A Project Report On
Arrhythmia classification with extracted features from ECG signal for arrhythmia detection

A project report submitted in fulfillment for the Diploma Degree in AI & ML Under Applied Roots with University of Hyderabad

Project Submitted By

Kukkala Balaiah - Roll No.: 40AIML236-21/2

Under the kind guidance of

Mentor: Sajit Sasidharan

Approved by: Sajit Sasidharan

University of Hyderabad

October, 2022

Declaration of Authorship

We hereby declare that this thesis titled “Arrhythmia classification with extracted features from ECG signal for arrhythmia detection” and the work presented by the undersigned candidate, as part of Diploma Degree in AI & ML.
All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: Kukkala Balaiah
Thesis Title: Arrhythmia classification with extracted features from ECG signal for arrhythmia detection

CERTIFICATE OF RECOMMENDATION

We hereby recommend that the thesis entitled "Arrhythmia classification with extracted features from ECG signal for arrhythmia detection" prepared under my supervision and guidance by Kukkala Balaiah be accepted in fulfilment of the requirement for awarding the degree of Diploma in AI & ML Under applied roots with University of Hyderabad. The project, in our opinion, is worthy for its acceptance.

Mentor: Sajit Sasidharan

Under Applied AI Roots with

University of Hyderabad

ACKNOWLEDGEMENT

Every project big or small is successful largely due to the effort of a number of wonderful people who have always given their valuable advice or lent a helping hand. I sincerely appreciate the inspiration; support and guidance of all those people who have been instrumental in making this project a success. I, Kukkala Balaiah student of applied roots, is extremely grateful to mentors for the confidence bestowed in me and entrusting my project entitled "Arrhythmia classification with extracted features from ECG signal for arrhythmia detection" with special reference.
At this juncture, I express my sincere thanks to Mentor: Sajit Sasidharan of applied roots for making the resources available at right time and providing valuable insights leading to the successful completion of our project who even assisted me in completing the project.

Kukkala Balaiah

Contents

Arrhythmia classification with extracted features from ECG signal for arrhythmia detection

Kukkala Balaiah

September 2022

Abstract

Cardiovascular disease refers to any critical condition that impacts the heart. Because heart diseases can be life-threatening, researchers are focusing on designing smart systems to accurately diagnose them based on electronic health data, with the aid of machine learning algorithms. This paper aims to use various machine learning algorithms and explore the influence between different algorithms and multi-feature in the time series. The ECG signals records constitute the time series as the research object. We extract various features from ECG signals between certain time periods. Moreover, we utilize Logistic Regression, Random Forest, Gradient Boosting, AdaBoost and XGBoost models to predict the individual is having arrythmia or not. Data preprocessing and feature selection steps were done before building the models. The models were evaluated based on the accuracy, precision and recall. The GBDT and XGBoost model performed best with 98.89% and 99.01% accuracy respectively.

1 Introduction

Cardiovascular Disease (CVD), commonly referred to as heart disease, encompasses a wide range of conditions that affect the heart, with the two most common conditions being ischemic heart diseases and strokes. The World Health Organization lists the most significant behavioural risk factors for CVD as maintaining an unhealthy diet, a sedentary lifestyle, tobacco use, and excessive consumption of alcohol. Prolonged exposure to these risk factors can present itself as an initial sign of CVD, which include elevated blood pressure, elevated blood glucose, raised blood lipids, and obesity. Warning signs listed by the American Heart Association include having one or more of the following: shortness of breath, persistent coughing or wheezing, swelling of the ankles and feet, constant fatigue, lack of appetite, and impaired thinking. Moreover, Coronavirus may cause heart disease. Worldwide, CVDs kill around 17 million a year, and death rates due to heart diseases have increased after the COVID-19 pandemic.

Initially, hospitals use regular tools like ECGs to determine and evaluate the stage of the disease.

Efficient early diagnosis can substantially reduce the risk and global burden of CVD by initiating treatment rapidly to prevent further health deterioration. Thus, there is an urgent need to develop machine learning models that can predict the probability of developing CVD depending on the risk factors present.

1.1 Problem Description

Arrhythmia is an abnormality of the heart's rhythm. It may beat too slowly, too quickly or irregularity. These abnormalities range from a minor inconvenience or discomfort to a potentially fatal problem. Narrowed heart arteries, a heart attack, abnormal heart valves, prior heart surgery, heart failure, cardiomyopathy and other heart damage are risk factors for almost any kind of arrhythmia. High Blood Pressure: This condition increases the risk of developing coronary artery disease.
The arrhythmia can be classified into two major categories.

- The first category consists of arrhythmia formed by a single irregular heartbeat called morphological arrhythmia.
- The other category consists of arrhythmia formed by a set of irregular heartbeats, are called rhythmic arrhythmia.

ECG: Electrocardiogram: It is a visual time series that records electrical activity generated by each cardiac cycle of the heart in real time. It is now widely used in heart-rate detection. The characteristics of ECG signals include random, low frequency and susceptible, resulting in the diagnostic results being unstable.

A successful ECG arrhythmia classification usually involves three important procedures: signal preprocessing, feature extraction, and classifier construction. Feature extraction is the important procedure that usually influences the classification performance of any ECG arrhythmia classification system. Therefore, the extraction of relevant features to achieve optimal classification results has become primary tasks for the ECG arrhythmia classification problems.

1.2 Problem Definition

Let $X = x_1, x_2, ...x_n$ are the features and $y \in N, S, V, F, Q$ are dependent classes which needs to be predicted. Firstly, we need to convert the multi label classes to binary classes $y \in 0, 1$. where 0 referred as Normal and 1 referred as heart attack. The main Objective is to study the extracted features of ECG signal and classify whether individual has arrhythmia or not.

2 Related Work

Abdelhaq Ouelli et al. [1] proposed a new automated detection method for cardiac arrhythmia. The detection system is implemented with integration of feature extraction and classification parts. In feature extraction phase of proposed method, the feature values for each arrhythmia are extracted using autoregressive (AR) and multivariate autoregressive (MVAR) modeling of one-lead and two-lead electrocardiogram signals. The classification is performed using a quadratic discriminant function (QDF) and a multilayer perceptron (MLP). The proposed method was tested over the entire MIT-BIH Arrhythmias Database, Creighton University Ventricular Tachyarrhythmia Database and MITBIH Supraventricular Arrhythmia Database and it yields an overall accuracy of 99.7%.

Can Ye et al. [2] proposed a new approach for arrhythmia classification based on a combination of morphological and dynamic features. Wavelet Transform (WT) and Independent Component Analysis (ICA) are applied separately to each heartbeat to extract corresponding coefficients, which are categorized as 'morphological' features. In addition, RR interval information is also obtained characterizing the 'rhythm' around the corresponding heartbeat providing 'dynamic' features. These two different types of features are then concatenated and Support Vector Machine (SVM) is utilized for the classification of heartbeats into 15 classes. The proposed method was tested over the entire MIT-BIH Arrhythmias Database and it yields an overall accuracy of 99.66% on 85945 heartbeats.

Rui Duan et al. [3] proposed a Ensemble method which is meta-algorithm to build strong classifiers based on a set of weak classifiers. This work explores some ensemble classifiers on UCI Arrhythmia Dataset to classify the heartbeat records. Two popular ensemble classifiers XGBoost and RandomForest are used, and the classic LogisticRegression is tried as a comparison. We build VotingClassifier based on the ensemble voting metaalgorithm and the above three built-in classifiers, and it outperforms even welltuned XGBoost and RandomForest. The best prediction accuracy 76% is achieved by the VotingClassifier in this multiclass classification problem.

Shi Haotian et al. [4] proposed a weighted extreme gradient boosting (XGBoost), a hierarchical classification method is proposed. A large number of features from 6 categories are extracted from the preprocessed heartbeats. Then recursive feature elimination is used for selecting features. Afterwards, a hierarchical classifier is constructed in classification stage. The hierarchical classifier is composed of threshold and XGBoost classifiers. And the XGBoost classifiers are improved with weights. The method was applied to an inter-patient experiment conforming AAMI standard. The obtained sensitivities for normal (N), supraventricular (S), ventricular (V), fusion (F), and Unknown beats (Q) were 92.1%, 91.7%, 95.1%, and 61.6%. Positive predictive values of 99.5%, 46.2%, 88.1%, and 15.2% were also provided for the four classes.

3 Dataset

- The datasets for training and evaluation is downloaded from INCART 2-lead Arrhythmia Database, MIT-BIH Arrhythmia Database, MIT-BIH Supraventricular Arrhythmia Database, Sudden Cardiac Death Holter Database.
- **Source:** https://www.kaggle.com/sadmansakib7/ecg-arrhythmia-classification-dataset
- The datasets contains features extracted two-lead ECG signal (lead II, V) from the MIT-BIH Arrhythmia dataset (Physionet).
- In this Dataset, We extract the features of ECG signal of viewpoints of type:
- Lead II represented by Limb Leads having 16 attributes.
- Lead V5 represented by Chest Leads having 16 attributes.
- Then we need to concatenate all these datasets to create one dataset which was used for our problem.
- The dataset constitutes of 963654 records and 34 features.

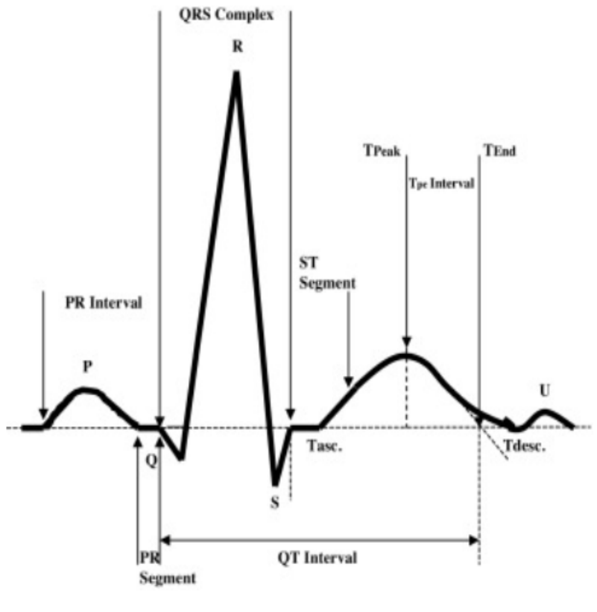


Figure 1: ECG Features

Normally, ECG signal consists of different waves to study the characteristics of Heart.

