Texas A&M University

Department of Industrial and Systems Engineering

Engineering Data Analysis (ISEN 613)

Project Report

On

Diagnosis of Cardiac Arrhythmia using ECG readings

Submitted By:

Team 12

Mustafa Panibharwala	(UIN:426006576)	20%
Nithin Pradeep	(UIN: 227001356)	20%
Samarth Mistry	(UIN: 326005991)	20%
Siddharth Ajit	(UIN: 227001842)	20%
Vrutant Shroff	(UIN: 926007672)	20%

Report for: Dr. Na Zou

Contents

Introduction:

Importance:

Even with an annual expenditure of more than $3 trillion, the U.S. healthcare system is far from optimal. For example, the third leading cause of death in the U.S. is preventable medical error. Computer based Decision Support Systems have been proposed to take care of the error to a reasonable extent. However, such systems have not been implemented to utilize the patient data obtained on a daily basis. Another big challenge is in the non-uniformity of diagnosis offered by doctors. The standard deviation in the doctor prescriptions still remains large.

A cardiac arrhythmia is any abnormal heart rate or rhythm. It can be classified into different classes. Some cases of arrhythmia can be critical and only with quick response can the patient reduce risks of complications. The diagnosis of arrhythmia involves handling of huge amount of ECG data which poses the risk of human error in the interpretation of data. If a health care provider failed to diagnose a severe case of arrhythmia, they can even be held liable for medical malpractice. Hence, computer assisted analysis of the ECG data and arrhythmia detection and classification can play a huge role as a decision support system to the doctors. Following are some of the facts which show the importance of the problem:

1. One study showed that up to 23% of misdiagnosed heart attacks were due to the improper reading of a patient's ECG.8
2. One study showed that as compared to electrocardiographers, the physicians misread as normal about 36% of abnormal T waves.9
3. One study showed that out of 1.5 million heart attacks occurring every year in United States, 11000 cases are because of misdiagnoses.7
4. About 4.7 billion dollars paid in 2008 to resolve *all* malpractice claims nationwide.11

Objective:

Machine learning algorithms are extensively used in developing decision support systems in medical field.
Through this project, we intend to use supervised machine learning algorithms to analyse the various features of the electrocardiogram (ECG) of the patients and other physiological characteristics. The Cardiac Arrhythmia data set publicly available on UCI Machine Learning Repository has been utilized for developing the classifiers. The objective of this project to build 2 different models.

a. ***Model 1***- Model for detecting the presence of Arrythmia.
b. ***Model 2***- Model for classifying Arrythmia into different classes.

Scope of Work:

The scope of this project comprises of the following aspects:

1. ***Data cleaning:***
   To replace the garbage values with null values. Further, impute the null values using column mean.
2. ***Feature selection by Boruta package:***
   To find significant predictors to be used for building the models.
3. ***Classification by various statistical learning methods:***
   To employ various supervised learning algorithms for achieving the aforementioned objectives.
4. ***Comparison of results:***
   To compare the results obtained from various models and understand the behaviour of different algorithms, and further use it as an application.
5. Documenting everything in a detailed report

Project Approach:

Data Description:

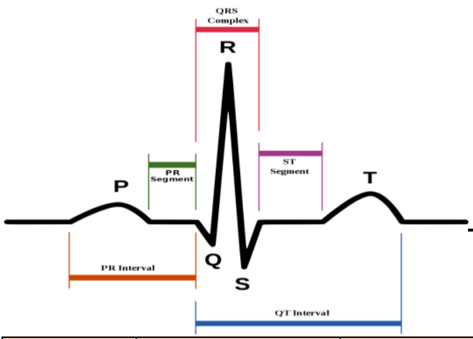
The data belongs to the ECG readings and some of the physical description of 451 patients. There are a total of 279 attributes (206 linear and 73 nominal) and a single output which is categorical. Each of 451 patients are divided into 16 classes based on the value of their attributes.

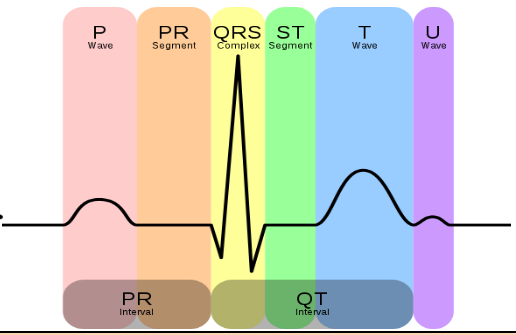
Attributes:

Sr. No.	Attribute	Type	No. of such Attributes	Remarks
1	Age, Weight, Height	Linear	3	Physical Description of patient
2	Sex	Nominal	1	Sex of the patient
3	Duration of wave	Linear	5	Measured in milliseconds
4	Vector Angles	Linear	5	Measured in degrees
5	Heart Rate	Linear	1	No. of heart beats per minute
6	Average width of wave	Linear	60	Measured in milliseconds
7	Number of intrinsic deflections	Linear	12	
8	Existence of ragged wave	Nominal	36	
9	Existence of diphasic derivation	Nominal	36	
10	Amplitude of wave	Linear	96	Multiplied with 0.1*milivolt
11	Areas of all segments divided by 10	Linear	12	Area=width*height/2
12	QRSTA	Linear	12	QRSA + 0.5 * width of T wave * 0.1 * height of T wave.

