-04 4.06370153e-04
 3.17032206e-04 2.14041920e-04 1.74018743e-04 1.32756913e-04
 1.29470012e-04 9.99480346e-05 5.80335851e-05 5.50244362e-05
 0.00000000e+00 0.00000000e+00]
```

Figure 7 Gini indexes of each of the input features arranged in descending order

For each feature, the amount of variance of the data represented by it could be calculated using equation (3) where n is the total number of features, m is the feature for which the variance is calculated and G_i is the Gini index corresponding to the feature i. The plot in figure 8 shows the cumulative variance value versus the number of featured that were considered.

$$V(m) = \frac{Gm}{\sum_{i=1}^{n} Gi} \quad (3)$$

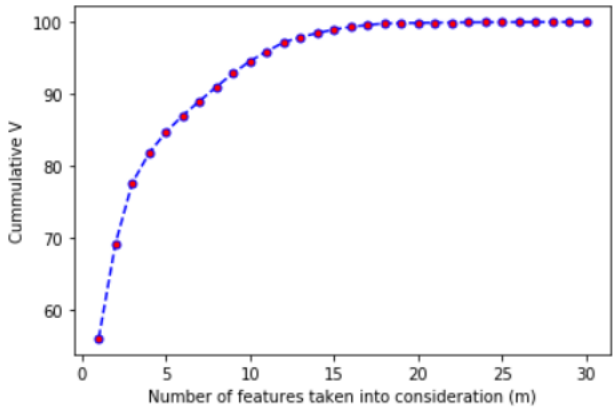


Figure 8 Cumulative variance of the data represented by the features in consideration vs the number of features in consideration

For feature set optimization, we trained and measured the model performance by using the features that represent 99%, 95% and 90% of the data variance. 99% of the variance of the data was found to represented by the top 16 features. 95% of the data variance was found to be represented by the top 11 features and 90% of the data variance was found to be represented by the top 8 features. Decision tree models were trained using the reduced feature sets. The performance of these models are discussed in the results section.

6 RESULTS

Table 1 shows the accuracy, precision and recall values against the test data on using the final models of NN, SVM and Decision tree on it.

Table 1 Performance of the classifiers

	Accuracy	AF		Non-AF	
		Precision	Recall	Precision	Recall
Neural Networks	0.96	0.94	0.94	0.96	0.97
SVM	0.95	0.94	0.91	0.96	0.97
Decision Tree	0.95	0.93	0.92	0.96	0.97

Table 2 shows the performance of the decision tree models that were trained using the reduced feature set.

Table 2 Performance of the decision tree classifiers that were trained using reduced feature sets

Feature set size	Accuracy	AF		Non-AF		Depth of the tree
		Precision	Recall	Precision	Recall	
99% [top16]	0.95	0.93	0.91	0.96	0.97	14
95% [top11]	0.945	0.92	0.92	0.96	0.96	12
90% [top 8]	0.94	0.91	0.91	0.95	0.96	11

The confusion matrices of all the models could be found in the appendix for reference.

7 CONCLUSION AND DISCUSSION

From table 1, we could tell that the performance of all the classifiers are almost similar but Neural Networks has a slight edge over the others. On taking into consideration the training time, decision trees were the most optimum. They significantly look very less time for training when compared to Neural networks and SVM but yield almost similar performance.

From table 2, we could conclude that we were able to significantly reduce the number of features with almost insignificant loss in the performance of the model. Out of the initial 30 features, using only the most important 8 features for classification still maintained the accuracy at 94%.

To answer the question ‘To what extent can one automatically detect episodes of AF’, using the data provided we were able to detect episodes of AF but real-time detection could be tricky. We would need to experiment in suitable environments before we could decide on whether this solution could be applied on a real-time basis with the required amount of accuracy. As it is in the case with the detection of any medical condition, during real-time AF detection, the number of False Negatives should be reduced to the maximum extent possible. In this work, we have not touched upon this aspect but we believe the ROC curve of the classifiers could help.

Apart from using classifiers and Machine learning algorithms for detecting AF episodes from RR intervals, we could also explore the effect and accuracy of the statistical methods. As explained in the related works section, there are works of literature available that use the variance in the RR interval to detect AF episodes. There are also algorithms that use CV test and Kolmogorov-Smirnov test to detect AF episodes from RR intervals. These methods are very much simpler and also do not involve complex computations. Our work and this report could be extended to evaluate the performance of these methods and experiment on the possibility of combining the two approaches for improved robust performance.

APPENDIX

Performance of Neural Network model:

```
nn_model = neural_network.MLPClassifier(solver='adam', alpha=1e-5,hidden_layer_sizes=50,activation='relu',
                                        learning_rate_init=0.001,max_iter=20000,shuffle=True)
classify(nn_model,train_d,train_y,test_d,test_y)
```

```
Confusion Matrix:
Predicted\Actual     AF    non-AF
AF                 3491      205
non-AF              222     7056
Accuracy: 0.9610898487333698
Error Rate: 0.038910151266630215