- P Wave: Depolarization of the atria.
- Q Wave: Activation of antereseptal region of the ventricular myocardium.

- R Wave: Depolarization of ventricular myocardium.
- S Wave: Activation of Postereobasal portion of the ventricles.
- T Wave: Rapid Ventricular repolarization.

Extracted Features:

- pre-RR: Determines RR interval with previous R Wave.
- post-RR: Determines RR interval with next R Wave.
- pPeak: Peak Amplitude of P wave.
- qPeak: Peak Amplitude of Q wave.
- rPeak: Peak Amplitude of R wave.
- sPeak: Peak Amplitude of S wave.
- tPeak: Peak Amplitude of T wave.
- pq-interval: Time interval from starting of P wave to the starting of Q wave.
- qt-interval: Time interval from starting of Q wave to the end of T wave.
- st-interval: Time interval from end of S wave to the end of T wave.
- qrs-interval: Time duration for completion of Q wave, R wave and S wave.
- Morphological Features determine the series of deflections away from the baseline on the ECG. These can be calculated from QRS-T obtained from ECG delineation. Features are like AUC of QRS-T segment, maximal positive deviation of QRS, maximal negative deviation of QRS.
  - qrs-morph0
  - qrs-morph1
  - qrs-morph2
  - qrs-morph3
  - qrs-morph4

Class Labels: This data can be classified into five types.

- N : Normal
- S : Supraventricular ectopic beat
- V : Ventricular ectopic beat
- F : Fusion beat
- Q : Unknown beat

4 Requirements and Analysis

4.1 Functional Requirements

- **Purpose**

  To predict the arrhythmia. The system tells whether there are chances for arrhythmia or not.

- **Inputs**

  Details of the extracted features of each heart-beat which was recorded in ECG device in a particular period are provided as the input to the system in the csv format.

- **Outputs**

  The system predicts the chances of arrhythmia. It tells whether there are chances for arrhythmia or not.

- **Usability**

  It provides ease of use. System is very user-friendly. It is less complicated to work.

4.2 Hardware Requirements

- Computer system
- 16, 64 GB Ram
- Intel i3 7th Gen processor
- Server Overhead
- Port 8051
- Internet connectivity

4.3 Software Requirements

- Notepad++ or any Code Editor
- Google Colab and Google Docs
- Streamlit
- MS-Excel
- Python 3
- Python libraries like pandas, NumPy, scikit-learn etc.
- Client environment may be Windows or Linux
- Web Browser

5 Exploratory Data Analysis and Featurization

1. **Dataset Concatenation**

   The datasets used for training and evaluation is downloaded from INCART 2-lead Arrhythmia Database, MIT-BIH Arrhythmia Database, MIT-BIH Supraventricular Arrhythmia Database, Sudden Car diac Death Holter Database.

   Firstly, We need to concatenate all datasets to form single dataset.

```python
import pandas as pd
import pickle
data1 = pd.read_csv("/content/drive/MyDrive/Project AI ML/Data/MIT-BIH Arrhythmia
    Database.csv/MIT-BIH Arrhythmia Database.csv")
data2 = pd.read_csv("/content/drive/MyDrive/Project AI ML/Data/MIT-BIH
    Supraventricular Arrhythmia Database.csv/MIT-BIH Supraventricular Arrhythmia
    Database.csv")
data3 = pd.read_csv("/content/drive/MyDrive/Project AI ML/Data/INCART 2-lead
    Arrhythmia Database.csv/INCART 2-lead Arrhythmia Database.csv")
data4 = pd.read_csv("/content/drive/MyDrive/Project AI ML/Data/Sudden Cardiac
    Death Holter Database.csv/Sudden Cardiac Death Holter Database.csv")
full_data = pd.concat([data1,data2,data3,data4], axis = 0,ignore_index=True)
if not os.path.isfile("/content/drive/MyDrive/Project AI ML/Data/Full_concat_data.
    pkl"):
  pickle.dump(full_data, open("/content/drive/MyDrive/Project AI ML/Data/
    Full_concat_data.pkl","wb"))
else:
  data = pickle.load(open("/content/drive/MyDrive/Project AI ML/Data/
    Full_concat_data.pkl","rb"))
```

Listing 1: Data Concatenation

2. **Imputation:**

   Dataset has missing values in all features. Hence for the further use, we should do imputation in order to fill missing values. To process the imputation I have used the average value of particular columns called Mean Imputation.

```python
for ele in data.columns:
    data[ele] = data[ele].fillna(data[ele].mean())
```

Listing 2: Data Imputation

3. **Remove the Null Category Rows**

   After concatenating the data, I checked the null values on the category type. Then, Some of the values are filled in either of five classes and around 76,000 rows which are not filled. Hence I broke down the dataset into set with all null values and set which have classes called Main dataset.

```python
lst = []
s = ['N', 'Q', 'SVEB', 'VEB', 'F']
for idx,row in data.iterrows():
  if(row.type not in s):
    lst.append(idx)
train_data = data.drop(index = lst,axis = 0)
```

Listing 3: Remove Null Category

4. **Convert the problem into Binary Class Classification**

   Then, In Main dataset I created one more column of name 'class' in order to convert the problem into Binary Class Classification, which defines 0 as Normal class and 1 as rest of 4 classes.

```python
def class_map(x):
    if(x == 'N'):
        return 0
    else:
        return 1
train_data['class'] = train_data['type'].map(class_map)

```

Listing 4: Convert Multi label classes into Binary class

5. **Plot the countplot**

```python
    import seaborn as sns
    sns.set_style('darkgrid')
    sns.countplot(x = 'class',data = train_data,hue= 'class')
    plt.show()

```

Listing 5: Distribution of class labels

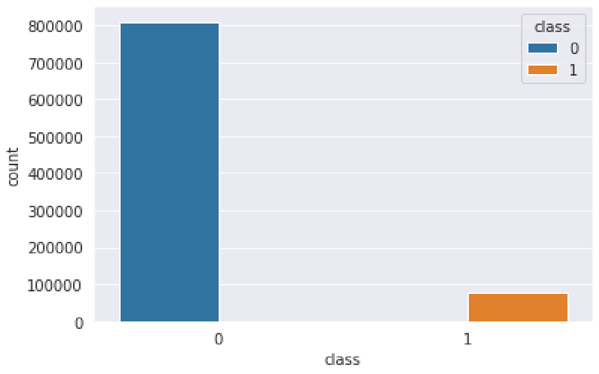


Figure 2: Count Plot

From this plot, It is clear that the dataset contains around 800,000 People with Normal Heart Beat and the rest around 80,000 people have some abnormal Heart Beats. Hence It is clear that, Dataset is with Imbalanced classes.

6. **Split the Dataset**

   The model prediction can only be verified if we check if our results are accurate or not. This can be done by performing validation on the training data itself. So, the train data set must be split so that one part of it can be used to train the model while the other part is used for validation. Now, On Main Dataset, we will split the dataset into train, validation, test with the ratio 60, 20, 20 respectively.

   Where 60% of the data will be used for training the model and 20% will be used for validating the results and 20% will be used for generalization of the model.

```python
import mlfuncs_clfn as ml
df_tr, df_eval, df_ts = ml.fn_tr_eval_ts_split_clf(df, eval_size = 0.2,
ts_size = 0.2)

```

Listing 6: Data Split

7. **Standardization**

   Perform the Standardization on train, validation and test datasets using following equation.

$$X_{standardize} = \frac{X - \mu}{\sigma}$$

```python
df_tr, df_eval, df_ts, _ = ml.fn_standardize_df(df_tr.iloc[:,2:], to_transform
 = [df_eval.iloc[:,2:], df_ts.iloc[:,2:]])

```

Listing 7: Data standardization

8. **3D Plot Distribution**

   Above plot is about 3D plot of Top 3 features.

   Using these plots also, we can't create simple decision based model since points of both classes can't be classified where all points looks like cumbersome.

   The below charts show that when the 1-pre-RR and 0-pre-RR are greater than 0, then all patients have Normal heart attack.

```python
ml.fn_plot_3d_clf(df_tr)

```

Listing 8: 3d Plot

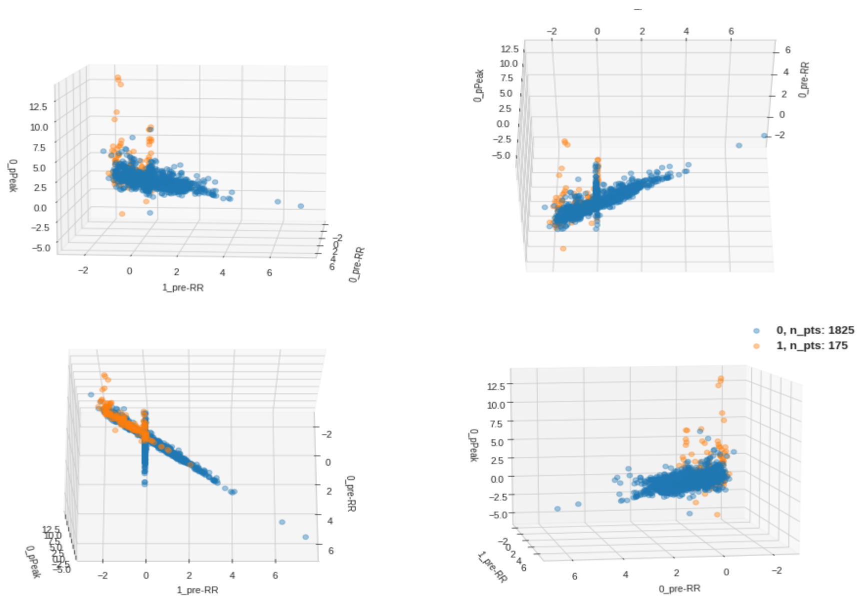


Figure 3: 3D Distribution Plot

9. **Visualize Feature Label Correlations**

   From this above bar plot, the feature that correlates more with class label are '0-pre-RR', '1-pre-RR', '0-pPeak', '1-qt-interval', '1-qrs-morph0'.