ECG Wave:

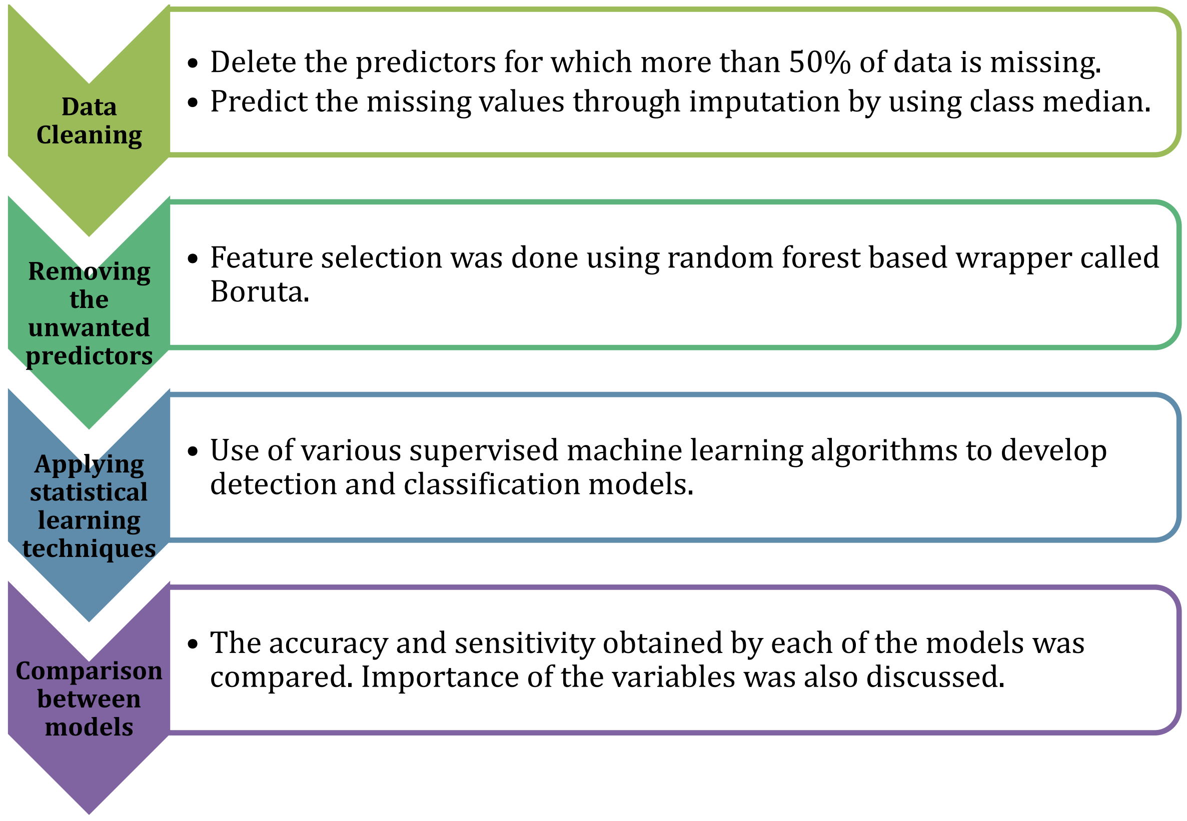
The following diagram shows the ECG wave and the meaning of each wave.[12]





Sr. No.	Predictor	Description
1	P Wave	The P wave represents atrial depolarization.
2	PR interval	The PR interval is measured from the beginning of the P wave to the beginning of the QRS complex.
3	QRS complex	The QRS complex represents ventricular depolarization.
4	J-point	The J-point is the point at which the QRS complex finishes and the ST segment begins.
5	ST segment	It represents the period when the ventricles are depolarized.
6	T wave	The T wave represents ventricular repolarization.

Flowchart:


Data Cleaning
• Delete the predictors for which more than 50% of data is missing.
• Predict the missing values through imputation by using class median.
Removing the unwanted predictors
• Feature selection was done using random forest based wrapper called Boruta.
Applying statistical learning techniques
• Use of various supervised machine learning algorithms to develop detection and classification models.
Comparison between models
• The accuracy and sensitivity obtained by each of the models was compared. Importance of the variables was also discussed.

Reasoning of Approach:

We have implemented models for two purposes.

a. Detection of cardiac arrhythmia

b. Classification of cardiac arrhythmia.

In the model for detection (referred as model 1 in this report), the data points have been classified into two classes- “Normal” & “Arrhythmia”. This model only identifies if the patient is normal (class 1) or suffers from any form of arrhythmia (class 2 to 16).

The model for classification (referred as model 2) classified the patient into one of 16 classes, with class 1 representing normal and classes 2 to 16 representing a condition of cardiac arrhythmia. The arrhythmia class will be treated as the ‘Positive’ class.

The following describes the step-by-step approach for implementing the two models.

Data Cleaning

The data set comprises of 279 attributes of 451 patients. The attributes are as indicated in the section above.

As the first step, the missing values were replaced with null values. Then the columns with same value in all the 451 rows were identified. 17 attributes were found to have a single value for all the data points. As these columns do not explain any variation between the data points and response, they are irrelevant and need not be considered for further analysis. Hence, they were deleted from the data set.

In the next step, we identified the missing values in the data set.

It was observed that five columns had missing values.

Attribute	No. of missing values
J_Angle	375
P_Angle	22
T_Angle	8
QRST_Angle	1
Heart	1

The columns J_Angle and P_Angle were deleted as they had significant number of missing values. The missing values in the remaining columns T_Angle, QRST_Angle and Heart were replaced with the mean value of the corresponding column.

It was observed that the following classes had very few data instances.

Class	**No. of instances**
7	3
8	1
14	4
15	5
16	22

For model 1, this is not a concern as all the above-mentioned classes will be merged together into a single class named “Arrhythmia”.

As far as building model 2 is concerned, the information available is not sufficient to build a model for predicting the low-instance classes. Moreover, for classes represented by very few instances, training the model by cross-validation will fail as sampling may result in a training dataset that does not contain the information of all the classes. Class 16 was deleted as these were unlabelled classes with no specific pattern and were adding noise to the data. We had initially tested the models with class 16 but no model was able correctly predict this class. Thus, we concluded this class had random unclassified information (as its name suggests) and was not adding much information for the model to learn. Therefore, we removed the instances of this class.