Prediction of AF
-------------------
Precision: 0.9445346320346321
Recall: 0.9402100727174791

Prediction of Non-AF
--------------------
Precision: 0.9694971145919209
Recall: 0.9717669742459716
```

Performance of SVM model:

```
svm_model = svm.SVC(kernel='rbf')
classify(svm_model,train_d,train_y,test_d,test_y)
```

```
Confusion Matrix:
Predicted\Actual     AF    non-AF
AF                 3391      228
non-AF              322     7033
Accuracy: 0.9498815381811555
Error Rate: 0.050118461818844544

Prediction of AF
-------------------
Precision: 0.9369991710417243
Recall: 0.9132776730406679

Prediction of Non-AF
--------------------
Precision: 0.9562202583276682
Recall: 0.9685993664784465
```

Performance of Decision tree model:

```
dm_model = DecisionTreeClassifier(max_depth=14)
classify(dm_model,train_d,train_y,test_d,test_y)
```

```
Confusion Matrix:
Predicted\Actual     AF    non-AF
AF                 3411      238
non-AF              302     7023
Accuracy: 0.950792782941498
Error Rate: 0.049207217058501916

Prediction of AF
-------------------
Precision: 0.9347766511372979
Recall: 0.9186641529760302

Prediction of Non-AF
--------------------
Precision: 0.9587713310580205
Recall: 0.9672221457099573
```

Performance of reduced feature set with the top 16 features:

```
dm_model = DecisionTreeClassifier(max_depth=14)
classify(dm_model,train_tf,train_y,test_tf,test_y)
```

```
Confusion Matrix:
Predicted\Actual     AF    non-AF
AF                 3401      239
non-AF              312     7022
Accuracy: 0.9497904137051212
Error Rate: 0.0502095862948788

Prediction of AF
-------------------
Precision: 0.9343406593406594
Recall: 0.9159709130083491

Prediction of Non-AF
--------------------
Precision: 0.9574584128715571
Recall: 0.9670844236331084
```

Performance of reduced feature set with the top 11 features:

```
dm_model = DecisionTreeClassifier(max_depth=12)
classify(dm_model,train_tf1,train_y,test_tf1,test_y)
```

```
Confusion Matrix:
Predicted\Actual     AF    non-AF
AF                 3403      291
non-AF              310     6970
Accuracy: 0.945234189903408
Error Rate: 0.05476581009659195

Prediction of AF
-------------------
Precision: 0.92122360584732
Recall: 0.9165095610018853

Prediction of Non-AF
--------------------
Precision: 0.9574175824175825
Recall: 0.9599228756369647
```

Performance of reduced feature set with the top 8 features:

```
dm_model = DecisionTreeClassifier(max_depth=11)
classify(dm_model,train_tf2,train_y,test_tf2,test_y)
```

```
Confusion Matrix:
Predicted\Actual     AF    non-AF
AF                 3367      316
non-AF              346     6945
Accuracy: 0.939675596865318
Error Rate: 0.060324403134681977

Prediction of AF
-------------------
Precision: 0.9142003801248981
Recall: 0.9068138971182332

Prediction of Non-AF
--------------------
Precision: 0.9525442326155534
Recall: 0.9564798237157416
```

The python code used is saved as an ipynb file in the below link.
https://drive.google.com/open?id=11pdVBWDh-HynW26TFFA39wFbkfS4Y_v2

REFERENCES

[1] Wikipedia contributors, "Atrial fibrillation," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Atrial_fibrillation&oldid=951134567

[2] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011

[3] Wikipedia contributors, "Electrocardiography," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Electrocardiography&oldid=950826800

[4] American Heart Association, "Electrocardiogram", https://www.heart.org/en/health-topics/heart-attack/diagnosing-a-heart-attack/electrocardiogram-ecg-or-ekg

[5] N. Larburu, T. Lopetegi, and I. Romero. Comparative study of algorithms for atrial fibrillation detection. In 2011 Computing in Cardiology, pages 265–268. IEEE, 2011

[6] Moody GB, Mark RG. A new method for detecting atrial fibrillation using R-R intervals. Computers in Cardiology 1983; 10:227-230

[7] Logan B, Healey J. Robust Detection of Atrial Fibrillation for a Long Term Telemonitoring System. Computers in Cardiology 2005; 32: 619-622

[8] Tatento K, Glass L. Automatic detection of atrial fibrillation using the coefficient of variation and density histograms of RR and ΔRR intervals. Medical & Biological Engineering & Computing 2001; 39: 664-671

[9] Mukherjee, A., Choudhury, A.D., Datta, S., Puri, C., Banerjee, R., Singh, R., Ukil, A., Bandyopadhyay, S., Pal, A. and Khandelwal, S., 2019. Detection of atrial fibrillation and other abnormal rhythms from ECG using a multi-layer classifier architecture. *Physiological measurement*, *40*(5), p.054006.