10. **Correlation-Correlation Matrix**

    Normally, correlation is the term that is referred to as the relation between two variables. If the correlation between variables is high, then variables have strong relationships with each other. If the correlation between variables is low, then variables have weak relationships or variables are hardly related. It should be done using a correlation metric between numeric (features) and categorical (labels) values like "f-ratio" From the correlation matrix, I have two feature pairs that have a correlation factor as 1. Those pairs are (0-qrs-morph0,0-qPeak) and (1-qrs-morph0, 1-qPeak) i.e., In a particular row, if one feature has high or low value then correspondingly the other feature has high or low value. Both features have the same effect on class labels. Hence we can remove one feature from the dataset. Hence It is better to have one feature from those pair of features.

```
    f_ratios_2,df_corr_2,best_feats_2 = ml.fn_feat_select_clfn(df_tr,
                                        thresh_feat_label = 50,
                                        thresh_feat_feat = 0.9,
                                        figsize = (15,7))
    df_tr = df_tr.loc[:, best_feats_2].assign(labels = y_tr)
    df_eval = df_eval.loc[:, best_feats_2].assign(labels = y_eval)
    df_ts = df_ts.loc[:, best_feats_2].assign(labels = y_ts)
```

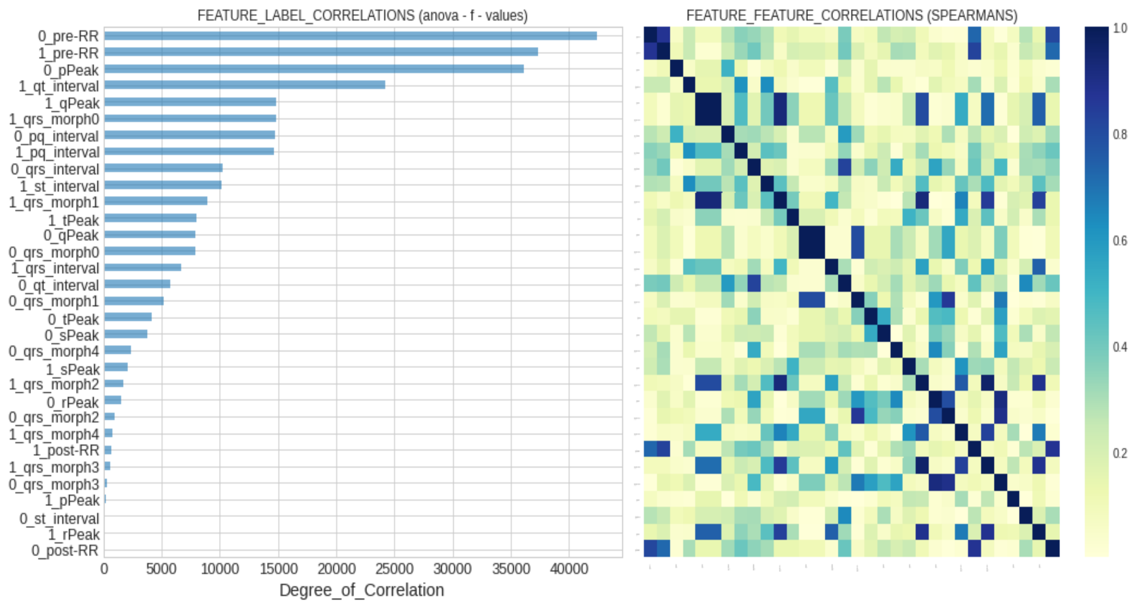


Figure 4: Feature Label Correlation

```
```

Listing 9: Feature selection

11. **Feature Selection**

ANOVA is a statistical analysis used to calculate the distance of difference (variance) between two clusters. ANOVA uses the f-ratio to calculate the magnitude of every feature and target class. Magnitude values above the f-ratio will be retained, and others will be discarded. In an ANOVA with class k, the variance among classes is defined as follows

$$\sigma^2_{v-all} = \frac{\Sigma(\bar{x_i} - \bar{x}) * n_i}{k-1}$$

where n_i is the value discovered from the calculation on the i-th class, $\bar{x_i}$ is the mean of the i-th class, and $\bar{x}$ is the mean of all classes; the class variance is defined as follows:

$$\sigma^2_{v-class} = \frac{\Sigma\Sigma(\bar{x_{ij}} - \bar{x})^2 - \Sigma(\bar{x_i} - \bar{x}) * n_i}{R-k}$$

Then, the f-ratio is calculated based on the degree of the two variances:

$$f\_ratio = \frac{\sigma^2_{v-all}}{\sigma^2_{v-class}}$$

Then, Select the features which have the f-ratio values above 50 and threshold correlation-correlation value above 0.90

12. **Distribution of Top 4 features correlated to class label**

```
df_feat_stats = ml.fn_distr_feats_labels(df_tr,n_top_feats = 4,figsize =
(15,7))

```

Listing 10: Distribution of top 4 features

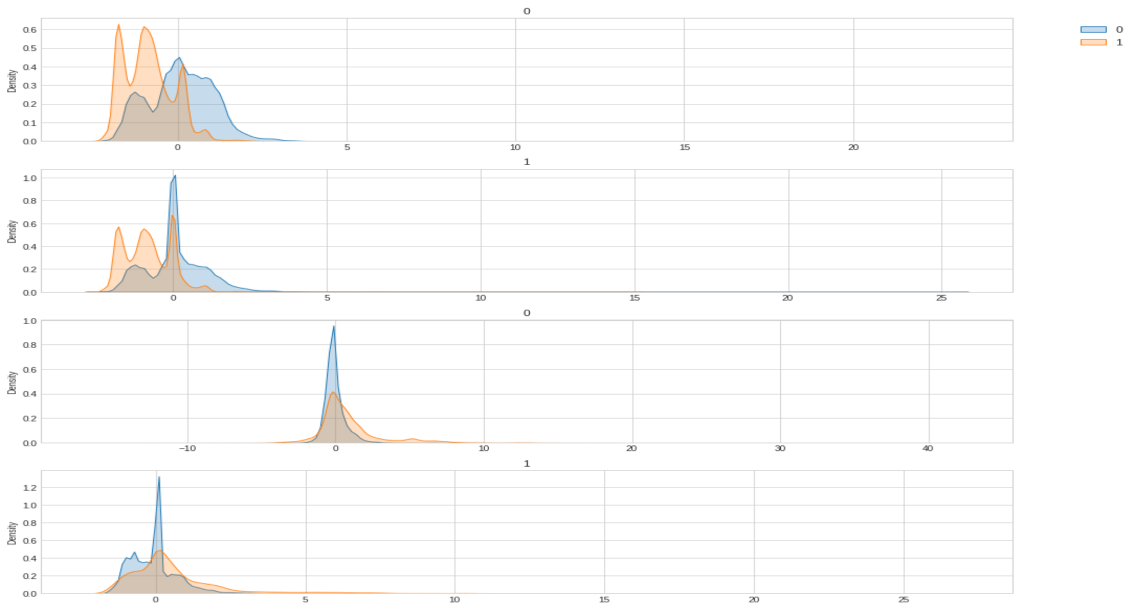


Figure 5: PDF values

13. **Principal Component Analysis**

    It is a dimensional reduction technique, which is often used for visualizing data. In this technique, firstly we will normalize the data D(n*d) matrix columns then we find the covariance matrix S(d*d). For this matrix, we find the eigenvalues and corresponding eigenvectors. We will sort the eigenvalues and corresponding eigenvectors can be placed in matrix as E(d*d‘). We will multiply the data matrix D and eigenvector matrix E. Then we will get the data with d‘ dimension.

14. **K means Clustering**

    It is a clustering algorithm. It is Unsupervised algorithm. In this technique, task is to group/-cluster "Similar" data points. Points in clusters are close together and points in a different clusters are farther away.

    Firstly, I applied the K-Means clustering with a different number of clusters and using the elbow method, I have taken the number of clusters as 4. Now, The plots above used to show distribution of each feature in each cluster.

15. **Centroid Visualization on PCA DATA of Principal Component 1 and Principal component 2**

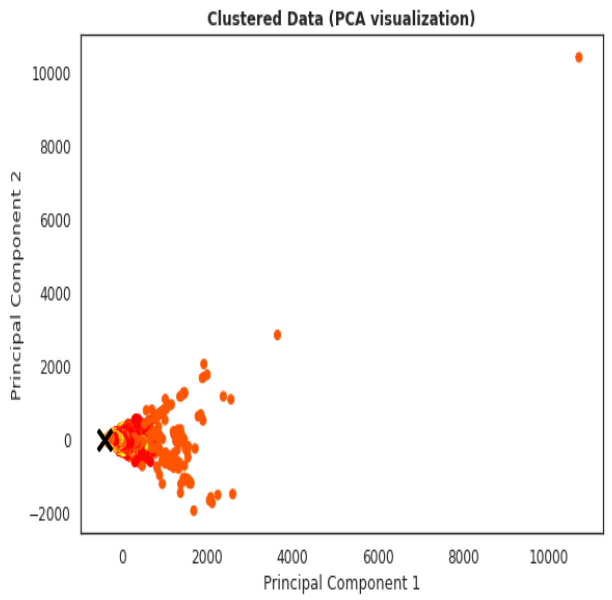


Figure 6: PCA values

6 Methods

6.1 Logistic Regression

- Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable(or output), y, can take only discrete values for given set of features(or inputs), X.
- Contrary to popular belief, logistic regression is a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as "1".
- Just like Linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.
- **Assumption:** Classes are almost or perfectly linearly separable.
- **Task:** Find a plane that best separates two classes.

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ...\beta_m x_m$$

where β's are the coefficient that needs to be learned, x's are the features/explanatory variables. Y is the response variable

6.2 Random Forest Classifier

- It is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.
- Unlike a Decision Tree that generates rules based on the data given, a Random Forest classifier selects the features randomly to build several decision trees and averages the results observed.
- Also, the over-fitting problem is fixed by taking several random subsets of data and feeding them to various decision trees.
- Yet, Decision Tree is computationally faster in comparison to Random Forest because of the ease in generating rules. In a Random Forest classifier, several factors need to be considered to interpret the patterns among the data points.