Merging Classes:

Model 1: As discussed above, all the instances belonging to classes 2 to 16 were merged to one class.

Sr. No.	**Class**	**Diagnosis**	**Instances**
1	1	Arrhythmia	207
2	2	Normal	245

Model 2: The data set is heavily skewed towards class 1 (normal) and hence, the information provided by the attributes of the limited data points of each arrhythmia classes may not be sufficient to individually distinguish them from the normal class. Hence, physiologically similar arrhythmia conditions were

identified and merged together. This merging of classes enhances the information available for each class making it easier to separate one from the other.

1. Old Anterior (class 3) and Old Inferior (class 4) Myocardial Infarction were treated as single class (Myocardial Infarction)
2. Sinus Tachycardy (class 5) and Sinus Bradycardy (class 6) were treated as single class (Sinus Arrhythmia)
3. Left (class 9) and Right (class 10) Bundle branch block will be treated as single class (Bundle Branch Block)

Output:

Sr. No.	Class	Diagnosis	Instances
1	1	Normal	245
2	2	Ischemic Changes	44
3	3	Myocardial Infarction	30
4	4	Sinus Arrhythmia	38
5	5	Bundle Branch Block	59

Feature Selection

Feature selection is the process of selecting a subset of informative and relevant variables which can be used for building the models. A good feature selection helps in reducing the error due to overfitting, simplifies the models and makes them easier to interpret. For model 1, 61 features were selected. Whereas for model 2, 81 variables were selected.

We did not use PCA as it does not consider predictive ability of features. It just selects linear combinations of variables with high variation. It is well suited for data representations in lower dimensions but may not be always desired for predictions. Also, it reduces the interpretability of the features.

For the purpose of feature selection, we used the Boruta library in R.

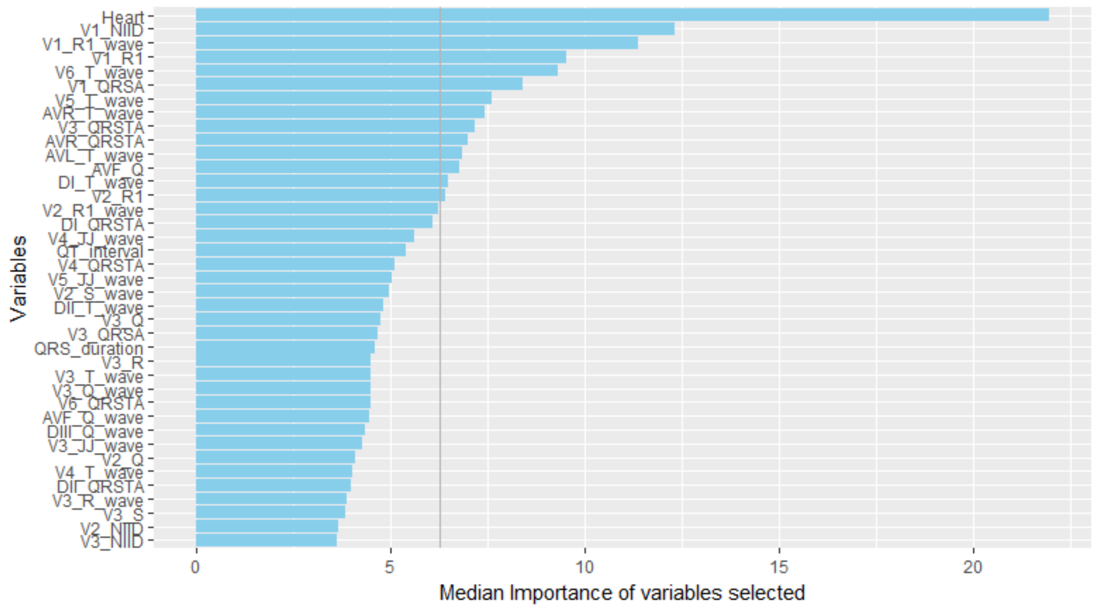
Why Boruta?

Boruta algorithm is a wrapper built around the random forest classification algorithm implemented in the R package randomForest. The trees are independently developed on different bagging samples of the training set. The importance measure of an attribute is obtained as the loss of accuracy of classification caused by the random permutation of attribute values between objects. In short, Boruta is based on the same idea which forms the foundation of the random forest classifier, namely, that by adding randomness to the system and collecting results from the ensemble of randomized samples one can reduce the misleading impact of random fluctuations and correlations. In the Boruta package Z score is used as the importance measure because it accounts the fluctuations in mean accuracy loss among the trees.

Boruta follows an all relevant feature search where it heuristically selects a variable which has high relevance with the response and low correlation with other features. It is different from traditional feature selection methods because it does not select a subset which minimizes error in the response by minimal optimal method. Instead it removes the features which are not relevant to the response and divides the relevant features into strongly relevant and weakly relevant based on Z score.

However, we cannot use the Z score calculated in Random Forest as a direct measure of variable importance as this Z score is not directly related to the statistical significance of the variable importance. This makes Boruta well suited for biomedical applications where one might be interested to determine which features are connected in some way to a particular medical condition (target variable).

The feature importance obtained from Boruta is as follows. As can be seen, heart beat rate is the most important variable which makes sense.



Implementation of models

Based on the relevant features selected by Boruta package, the models were built using the following algorithms. We have used cross-validation on the training data set to arrive at the best model. Test and train data were split in the ratio of 25:75. Validation set approach was not used as it has high variance.

1. Logistic regression
2. LDA
3. QDA
4. KNN
5. Support Vector Machine
6. Support Vector Classifier
7. Decision Tree
8. Random Forest
9. XgBoost (regularized GBM)
10. Neural Networks

The details of implementation are discussed in the next section.