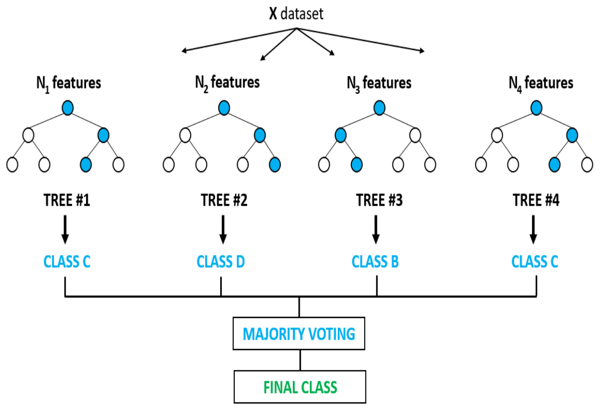


Figure 7: Random Forest Classifier

6.3 Gradient Boost Decision Tree

- Till now, We are using random forest method, which achieves better performance than a single decision tree simply by averaging the predictions of many decision trees.
- We refer to the random forest method as an "ensemble method". By definition, ensemble methods combine the predictions of several models (e.g., several trees, in the case of random forests).
- Next, we'll learn about another ensemble method called gradient boosting.
- **Gradient boosting** is a method that goes through cycles to iteratively add models into an ensemble.

- It begins by initializing the ensemble with a single model, whose predictions can be pretty naive. (Even if its predictions are wildly inaccurate, subsequent additions to the ensemble will address those errors.)
- Then, we start the cycle:
  - First, we use the current ensemble to generate predictions for each observation in the dataset. To make a prediction, we add the predictions from all models in the ensemble.
  - These predictions are used to calculate a loss function (like mean squared error, for instance).
  - Then, we use the loss function to fit a new model that will be added to the ensemble. Specifically, we determine model parameters so that adding this new model to the ensemble will reduce the loss.
  - Finally, we add the new model to ensemble, and ...
  - ... repeat!
- Gradient Boosting uses a standard method to build regression trees, where a typical metric such as MSE (Mean Squared Error) or a similar one is used to determine the best split for the tree.

  $$MSE = \frac{1}{n}\sum_{i=1}^{n}(predicted_i - observed_i)^2$$

  The algorithm calculates MSE for every possible node split and then picks the one with the lowest MSE as the one to use in the tree.
- Gradient Boosting algorithm uses the following formula to calculate the output value:

  $$value = \frac{(\sum_{i=1}^{n} Residual_i)^2}{\sum_{i=1}^{n}[Previous\ Probaility_i * (1 - Previous\ Probability_i)]}$$

  Residual is $actual\ value - predicted\ value$

  Previous probability is the probability of an event calculated at a previous step. The initial probability is assumed to be 0.5 for every observation, which is used to build the first tree. For any subsequent trees, the previous probability is recalculated based on initial prediction and predictions from all prior trees.

6.4 Adaptive Boost Classifier

- AdaBoost or Adaptive Boosting is one of the ensemble boosting classifiers.
- It combines multiple classifiers to increase the accuracy of classifiers and is a iterative ensemble method..
- AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier.

- The basic concept behind Adaboost is to set the weights of classifiers and training data samples in each iteration such that it ensures the accurate predictions of unusual observations.
- The classifier should be trained iteratively on various weighted training examples. In each iteration, It tries to provide a good fit to these examples by minimizing training error.
- It iteratively corrects the mistakes of the weak classifier and improves accuracy by combining weak learners. You can use many base classifiers with AdaBoost. AdaBoost is not prone to overfitting.
- AdaBoost is sensitive to noise data. It is highly affected by outliers because it tries to fit each point perfectly. AdaBoost is slower compared to XGBoost.

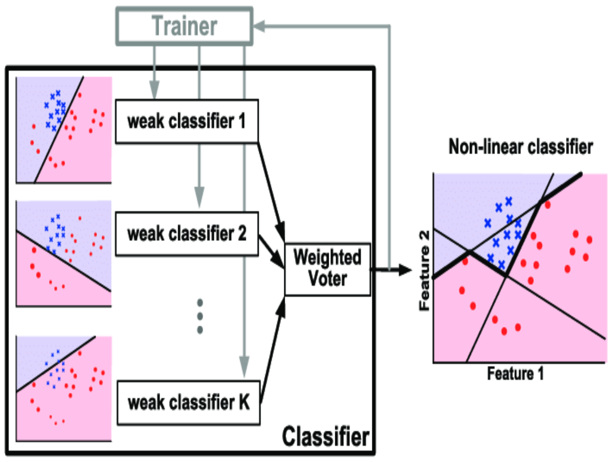


Figure 8: Adaptive Boost Classifier

6.5 Extreme Gradient Boost Classifier (XGBoost)

- XGBoost uses its own method of building trees where the Similarity Score and Gain determine the best node splits.

$$Similarity\ score = \frac{(\sum_{i=1}^{n} Residual_i)^2}{\sum_{i=1}^{n}[Previous\ Probaility_i * (1 - Previous\ Probability_i)] + \lambda}$$

Residual is $actual\ value - predicted\ value$

Previous probability is the probability of an event calculated at a previous step. The initial probability is assumed to be 0.5 for every observation, which is used to build the first tree. For any subsequent trees, the previous probability is recalculated based on initial prediction and predictions from all prior trees.

λ is a regularization parameter. Increasing lambda disproportionately reduces the influence of small leaves while having only a minor impact on larger leaves.

- Once we have the Similarity Score for each leaf, we then calculate the Gain using the following formula:

$$Gain = Left\ Leaf_{similarity} + Right\ Leaf_{similarity} - Root_{similarity}$$

  The node split with the highest Gain is then chosen as the best split for the tree.

- XGBoost also has a Gamma hyperparameter, which you can set manually. It allows you to prune nodes that have minimal Gain. Pruning happens where

$$Gain - Gamma < 0$$

  .

- XGBoost algorithm uses the following formula to calculate the output value:

$$value = \frac{(\sum_{i=1}^{n} Residual_i)^2}{\sum_{i=1}^{n}[Previous\ Probaility_i * (1 - Previous\ Probability_i)] + \lambda}$$

7 Evaluation Metric and Results

7.1 Key metric (KPI) to optimize

Metric is a function that is used to judge the performance of the model. It is a parameter on which the performance of our model depends. This problem comes under binary classification as this problem here is to build a model to predict whether individual has arrhythmia or not. The commonly used Performance Metrics for Classification Machine Learning Problems are:

- Accuracy
- Confusion Matrix
- Precision
- Recall
- F1-Score
- Receiving Operating Characteristic Curve
- Log-Loss

In this Project we going to use F1 score to evaluate the performance of the algorithm, we often use F1 score when the classes are imbalanced and there is a serious downside to predicting false negatives. This Metric is based on the confusion matrix. A confusion matrix is a summary of prediction results on a classification problem which consists of four parameters: true positive, false positive, true negative and false negative.

- **Confusion Matrix:**

  A confusion matrix or error matrix is a table that is often used to describe the performance of a classification model on a set of data for which the values are known. It allows the visualization of the performance of an algorithm. It allows very easy identification of confusion among classes. Many performance measures are identified from the confusion matrix.

  The number of correct and incorrect predictions are easily visualized through a confusion matrix. It shows how the model is confused by the predictions and the actual values. It also gives the type of error given by the model.

  The various terms of a confusion matrix are:

  1. **True Positive (TP):** The results under TP show the number of predicted positive results that are actually positive in the validation data.
  2. **True Negative (TN):** The results under TN show the number of predicted negative results that are actually negative in the validation data.
  3. **False Positive (FP):** The results under FP show the number of predicted positive results that are actually negative in the validation data.
  4. **False Negative (FN):** The results under TP show the number of predicted negative results that are actually positive in the validation data.

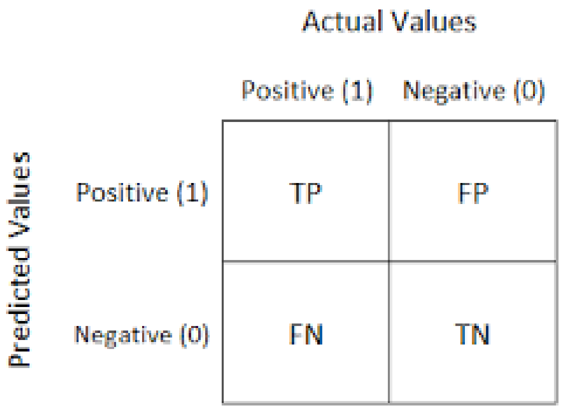


Figure 9: Confusion Matrix

```
TN = 0
FN = 0
FP = 0
TP = 0
random.seed(10)
y_pred = np.random.randint(0,2,100)
y_true = np.random.randint(0,2,100)
```

```python
for pair in zip(y_pred,y_true):
  if(pair[0] == 0):
    if(pair[1] == 0):
      TN +=1
    else:
      FN +=1
  else:
    if(pair[1] == 0):
      FP +=1
    else:
      TP +=1

Confusion_matrix = [[TN,FN],
                    [FP,TP]]
print("Confusion Matrix:",Confusion_matrix)
```

Listing 11: Confusion Matrix

- **Accuracy:**

  Of all cases, how many are correctly predicted?

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

```python
print("Accuracy:",float((TP+TN)/(TP+TN+FN+FP)))
```

Listing 12: Accuracy

- **Precision:**

  Of all predicted as positive cases, how many are really positive?

$$precision = \frac{TP}{TP + FP}$$

```python
precision = float(TP/(TP+FP))
print("Precision:",precision)
```

Listing 13: Precision

- **Recall:**

  Of all cases which are really positive, how many are predicted as positive?

$$recall = \frac{TP}{TP + FN}$$

```python
recall = float(TP/(TP+FN))
print("Recall:",recall)
```

Listing 14: Recall

- **F1-Score:**

  F1 score is the harmonic mean of precision and recall.