QDA could not be implemented for the classification model as certain classes had too less instances in comparison with the number of attributes.
The details of implementation are discussed in the next section.

Implementation Details:

We have applied 10 following statistical learning methods for both the Approaches. Their description, tables, plots and diagnosis is also described for easy understanding and interpretation of the results.

LDA:[13]

Type: *Parametric*
Can be used in: *Classification*
Used for: *pattern recognition*

LDA is similar to Bayes classification, however the distribution of data points is considered as normal or Gaussian distribution, which is not the case always. Also the variance of each class is considered the same in LDA classification.

As can be seen above, LDA performs poorly in detecting the presence of arrhythmia. The very low sensitivity of the model (0.5806) implies that the model gives the undesirable output of classifying a considerable proportion of arrhythmia patients to normal. Since the model has a high specificity of 0.8039, reducing the threshold for the arrhythmia class can give a better and acceptable trade-off between sensitivity and specificity.

Model – 1

CONFUSION MATRIX

		Acutal Class	
		Arrhythmia	Normal
Predicted Class	Arrhythmia	36	10
	Normal	26	41

TEST PERFORMANCE MEASURES

Accuracy	68.14%
Sensitivity	0.5806
Specificity	0.8039

Model – 2

Cross-validated LDA model was applied on the multi-class test data set. We can clearly see from the confusion matrix that LDA is able to predict Class 1 and Class 3 with high accuracy, but performs poorly in classifying class 2, 4 & 5.

As far as classes with low sensitivity (2, 4 & 5) are concerned, it can be seen from the confusion matrix that majority of misclassified observations have been classified to class 1 (Normal). This is particularly undesirable as this would mean a considerable proportion of the people suffering from Arrhythmia are classified as normal.

Overall AUC of LDA is 0.6799 which means the weighted average of the efficiency over all the classes to rightly classify the observations into their respective classes is only 0.6799, which is very low.

CONFUSION MATRIX

	Actual class					
Predicted class	■	1	2	3	4	5
	1	51	6	1	6	8
	2	3	5	0	0	0
	3	1	1	5	0	1
	4	3	1	0	2	1
	5	1	0	0	0	8

	Class 1	Class 2	Class 3	Class 4	Class 5
Sensitivity	0.86	0.38	0.83	0.25	0.44
Specificity	0.53	0.97	0.97	0.95	0.98
Precision	0.70	0.63	0.63	0.29	0.88
Recall	0.86	0.38	0.83	0.25	0.44
F1	0.78	0.48	0.71	0.27	0.59
Balanced accuracy	0.70	0.68	0.90	0.60	0.71

Note on poor performance of LDA:

LDA works on the assumption that the distribution of the predictors in each class is approximately Gaussian. It is also assumed that the predictors share a common variance across all the classes. Since this assumption may not hold true for this data set, LDA may not perform well for detection or classification. Also, it maybe noted that LDA is a linear classifier. Since we have a large number of features through boruta package that have a strong relationship with the response, accommodating the effect of all these features through a linear classifier is difficult.

Logistic Regression:[14]

Type: *Parametric*
Can be used in: *Classification*
Used for: *predicting outcomes of categorically distributed dependent variable*

Multinomial Logistic Regression is a classification method and is similar to logistic regression. However it is a more generalize model and it is used when there is more than two possible discrete outcomes.

Model – 1

Logistic regression model with both ridge and lasso regularization were applied on the training set. Model with higher accuracy was obtained with ridge regularization. This was because we had already selected predictors which had high correlation with the response through Boruta feature selection. This violates the implicit assumption of lasso that some predictors are insignificant to the response variable. Thus, logistic regression with ridge regularization performed better. The optimal value of tuning parameter lambda for ridge regularization was found to be 0.14415.

CONFUSION MATRIX ***TEST PERFORMANCE MEASURES***

		Acutal Class	
		Arrhythmia	Normal
Predicted Class	Arrhythmia	46	5
	Normal	16	46

Accuracy	81.42%
Sensitivity	0.7419
Specificity	0.902

Though the logistic regression does a good as far as accuracy is concerned, 16 arrhythmia cases have been classified as normal, which is undesirable. This can however be improved by reducing the threshold from 0.5 as the objective of the model is only to detect the presence of arrhythmia.

Model – 2

Multinomial logistic regression model was applied to the data set to classify the data points into 5 classes. The model gave a very poor test accuracy of 63.46%. As maybe noted from the confusion matrix and test performance measures below, the model performs very poorly in classifying class 4 correctly. The only class the model predicts with a reasonable accuracy is class 1.

It may be noted that for multiple classes, discriminant analysis performs better and multinomial logistic regression is not used often. Also, when the number of instances in certain classes is relatively low, logistic regression tends to be unstable.

CONFUSION MATRIX   TEST PERFORMANCE MEASURES

	Actual class							Class 1	Class 2	Class 3	Class 4	Class 5
Predicted class	■	1	2	3	4	5	Sensitivity	0.64	0.46	0.83	0.37	0.44
	1	38	4	1	4	2	Specificity	0.75	0.82	0.92	0.91	0.97
	2	10	6	0	0	6	Precision	0.77	0.27	0.41	0.27	0.8
	3	4	2	5	1	0	Recall	0.64	0.46	0.83	0.37	0.44
	4	6	0	0	3	2	F1	0.7	0.34	0.55	0.31	0.57
	5	1	1	0	0	8	Balanced accuracy	0.56	0.12	0.05	0.07	0.17

SVC:[15]

Type: *Non-Parametric*
Can be used in: *Classification and regression*
Used for: *Improving the prediction accuracy of a maximal margin classifier by implementing soft margin.*

Support vector Classifier uses soft margin to take into account the variance in the training data, whereas in Maximal margin classifier we use hard margin and no violations are allowed.

Model - 1

SVM with radial basis function kernel was used to classify the data points. Upon performing grid search on gamma and cost using a 5-fold cross validation on training data set, the optimal parameters were found to be gamma= 0.001953125 and cost=2. The model gave a reasonably high accuracy of 80.53%. However, the sensitivity is relatively low at 0.7581. 207 data points of training data form the support vectors. Since there are multiple classes of arrhythmia with distinct features scattered around in the feature space, a radial separating hyperplane may fail to separate all of them from the distinct 'normal' class. Hence, SVM with radial basis function kernel cannot club all the arrhythmia classes and separate them from the single normal class.

CONFUSION MATRIX   ***TEST PERFORMANCE MEASURES***

		Acutal Class	
		Arrhythmia	Normal
Predicted Class	Arrhythmia	36	10
	Normal	26	41

Accuracy	80.53%
Sensitivity	0.7581
Specificity	0.8627

Model – 2

We selected an RBF kernel as the first choice because it can handle cases when the relationship between class labels and attributes is non-linear. This kernel non-linearity maps the samples into a higher dimensional space enabling it to handle the non-linear relationships.

We used radial basis function/ gaussian kernel with gamma= 0.0078125 and cost= 16. We performed a 'grid search' on gamma and cost using cross validation. Various values of gamma and cost were tried and the one with the best cross validation accuracy (gamma= 0.0078125 and cost= 16) was picked. We tried over an exponentially growing sequence of cost and gamma where the range of cost was 2-5 to 215 and that of gamma was from 2-15 to 23. We have used this naïve straightforward grid search approach for cross validation for two reasons. First, computational time required to find good parameters by grid search is not as high as other advanced methods (since there are only two parameters, cost and gamma). Second, grid search can be easily parallelized.

CONFUSION MATRIX

	Actual class							Actual class							Class 1	Class 2	Class 3	Class 4	Class 5
Predicted class	■	1	2	3	4	5	Predicted class	■	1	2	3	4	5	Sensitivity	0.85	0.46	0.83	0.13	0.67
	1	50	5	1	6	6		1	50	5	1	6	6	Specificity	0.60	0.97	0.96	0.98	0.97
	2	3	6	0	0	0		2	3	6	0	0	0	Precision	0.74	0.67	0.56	0.33	0.80
	3	2	1	5	1	0		3	2	1	5	1	0	Recall	0.85	0.46	0.83	0.13	0.67
	4	2	0	0	1	0		4	2	0	0	1	0	F1	0.79	0.55	0.67	0.18	0.73
	5	2	1	0	0	12		5	2	1	0	0	#	Balanced accuracy	0.72	0.71	0.90	0.55	0.82

We find that SVM with RBF kernel cannot predict class 4 with high sensitivity. Majority of the misclassifications have been classified to class 1 which is particularly undesirable.

We have observed that the cross-validated accuracy is 81% and test accuracy is 71.15%. Since there is a huge difference between the training and test accuracy, it can be concluded that the RBF model is overfitting the data.

It is also observed that there are 223 support vectors in this SVM model which is approximately 71% of the training data. This shows that our model has a low variance and a high bias which is the reason for low test accuracy in comparison to training accuracy.

KNN (K-Nearest Neighbours):[16]

Type: *Non Paramteric*
Can be used in: *Classification and regression*
Used for: *Pattern Recognition*

For any positive value of K, the method attempts to find K nearest points to the test data point and assigns the class to it on basis of majority for K nearest points (assigns the value of mean of K nearest points in case of regression).

Model – 1

KNN has a very poor sensitivity and very good specificity as classification of normal case is quite accurate KNN uses a small number of neighbors here i.e. 3 and small enough threshold to yield a complex classifier favoring the normal class.

CONFUSION MATRIX ***TEST PERFORMANCE MEASURES***

		Acutal Class	
		Arrhythmia	Normal
Predicted Class	Arrhythmia	32	7
	Normal	30	44

Accuracy	67.25%
Sensitivity	0.516
Specificity	0.862

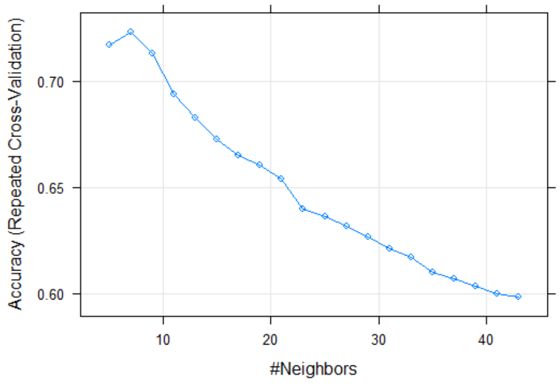
Model – 2:

As seen above, KNN cannot accurately classify test observations for classes 2 and 5 whereas, it completely misclassifies them for class 4. Most of the misclassified observations have been classified to class 1.

	Actual class					
Predicted class	■	1	2	3	4	5
	1	56	5	1	8	11
	2	2	6	0	0	0
	3	1	1	5	0	0
	4	0	0	0	0	0
	5	0	1	0	0	7

	Class 1	Class 2	Class 3	Class 4	Class 5
Sensitivity	0.95	0.46	0.83	0.00	0.39
Specificity	0.44	0.98	0.98	1.00	0.99
Precision	0.69	0.75	0.71	NA	0.88
Recall	0.95	0.46	0.83	0.00	0.39
F1	0.80	0.57	0.77	NA	0.54
Balanced accuracy	0.70	0.72	0.91	0.50	0.69

KNN is a non-parametric method and makes no assumptions. It classifies a test observation to the class that is most common among its neighbors, which in this case is class 1 since the dataset is imbalanced and skewed towards class 1. Hence, a considerable number of patients who suffer from arrhythmia will not be correctly diagnosed if we follow this model.