  It is used when False Negative and False positive is more important than True Positive and True Negative.

$$f1_{score} = \frac{2 * precision * recall}{precision + recall}$$

```python
f1_score = float((2*precision*recall)/(precision+recall))
print("F1 Score:",f1_score)
```

Listing 15: F1-Score

- **Receiver Operating Characteristic (ROC):**

  An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

  - True Positive Rate
  - False Positive Rate

  True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR) is defined as follows:

$$FPR = \frac{FP}{FP + TN}$$

An ROC curve plots TPR vs FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.

```python
def function_tpr_fpr(y_pred,y_true):
  TN = 0
  FN = 0
  FP = 0
  TP = 0
  for pair in zip(y_pred,y_true):
    if(pair[0] == 0):
```

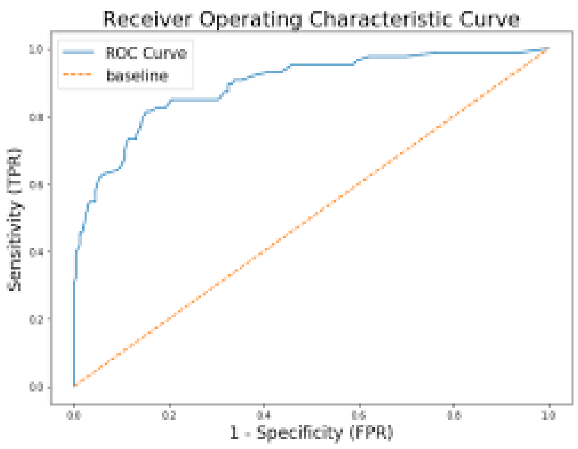


Figure 10: ROC Curve

```
      if(pair[1] == 0):
        TN +=1
      else:
        FN +=1
    else:
      if(pair[1] == 0):
        FP +=1
      else:
       TP +=1
  TPR = TP/(TP+FN)
  FPR = FP/(TN+FP)
  return TPR,FPR

y_true = np.random.randint(0,2,100)
y_pred = np.random.uniform(0,1,100)

dictionary = {}
for pair in zip(y_true,y_pred):
  dictionary[pair[1]] = pair[0]
y_pred_sorted = sorted(dictionary,reverse = True)

sorted_dict = {}
for key in y_pred_sorted:
  sorted_dict[key] = dictionary[key]
keys = sorted_dict.keys()
values =sorted_dict.values()
tpr = []
fpr = []

for idx,key in enumerate(keys):
  #print("Iteration No:",idx+1)
  a = np.array([0]*y_true.shape[0])
  a[:idx+1] = 1
  a[idx+1:] = 0
  TPR,FPR = function_tpr_fpr(a,values)
  tpr.append(TPR)
  fpr.append(FPR)

tpr = np.array(tpr)
fpr = np.array(fpr)
plt.plot(tpr,fpr,'b-')
plt.show()
```

Listing 16: ROC Curve

- **Log-Loss:**

  Log-loss is indicative of how close the prediction probability is to the corresponding actual/true value (0 or 1 in case of binary classification). The more the predicted probability diverges from the actual value, the higher is the log-loss value.

$$logloss_i = y_i * ln(p_i) + (1 - y_i) * ln(1 - p_i)$$

  where i is the given observation/record, y is the actual/true value, p is the prediction probability, and ln refers to the natural logarithm (logarithmic value using base of e) of a number.

```
def func(y_test,y_pred):
  sum = 0
  N = y_test.shape[0]
  for i in range(0,N):
    sum += np.log(y_pred[i])
  return (-1/N)*sum

y_test = np.array([0,1,0,1,2,1,0,2])
y_pred = np.array([0.9,0.5,0.6,0.8,0.3,0.6,0.9,0.2])
log = func(y_test,y_pred)
print(log)
```

Listing 17: LogLoss

7.2 Evaluation and Comparison of Methods

- **Logistic Regression**

  In Logistic regression, We trained the Logistic regression model with Hyperparameters C and penalty.

  Threshold Value : 0.72

  Best model:

  $$\{'C' : 10000000000,' class\_weight' :' balanced',' max\_iter' : 30000,' penalty' :' l2'\}$$

- **Random Forest**

  In Random Forest, We trained the Random Forest model with Hyperparameters max_depth and n_estimators.

  Threshold Value : 0.36

  Best model:

  $$\{'bootstrap' : True,' class\_weight' :' balanced',' max\_depth' : 15,' n\_estimators' : 120, \}$$

- **Gradient Boost Decision Tree**

  In Gradient Boosting Classifier, We trained the Gradient Boosting model with Hyperparameters max_depth and n_estimators.

  Threshold Value : 0.10

  Best model:

  $$\{'max\_depth' : 15,' max\_features' :' sqrt',' n\_estimators' : 200,' random\_state' : 0\}$$

- **Adaptive Boost Classifier**

  In Adaptive Boosting Classifier, We trained the Adaptive Boosting model with Hyperparameters learning_rate and n_estimators.

  Threshold Value : 0.50

  Best model:

  $$\{'learning\_rate' : 0.8,' n\_estimators' : 200,' random\_state' : 0\}$$

- **XGBoost Classifier**

  In XG Boosting Classifier, We trained the XG Boosting model with Hyperparameters max_depth and n_estimators.

  Threshold Value : 0.37

  Best model:

  $$\{'class\_weight' :' balanced',' learning\_rate' : 0.2,' max\_depth' : 15,' n\_estimators' : 120\}$$

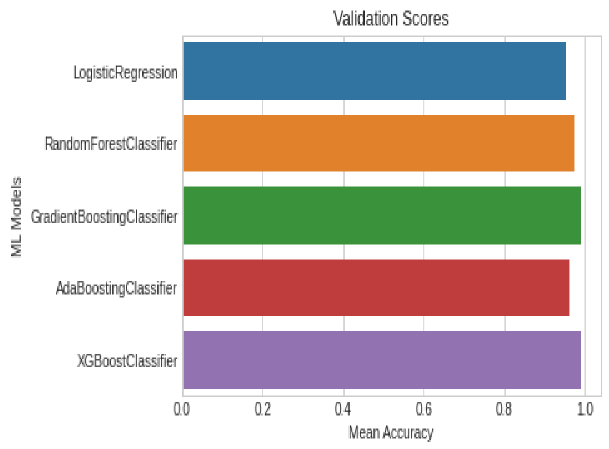


Figure 11: Models Performance Comparision

Arrhythmia Prediction Performance			
Methods	Accuracy	Precision	Recall
Logistic Regression	0.95255	0.73090	0.72840
Random Forest	0.97520	0.79868	0.96477
GBDT	0.98894	0.92122	0.95681
AdaBoost Classifier	0.96109	0.89030	0.63636
XGBoost Classifier	0.99019	0.94942	0.93863

Table 1: Evaluation of Each Model for Arrhythmia Prediction

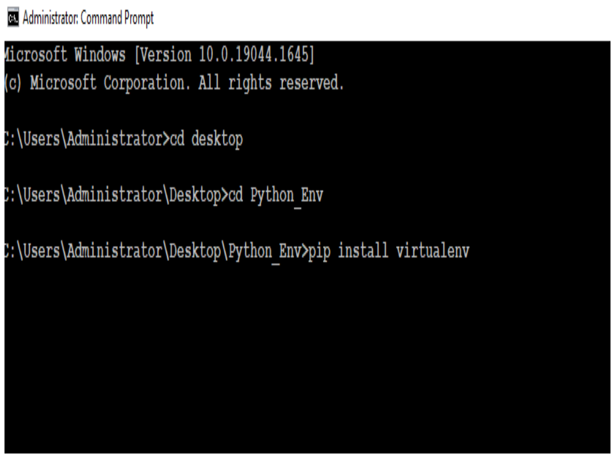
- For this problem: Arrhythmia classification, we should focus on the positive class that determines whether this positive class is detected accurately or not.
- i.e., If the patient is having a heart attack, he should be detected positively for further medical procedures. Hence, for class_1, precision should be high and also recall should be high.
- And also, If the patient didn't have a heart attack, he can be predicted as a positive class. Then also It won't be that much of a problem.
- So, From above plot, It is clear that XG boost and Gradient Boosting Algorithms have pretty good accuracy.
- And also, the precision and recall of both models is also high.

8 Productionization and Deployment

8.1 Virtual Environment

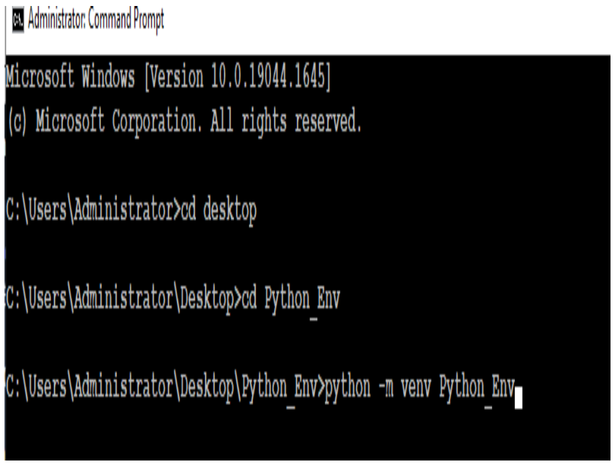
- Virtualization is the technology that can simulate your physical hardware (such as CPU cores, memory, disk) and represent it as a separate machine. It has its own Guest OS, Kernel, process, drivers, etc. Therefore, it is hardware-level virtualization. Most common technology is ”VMware” and ”Virtual Box”.
- A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them. This is one of the most important tools that most of the Python developers use.
- Imagine a scenario where you are working on two python projects and one of them uses tensorflow 1.0 and other uses tensorflow 2.0 and so on. In such situations virtual environment can be really useful to maintain dependencies of both the projects.
- By default, every project on our system will use these same directories to store and retrieve site packages.
- Now, in the above example of two projects, you have two versions of Django. This is a real problem for Python since it can’t differentiate between versions in the “site-packages” directory. So both v1.9 and v1.10 would reside in the same directory with the same name.
- This is where virtual environments come into play. To solve this problem, we just need to create two separate virtual environments for both the projects.The great thing about this is that there are no limits to the number of environments you can have since they’re just directories containing a few scripts.
- We use a module named virtualenv which is a tool to create isolated Python environments. virtualenv creates a folder which contains all the necessary executables to use the packages that a Python project would need.
- Virtualization supports executing multiple operating systems on a single physical server.
- On Command Prompt,

```
        pip install virtualenv
        python -m venv Python_Env
        Scripts\activate.bat

```



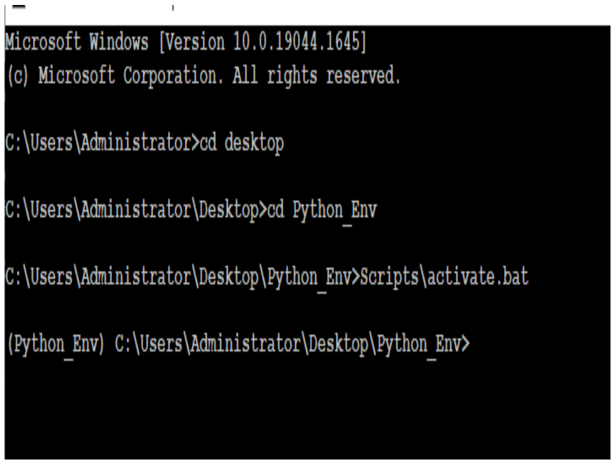
```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd desktop

C:\Users\Administrator\Desktop>cd Python_Env

C:\Users\Administrator\Desktop\Python_Env>pip install virtualenv
```

Figure 12: Virtual Environment Module



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd desktop

C:\Users\Administrator\Desktop>cd Python_Env

C:\Users\Administrator\Desktop\Python_Env>python -m venv Python_Env
```

Figure 13: Creating Virtual Environment



```
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator>cd desktop

C:\Users\Administrator\Desktop>cd Python_Env

C:\Users\Administrator\Desktop\Python_Env>Scripts\activate.bat

(Python_Env) C:\Users\Administrator\Desktop\Python_Env>
```

Figure 14: Activate the Virtual Environment

8.2 Containerization

- Containerization is "OS-level virtualization". It doesn't simulate the entire physical machine. It just simulates the OS of your machine. Therefore, multiple applications can share the same OS kernel. Containers play similar roles as virtual machine but without hardware virtualization. Most common container technology is "Docker".
- When code is developed in a specific computing environment and transferred to a different environment, there is a high possibility of the code to result in bugs and errors due to missing dependencies, libraries, or any configuration setting files.
- Containerization is the process of bundling the application code along with the libraries, configuration files, and dependencies required for the application to run cross-platform.
- Thus, Containerization is an application-packaging approach where the code is written once and capable of executing anywhere, making the application highly portable. This itself is an advantage of containerization. Docker is the example of a containerization platform.
- The single package of software built using containerization technology is called a 'Container'. The container is a standalone package, independent of the host operating system. Being as such, it can then execute across multiple platforms, without any issues.
- Containerization supports deploying multiple applications developed in the environment of one operating system residing on a single virtual machine or a server.

8.3 IAAS and PAAS

IAAS

- IAAS stands for Infrastructure As A Service.
- It is a form of cloud computing service that offers compute, storage and networking resources on-demand, usually on a pay-as-you-go basis.
- Businesses can purchase resources on-demand and as-needed instead of having to buy the hardware outright.
- IaaS gives you virtualized resources such as servers, disks, networks, and IP addresses, you are still responsible for administering the operating system, data, applications, middleware and runtimes.
- A dashboard or an API gives you complete control over the entire infrastructure.

- IaaS gives you the flexibility to purchase only the computing you need and scales them up or down as needed. If you are looking to migrate an application as-is from an on-premises data center to the cloud, choose the IaaS model.
- Because of its speed of deployment, IaaS is a quick and flexible way to build up and take down development and testing environments.
- Examples of IaaS include Rackspace, Amazon Web Services (AWS) Elastic Compute Cloud (EC2), Microsoft Azure, Google Compute Engine (GCE) and Joyent.

PAAS

- PAAS stands for Platform As A Service.
- It Builds on virtualization technology, so resources can easily be scaled up or down as your business changes.
- It provides a framework for application creation and deployment.
- Accessible to numerous users via the same development application and Integrates web services and databases.
- The technical stack required for application development is available on the cloud, which requires no download or local installation.
- With PaaS, developers can focus on building their applications without having to worry about operating systems, software updates, storage or infrastructure.
- PaaS applications are scalable and highly available as they take on certain cloud characteristics.
- Popular PaaS services include AWS Elastic Beanstalk, Engine Yard, Red Hat OpenShift, Google App Engine, Heroku, Appfog and Azure App Service.

8.4 Heroku Platform

- Heroku is a container-based cloud Platform As A Service that enables developers to build, run, and operate applications entirely in the cloud.
- The Heroku network runs the customer's apps in virtual containers which execute on a reliable runtime environment.
- Applications that are run on Heroku typically have unique domain names, which are used to route HTTP requests to the correct container.
- Applications as services use application containers. Containers are designed to package and run services.
- Application containers — referred to as "dynos" in the context of the Heroku platform.
- Heroku calls these containers "Dynos".
- These Dynos can run code written in Python. Heroku also provides custom buildpacks with which the developer can deploy apps in any other language.
- Dynos are simply lightweight Linux containers dedicated to running your application processes. At the most basic level, a newly deployed app to Heroku will be supported by one Dyno for running web processes.
- You then have the option of adding additional Dynos and specifying Dyno processes in your procfile.
- There are three varieties of dynos.
  - Web Dynos
  - Worker Dynos
  - One-off Dynos
- Projects created in Heroku are bound to repositories in GitHub. Heroku's integration with GitHub provides automated builds and deploys of the latest version of code.
- GitHub is a trusted repository of source code, so Heroku's integration with GitHub and other tools helps developers optimize their efforts and save stakeholders time and money over the course of development projects.

8.5 App

Streamlit APP Design

- In this step a web application developed using Streamlit. It provides a functionality to the user for uploading the input in csv format.
- As part of user interface, I will be developing a Streamlit app. Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science.
- Streamlit allows you to write an app the same way you write a python code. The file will be stored in .py format.
- We are creating a web app with Title as **Arrhythmia classification with extracted features from ECG signal for arrhythmia detection**, with menus as Home and Prediction.

```python
import pandas as pd
import numpy as np
from joblib import load
import streamlit as st
import pickle

def main():
    st.title("Arrhythmia classification with extracted features from ECG signal for arrhythmia detection")
    app_mode = st.sidebar.selectbox('Select Page',['Home','Prediction'])
    if app_mode == "Home":
        st.header("Home")
    elif app_mode == 'Prediction':
        st.header("Model Deployment")
    return app_mode

def model_prediction(X):
    st.write('<p style="font-family:sans-serif; color:Brown; font-size: 22px;">Our Best Model is GradientBoost and XGBoost</p>',unsafe_allow_html = True)

    choice = st.selectbox(

    'Select the model you want?',

    ('RandomForest','GradientBoost','AdaBoost','LogisticRegression',"XGBoost"))

    #displaying the selected option

    st.write('You have selected:', choice)
    if(choice == 'RandomForest'):
        model = st.file_uploader("CHOOSE RandomForest CLASSIFIER JOBLIB FILE")
        threshold = pickle.load(open("./Main_Data/rf_threshold.pkl","rb"))
    elif(choice =='GradientBoost'):
        model = st.file_uploader("CHOOSE GradientBoosting CLASSIFIER JOBLIB FILE")
        threshold = pickle.load(open("./Main_Data/gb_threshold.pkl","rb"))
    elif(choice == "AdaBoost"):
        model = st.file_uploader("CHOOSE AdaBoost CLASSIFIER JOBLIB FILE")
        threshold = pickle.load(open("./Main_Data/ab_threshold.pkl","rb"))
    elif(choice =="LogisticRegression"):
        model = st.file_uploader("CHOOSE LogisticRegression CLASSIFIER JOBLIB FILE")
        threshold = pickle.load(open("./Main_Data/log_threshold.pkl","rb"))
    else:
        model = st.file_uploader("CHOOSE XGBoost CLASSIFIER JOBLIB FILE")
        threshold = pickle.load(open("./Main_Data/xg_threshold.pkl","rb"))

    st.write("Threshold value:",threshold)
    if model is not None and X is not None:
        model = load(model)
        #st.write(model.n_features_)
        y_pred = model.predict_proba(X.values)[:, 1]
        y_pred_updated = np.array([1 if ele >= threshold else 0 for ele in y_pred])
        #st.write("PREDICTED values:",y_pred_updated)
        values = st.number_input(
            "Pick a number",
            0,X.shape[0])
        st.write("CHOSEN TEST CASE ENTRY:",values)
        if st.button('Predict Arrhythmia'):
            if(y_pred_updated[values]  == 0):
                t = '<p style="font-family:sans-serif; color:Green; font-size: 42px;">It\'s great that You don\'t have heart attack...</p>'
                st.write(t,unsafe_allow_html = True)
            else:
                l = '<p style="font-family:sans-serif; color:Red; font-size: 42px;">It\'s sad that you had heart attack...</p>'
                st.write(l,unsafe_allow_html  = True)

def load_data():
    data_file = st.file_uploader("CHOOSE CSV FILE CONTAINING VECTORS TO BE PREDICTED UPON")
    if data_file is not None:
        df = pd.read_csv(data_file)
        if st.checkbox("Show Data"):
            st.dataframe(df)
        return df


```

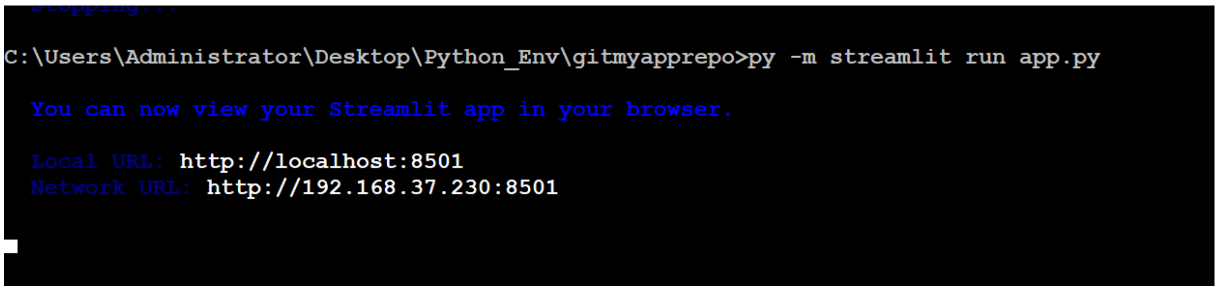
```
def load_mean_std_data():
    df_mean_std = pd.read_csv("./Main_Data/df_mean_std.csv")
    if st.checkbox("Show Mean and standard deviation of each features..."):
        st.write(df_mean_std)