We find the optimal K that achieves the best accuracy is 7, as seen from the graph, by performing repeated cross-validation. We use 20 different values to try for each parameter here.

Overall AUC of KNN is 0.7097 which means the weighted average of efficiency over all the classes to rightly classify observations into their respective classes is only 0.7097.

SVM:[15]

Type: *Non-Parametric*
Can be used in: *Classification and regression*
Used for: *Improving the prediction accuracy of a maximal margin classifier and support vector classifier by use of various type of non-linear kernels (radial and polynomial).*

Support vector Classifier uses soft margin to take into account the variance in the training data, however Support Vector Machine is used to classify non-linear classes. It does so by expanding the predictor space by use of various type of Kernels.

In our report we have used 3 kernels. 1. Linear 2. Radial and 3. Polynomial

Model – 1

SVM with radial basis function kernel was used to classify the data points. Upon performing grid search on gamma and cost using a 5-fold cross validation on training data set, the optimal parameters were found to be gamma= 0.001953125 and cost=2. The model gave a reasonably high accuracy of 80.53%. However, the sensitivity is relatively low at 0.7581. 207 data points of training data form the support vectors. Since there are multiple classes of arrhythmia with distinct features scattered around in the feature space, a radial separating hyperplane may fail to separate all of them from the distinct ‘normal’ class. Hence, SVM with radial basis function kernel cannot club all the arrhythmia classes and separate them from the single normal class.

CONFUSION MATRIX

		Acutal Class	
		Arrhythmia	Normal
Predicted Class	Arrhythmia	36	10
	Normal	26	41

TEST PERFORMANCE MEASURES

Accuracy	80.53%
Sensitivity	0.75
Specificity	0.86

Model - 2

We selected an RBF kernel as the first choice because it can handle cases when the relationship between class labels and attributes is non-linear. This kernel non-linearity maps the samples into a higher dimensional space enabling it to handle the non-linear relationships.

We used radial basis function/ gaussian kernel with gamma= 0.0078125 and cost= 16. We performed a 'grid search' on gamma and cost using cross validation. Various values of gamma and cost were tried and the one with the best cross validation accuracy (gamma= 0.0078125 and cost= 16) was picked. We tried over an exponentially growing sequence of cost and gamma where the range of cost was 2-5 to 215 and that of gamma was from 2-15 to 23. We have used this naïve straightforward grid search approach for cross validation for two reasons. First, computational time required to find good parameters by grid search is not as high as other advanced methods (since there are only two parameters, cost and gamma). Second, grid search can be easily parallelized.

CONFUSION MATRIX

	Actual class							Class 1	Class 2	Class 3	Class 4	Class 5
Predicted class	■	1	2	3	4	5	Sensitivity	0.85	0.46	0.83	0.13	0.67
	1	50	5	1	6	6	Specificity	0.60	0.97	0.96	0.98	0.97
	2	3	6	0	0	0	Precision	0.74	0.67	0.56	0.33	0.80
	3	2	1	5	1	0	Recall	0.85	0.46	0.83	0.13	0.67
	4	2	0	0	1	0	F1	0.79	0.55	0.67	0.18	0.73
	5	2	1	0	0	12	Balanced accuracy	0.72	0.71	0.90	0.55	0.82

We find that SVM with RBF kernel cannot predict class 4 with high sensitivity. Majority of the misclassifications have been classified to class 1 which is particularly undesirable.

We have observed that the cross-validated accuracy is 81% and test accuracy is 71.15%. Since there is a huge difference between the training and test accuracy, it can be concluded that the RBF model is overfitting the data.

It is also observed that there are 223 support vectors in this SVM model which is approximately 71% of the training data. This shows that our model has a low variance and a high bias which is the reason for low test accuracy in comparison to training accuracy.

Random Forest:[17]

Type: *Ensemble*
Can be used in: *Classification, regression and other tasks(by constructing multitude of decision trees)*
Used for: *reducing correlation between predictors*

In general, trees which are grown to more depth tend to suffer from over fitting (low bias and high variance). So Random Forest works sin a way similar to bagging of trees, but it only takes a subset of predictors for forming single tree and then take average of all uncorrelated trees so as to reduce variance.

Model – 1

Random forest does a very good job in detecting the presence of arrythmia. 10-fold cross validation was used to fit the model. The resulting best model gave a test accuracy of 83.19%. The test performance measures and confusion matrix are as given below. It can be seen that the model has a similar sensitivity and specificity measure.

CONFUSION MATRIX

		Acutal Class	
		Arrhythmia	Normal
Predicted Class	Arrhythmia	51	8
	Normal	11	43

TEST PERFORMANCE MEASURES

Accuracy	83.19%
Sensitivity	0.8225
Specificity	0.8431

Since the primary goal of this model is to detect the presence of cardiac arrhythmia, a higher sensitivity at the cost of reduced specificity maybe acceptable. A reduced threshold of 0.30 gave the following output.

CONFUSION MATRIX

		Acutal Class	
		Arrhythmia	Normal
Predicted Class	Arrhythmia	56	14
	Normal	6	37

TEST PERFORMANCE MEASURES

Accuracy	82.30%
Sensitivity	0.9032
Specificity	0.7254

The model increased the sensitivity from 0.8225 to 0.9032 with a small and acceptable reduction in accuracy and specificity.