    return df_mean_std

def standardization(data, mean_std):
    if data is not None:
        columns = data.columns
        for ind, row in mean_std.iterrows():
            data[columns[ind]] = (data[columns[ind]] - row['means'])/row['std']

        return data

def load_best_features():
    best_feats = pickle.load(open("./Main_Data/best_features.pkl","rb"))
    if st.checkbox("Show Best features..."):
        st.write(best_feats)

    return best_feats

def featurnselection(data, best_feats):
    if data is not None:
        y = data.loc[:,'class']
        X = data.loc[:,best_feats]

        return X


if __name__ == '__main__':
    choice = main()


if choice == "Home":
    """Arrhythmia is an abnormality of the h e a r t s rhythm. It may beat tooslowly,
    too quickly or irregularity. These abnormalities range from a minor inconvenience
    or discomfort to a potentially fatal problem. Narrowed heart arteries, a heart
    attack, abnormal heart valves, prior heart surgery, heart failure, cardiomyopathy
    and other heart damage are risk factors for almost any kind of arrhythmia. High
    Blood Pressure: This condition increases the risk of developing coronary artery
    disease. The arrhythmia can be classified into two major categories. The first
    category consists of arrhythmia formed by a single irregular heartbeat called
    morphological arrhythmia. The other category consists of arrhythmia formed by a set
     of irregular heartbeats, are called rhythmic arrhythmia. The Cardiovascular
    diseases are the leading cause of death. The current identification method of
    diseases is analyzing the Electrocardiogram [ECG], which is medical monitoring
    technology recording cardiac activity. Unfortunately, looking for experts to
    analyze a large amount of ECG data consumes too many medical resources. Therefore,
    a method based on Machine Learning or Deep Learning to accurately classify ECG
    characteristics."""

elif choice == 'Prediction':
    st.markdown("## Dataset :")
    data = load_data()
    st.markdown("## Mean and Variance of Each Features :")
    data_load_state = st.text("Loading mean and std data...")
    data_mean_std = load_mean_std_data()
    data_load_state.text("Loading mean and std data...done!")
    st.markdown("## Standardized Data :")
    input_data = standardization(data, data_mean_std)
    if st.checkbox("Show Standardized Data..."):
        st.write(input_data)
    st.markdown("## Feature Selection :")
    data_load_state = st.text("Loading best features...")
    best_feats = load_best_features()
    data_load_state.text("Loading best features...done!")
    X =  featurnselection(input_data,best_feats)
    if st.checkbox("Show Selected Features Data..."):
        st.write(X)
    st.markdown("## Model Predictions :")
    model_prediction(X)
```

Listing 18: Streamlit app code:app.py

Firstly, Create the Virtual Environment in local system and Install required packages for saved model to run smoothly.

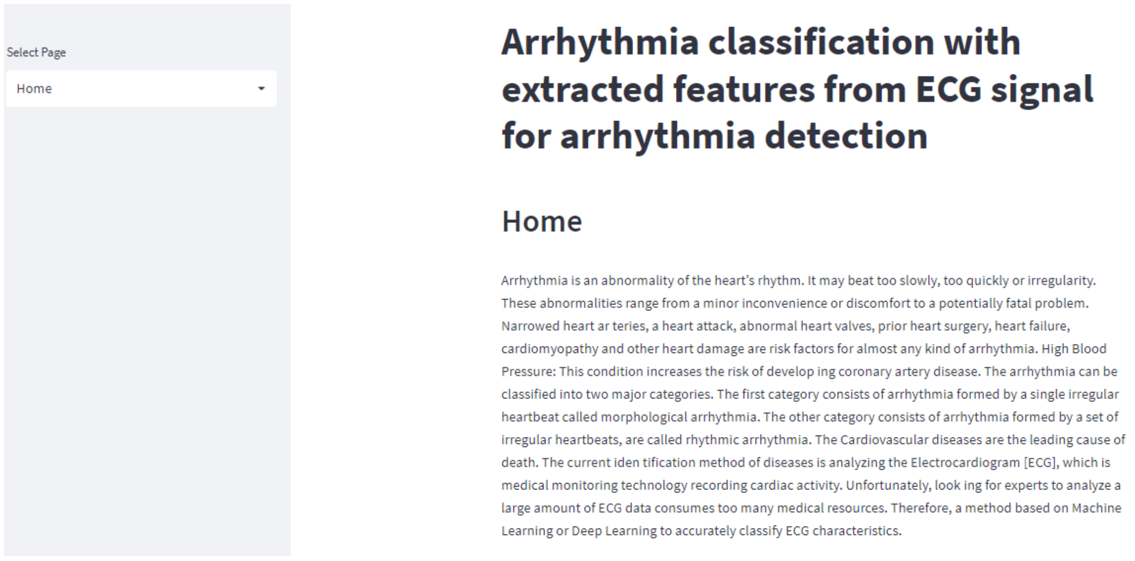
```
pip install virtualenv
python -m venv Python_Env  #To create Pyhton Environment
Scripts\activate.bat  #To activate python environment
```

To run above created streamlit app in local system by using following command:



```
C:\Users\Administrator\Desktop\Python_Env\gitmyapprepo>py -m streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.37.230:8501
```

Figure 15: Run Streamlit App

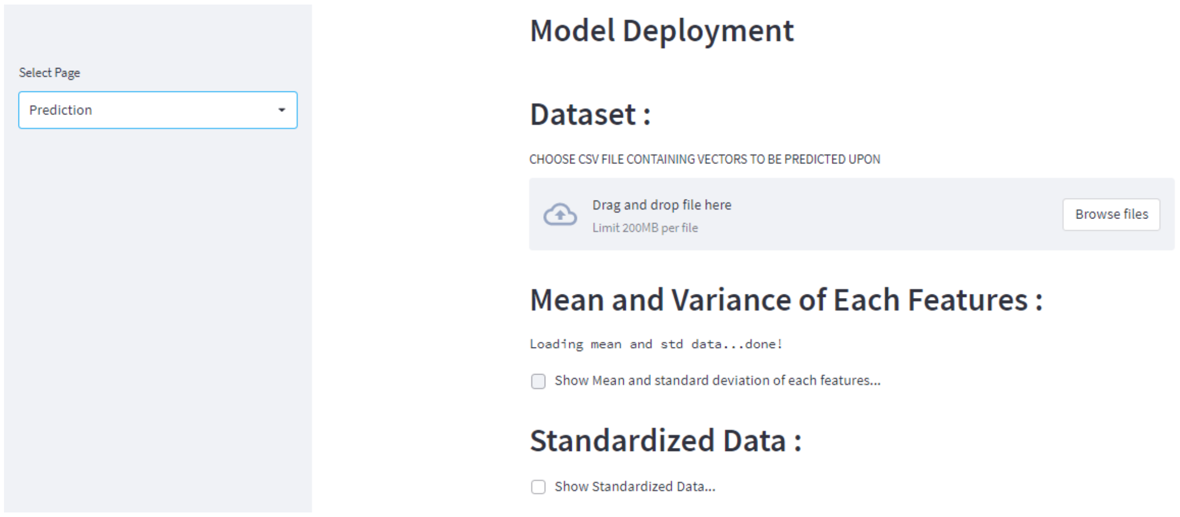


Arrhythmia classification with extracted features from ECG signal for arrhythmia detection

Home

Arrhythmia is an abnormality of the heart's rhythm. It may beat too slowly, too quickly or irregularity. These abnormalities range from a minor inconvenience or discomfort to a potentially fatal problem. Narrowed heart ar teries, a heart attack, abnormal heart valves, prior heart surgery, heart failure, cardiomyopathy and other heart damage are risk factors for almost any kind of arrhythmia. High Blood Pressure: This condition increases the risk of develop ing coronary artery disease. The arrhythmia can be classified into two major categories. The first category consists of arrhythmia formed by a single irregular heartbeat called morphological arrhythmia. The other category consists of arrhythmia formed by a set of irregular heartbeats, are called rhythmic arrhythmia. The Cardiovascular diseases are the leading cause of death. The current iden tification method of diseases is analyzing the Electrocardiogram [ECG], which is medical monitoring technology recording cardiac activity. Unfortunately, look ing for experts to analyze a large amount of ECG data consumes too many medical resources. Therefore, a method based on Machine Learning or Deep Learning to accurately classify ECG characteristics.

Figure 16: Streamlit app screenshot



Model Deployment

Dataset :

CHOOSE CSV FILE CONTAINING VECTORS TO BE PREDICTED UPON

Mean and Variance of Each Features :

```
Loading mean and std data...done!
```

☐ Show Mean and standard deviation of each features...

Standardized Data :

☐ Show Standardized Data...

Figure 17: Streamlit app screenshot

App Deployment on Heroku

- In this app, we are deploying the whole machine learning pipeline into a production system, into a real-time scenario.
- Initially I created local folder having app.py file along with Data folder and requirements.txt and runtime.txt
- In requirements.txt file, I stored all required packages that I have used my model to run

```
numpy==1.21.6
pandas==1.3.5
```

```
protobuf==3.19.6
scikit-learn==1.0.2
pickleshare==0.7.5
joblib==1.2.0
streamlit==1.13.0

```

Listing 19: requirement.txt

- In runtime.txt file, I added the python version to be installed on heroku stack.

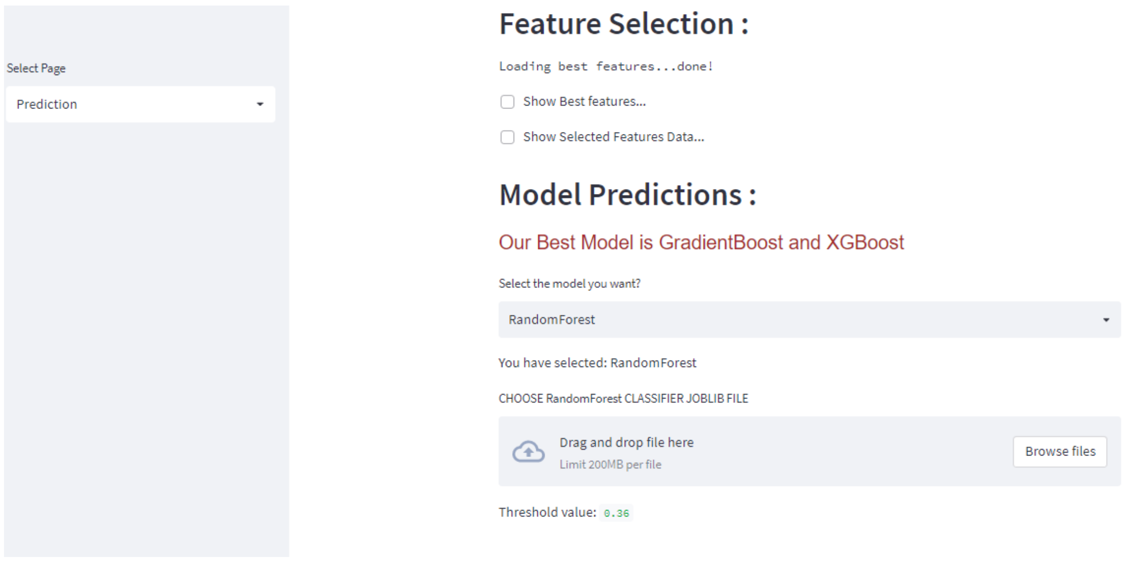
```
python-3.10.7

```

Listing 20: runtime.txt

- Procfile indicates which is the starting file to run on web app.

```
web: streamlit run app.py --server.headless true\
                         --server.enableXsrfProtection false\
                        --server.enableCORS false --server.port $PORT\
                       --server.enableWebsocketCompression false

```

In Procfile, I defined web dyno for running web processes.



Feature Selection :

Loading best features...done!

☐ Show Best features...

☐ Show Selected Features Data...

Model Predictions :

Our Best Model is GradientBoost and XGBoost

Select the model you want?

RandomForest

You have selected: RandomForest

CHOOSE RandomForest CLASSIFIER JOBLIB FILE

Threshold value: 0.36

Figure 18: Streamlit app screenshot

- **Github**

  Firstly, I created the GitHub account.

  Then, I created one public repository on Github.

  I have done Git Install on my windows system.

  Then I have created git repository and add all required files for app using git command line.

```
"""Add all required files for app to run on heroku in folder"""
$git init   #Initialize the Git repository on local folder
$git add .  #Add files in local folder to git repository
$git commit -m "first commit"  #Commit all files local git repository
$git remote add origin https://github.com/balaiahkukkala/myapp.git
$git push -u origin master   #Push all local repo to git repo

```

Listing 21: Git commands on CMD

- **Heroku**

  Firstly, I created the Heroku account.

  In heroku, I created app name: **vikash-app**

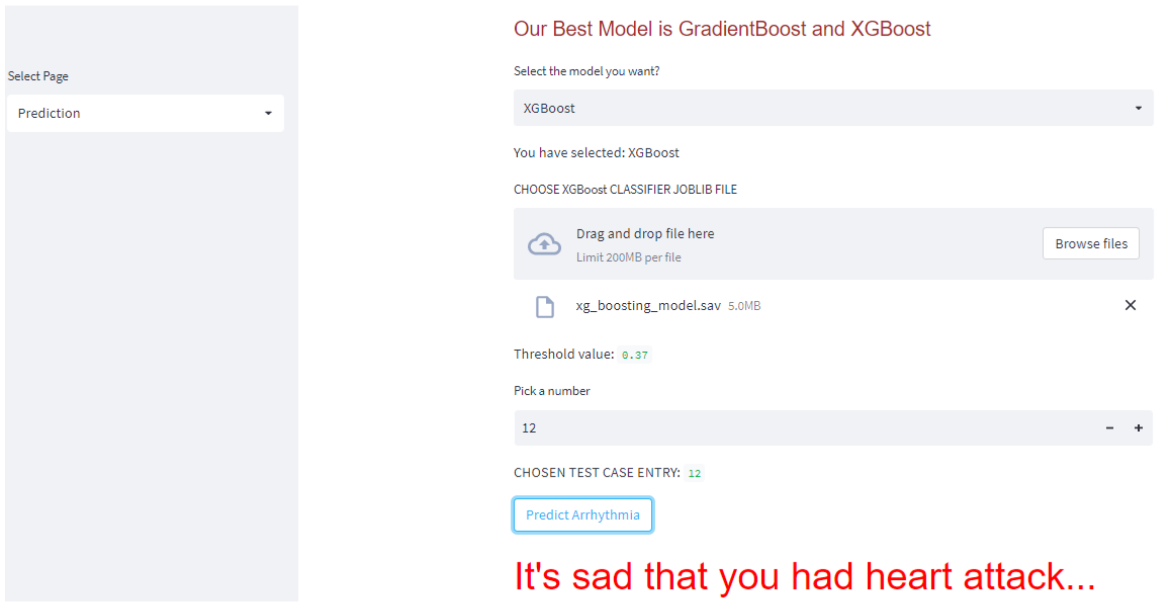
  Then, I connected to my Github **myapp** repository through Heroku.

  Thenafter, I deploy the app where link for app will be generated.

  App link: `https://vikash-app.herokuapp.com/`

  github app repository link: `https://github.com/balaiahkukkala/myapp`

  In my git app repo, our saved models and required data are stored folder **Main_Data**


Select Page: Prediction

Our Best Model is GradientBoost and XGBoost

Select the model you want?: XGBoost

You have selected: XGBoost

CHOOSE XGBoost CLASSIFIER JOBLIB FILE

Drag and drop file here
Limit 200MB per file
Browse files

xg_boosting_model.sav 5.0MB

Threshold value: 0.37

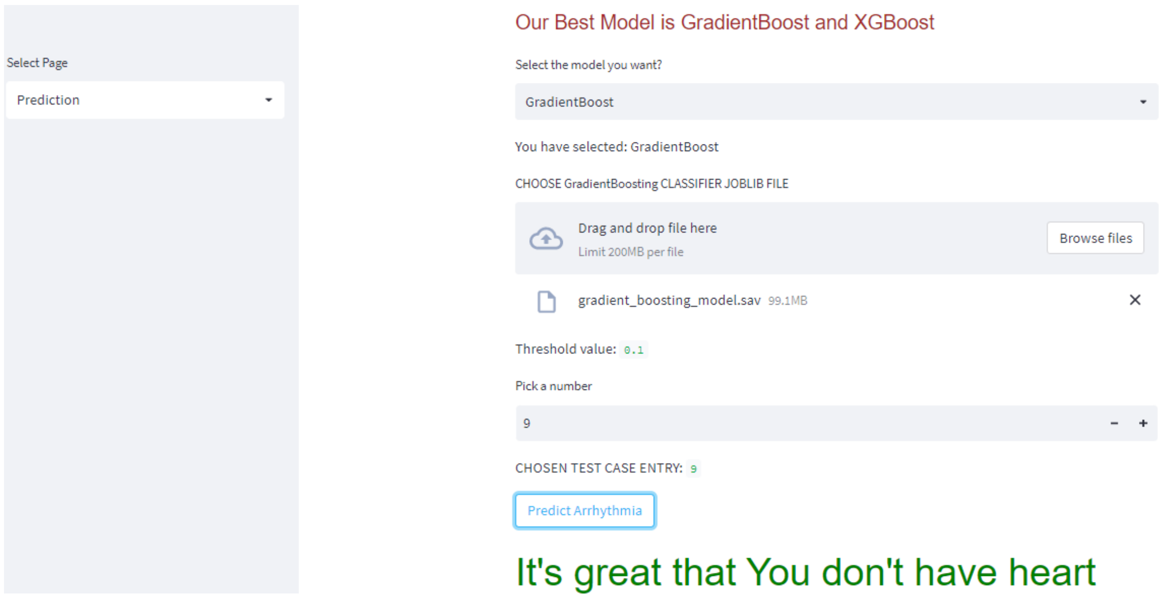
Pick a number: 12

CHOSEN TEST CASE ENTRY: 12

Predict Arrhythmia

It's sad that you had heart attack...

Figure 19: Streamlit app screenshot



Our Best Model is GradientBoost and XGBoost

Select the model you want?

GradientBoost

You have selected: GradientBoost

CHOOSE GradientBoosting CLASSIFIER JOBLIB FILE

gradient_boosting_model.sav 99.1MB

Threshold value: 0.1

Pick a number

9

CHOSEN TEST CASE ENTRY: 9

It's great that You don't have heart

Figure 20: Streamlit app screenshot

9 Conclusion

Cardiovascular disease is a major health problem in today's world. The early diagnosis of cardiac arrhythmia highly relies on the ECG. Unfortunately, the expert level of medical resources is rare, visually identify the ECG signal is challenging and time-consuming. In this paper we work on datasets belongs to four different arrhythmia databases. In this paper we used different classical machine learning models like Logistic Regression, Random Forest, GBDT, AdaBoost, XGBoost. Of all used models, XGBoost shows an outstanding performance in the overall classification accuracy of 99.01%, precision of 94.94%, recall of 93.86%.

References

[1] Ouelli, Abdelhaq, et al. ”Electrocardiogram features extraction and classification for arrhythmia detection.” Database (SVDB) 2.3 (2015).

[2] Ye, Can, Miguel Tavares Coimbra, and BVK Vijaya Kumar. ”Arrhythmia detection and classification using morphological and dynamic features of ECG signals.” 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology. IEEE, 2010.

[3] Rui Duan, Sabah Mohammed and Jinan Fiaidhi, ”Ensemble Methods For ECG Based Heart-beat Classification”, International Journal of Control and Automation, 12.4 (2019): 29-46.

[4] Shi, Haotian, et al. ”A hierarchical method based on weighted extreme gradient boosting in ECG heartbeat classification.” Computer methods and programs in biomedicine 171 (2019): 1-10.