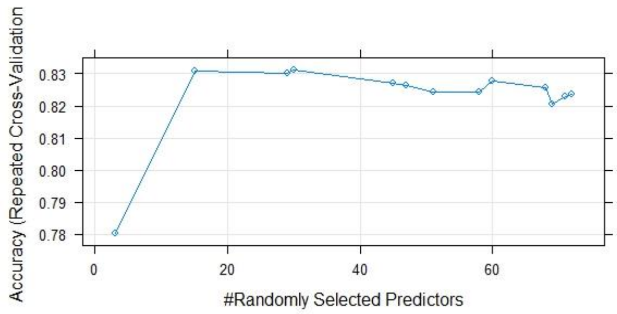
***Model** – 2*

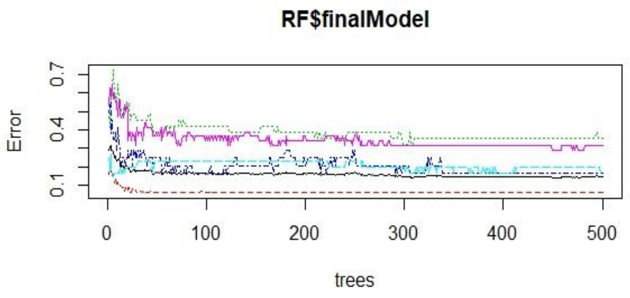
A 5-fold cross validation was performed on training set to obtain a random forest classification model. We cross-validated on number of trees and mtry (number of randomly selected predictors at each split) to find the best model with least cross-validated error. The best model had mtry as 34 and number of trees as 500. We found that Random Forest algorithm is most efficient compared to other models in classifying Arrhythmia into four classes. The cross-validated accuracy of the best model obtained through cross-validation was 84.48%.

This is the only classifier that gives a reasonable sensitivity in detecting class 2 and class 4. Random forest classifier also has the overall highest accuracy (Test accuracy of 80.77%). This is because random forest does not suffer from high variance. So even though we have a skewed data set with high number of variables and low number of data points, random forest is able to predict the classes with a reasonably well accuracy. Also, by imparting the randomness in splitting and generating dissimilar trees, random forest tackles the problem of having a single tree which as we saw suffered from the problem of many terminal nodes having class 1 as the majority class.

CONFUSION MATRIX

	Actual class							Class 1	Class 2	Class 3	Class 4	Class 5
Predicted class		1	2	3	4	5	Sensitivity	0.88	0.54	0.67	0.75	0.67
	1	52	3	1	2	4	Specificity	0.78	0.98	0.99	0.93	0.97
	2	2	7	0	0	0	Precision	0.84	0.78	0.80	0.46	0.80
	3	0	1	4	0	0	Recall	0.88	0.54	0.67	0.75	0.67
	4	3	1	1	6	2	F1	0.86	0.64	0.73	0.57	0.73
	5	2	1	0	0	12	Balanced accuracy	0.83	0.76	0.83	0.84	0.82





From the graph it can be seen that the accuracy of the model is maximum while considering 34 random predictors at every split.

The final model of random forest has 500 trees at which the error rate stabilizes.

Decision Trees:[18]

Type: *Ensemble*
Can be used in: *Classification and regression*
Used for: *Data Mining*

Decision tress are used to create the model that predicts the value of a target variable based on several input variables. It can be used to visually and explicitly represent decision and decision making.
Model – 1

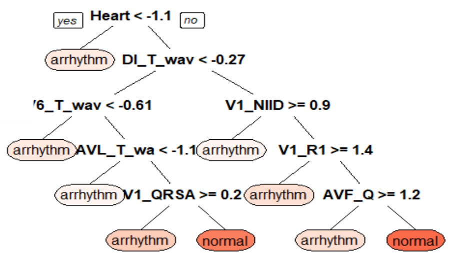
The decision tree was fitted on the binary class. The best split is obtained based on Gini index. Upon cross-validation over the training set to fit the model, it was found that the lowest misclassification error occurred at a tree depth of 9. Hence, the tree was pruned to obtain the sub-tree having 9 terminal nodes. As can be seen from the confusion matrix and test performance measures, the decision tree performs very poorly in identifying the arrhythmia classes.

CONFUSION MATRIX

		Acutal Class	
		Arrhythmia	Normal
Predicted Class	Arrhythmia	39	7
	Normal	23	44

TEST PERFORMANCE MEASURES

Accuracy	73.45%
Sensitivity	0.6209
Specificity	0.8627

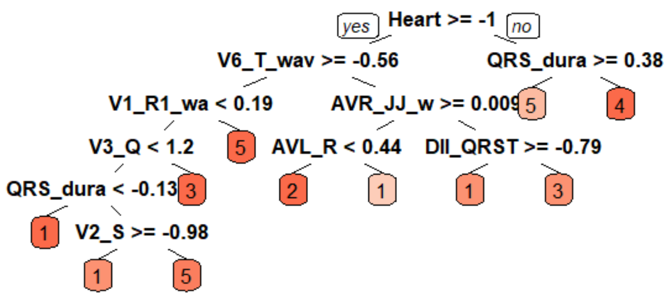


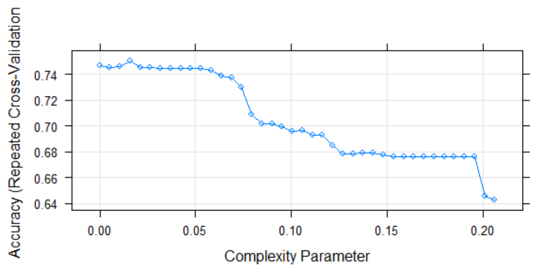
Model – 2

Decision trees for multiple class also have a low test accuracy of 69.23% and very low sensitivity for classes 2 and 5. We attribute the low value of accuracy and sensitivity to high variance of the data. The tree classifies the observations based on the majority class at each terminal node. Since the data set is heavily skewed towards class 1, there will be a considerable number of terminal nodes that classify the observation to class 1. This could explain the reason for the observations of class 2-5 being misclassified to class 1.

Predicted class	Actual class					
		1	2	3	4	5
	1	50	6	0	2	6
	2	2	3	2	0	1
	3	1	2	3	0	0
	4	3	1	1	6	1
	5	3	1	0	0	10

	Class 1	Class 2	Class 3	Class 4	Class 5
Sensitivity	0.84	0.23	0.5	0.75	0.55
Specificity	0.68	0.94	0.96	0.93	0.95
Precision	0.78	0.37	0.5	0.5	0.71
Recall	0.84	0.23	0.5	0.75	0.55
F1	0.81	0.28	0.5	0.6	0.62
Balanced accuracy	0.76	0.58	0.73	0.84	0.75





This is snippet of the fitted decision tree. Splitting criteria for decision tree is kept as Information gain. We first grew a full decision tree and pruned it based on maximum information gain. We have cross-validated over a range of values of Cp to calculate the best possible spilt. Here Cp value indicates the minimum improvement needed in the model at each node. We have pruned the tree based on the Cp value. The best model has a Cp value of 0.01587302. This value acts a stopping parameter. It speeds up the search for splits because it can identify splits that don't meet these criteria and prune them before going too far.

XG-Boost:[20]

Type: *Ensemble*
Can be used in: *Classification and regression*
Used for: *Improving the predicting accuracy of a boosted model*
Similar to Gradient Boosting in principle, xgboost uses a more regularized model formalization so as to control over-fitting.
Model – 1
Extreme gradient boosting does a very good job in detecting the presence of arrythmia.

CONFUSION MATRIX

		Acutal Class	
		Arrhythmia	Normal
Predicted Class	Arrhythmia	47	4
	Normal	15	47

TEST PERFORMANCE MEASURES

Accuracy	83.19%
Sensitivity	0.7581
Specificity	0.9216

The area under the curve for binary classification model is 0.8398

Model – 2

Predicted class	Actual class					
	■	1	2	3	4	5
	1	53	3	1	2	4
	2	1	8	0	0	0
	3	1	1	4	0	0
	4	3	0	1	6	1
	5	1	1	0	0	13

	Class 1	Class 2	Class 3	Class 4	Class 5
Sensitivity	0.90	0.62	0.67	0.75	0.72
Specificity	0.78	0.99	0.98	0.95	0.98
Precision	0.84	0.89	0.67	0.55	0.87
Recall	0.90	0.62	0.67	0.75	0.72
F1	0.87	0.73	0.67	0.63	0.79
Balanced accuracy	0.84	0.80	0.82	0.85	0.85

We can see from above that extreme gradient boosting classifies the test observations almost correctly with a test accuracy of 84%.

The algorithm applies regularization in addition to boosting by setting parameters like learning rate, subsampling and L2 term on weights to prevent overfitting by reducing variance. We select optimal values of hyperprameters by using a randomized search. Running grid search would be very expensive as we tune seven hyperparameters that govern the tree architecture.

There is relatively high specificity and low sensitivity for all classes except class 1 i.e the model correctly identifies those observations that do not belong to these classes. The sensitivity for classes 2 & 3 is low, classes 4 & 5 is moderate while class 1 is high. Sensitivity refers to true positive rate which is highly desirable in cardiac arrhythmia prediction.

Neural Networks:[21]

Neural network is a machine learning algorithm that can perform supervised and unsupervised learning tasks. However, in our project, neural network was employed for classification in model 1 and 2. A neural network tries to mimic the functioning of neurons in the human brain. Neural network comprises of an input, output and hidden layers. Each of these layers have processing units referred as neurons. The input(predictors) are fed to the neurons, weights are assigned to the inputs to minimize a predefined loss function, in classification often the misclassification rate. This loss function is minimized using nonlinear optimization models. In this project, library caret, nnet was used in implementing the neural network model. The test dataset was cross-validated for best model selection based on classification accuracy.

Model - 1:

CONFUSION MATRIX

		Acutal Class	
		Arrhythmia	Normal
Predicted Class	Arrhythmia	45	8
	Normal	17	43

TEST PERFORMANCE MEASURES

Accuracy	77.87%
Sensitivity	0.725
Specificity	0.8431

The optimized hyperparameter values are depth = 7 and decay = 0.7. Depth refers to the number of total layers in the model including input and output layers. Decay is a parameter of the optimization algorithm used in the NN model. Neural network does a decent job in predicting arrhythmic class with a test accuracy of 77.8 and sensitivity of arrhythmic class is 0.725.

CONFUSION MATRIX

	Actual class					
Predicted class	■	1	2	3	4	5
	1	52	4	1	7	5
	2	1	6	0	0	1
	3	2	1	5	0	0
	4	2	0	0	1	2
	5	2	2	0	0	10

	Class 1	Class 2	Class 3	Class 4	Class 5
Sensitivity	0.88	0.46	0.8	0.83	0.12
Specificity	0.62	0.97	0.96	0.95	0.95
Precision	0.75	0.75	0.62	0.2	0.71
Recall	0.88	0.46	0.83	0.12	0.55
F1	0.81	0.57	0.71	0.15	0.62
Balanced accuracy	0.75	0.71	0.9	0.54	0.75

The final values used for the multi-classification model were depth = 7 and decay = 0.5. Here, we get the accuracy of 71.15%. The sensitivity of classes 1, 3 and 4 is good but that of class 2 and 4 is poor. Whereas, the specificity for class 1 is low.

QDA:[22]

Type: *Parametric*
Can be used in: *Classification*
Used for: *pattern recognition*

Similar to LDA in principle but QDA will not assume that the data is normally distributed and the covariance of each class is not assumed to be identical.

Model – 1

QDA is an alternative approach to LDA assuming that every class comes from a Gaussian Distribution having its own covariance matrix. QDA has a non-linear decision boundary and thus performs better than LDA when performing a binary classification. We can see a considerable improvement in accuracy over LDA because of a non-linear decision boundary. QDA has a reasonable high-test accuracy of 79.65% and a reasonably high sensitivity. It has misclassified 17 arrhythmia cases as normal which is undesirable. Overall the model has an AUC of 0.8041 i.e. weighted average of the efficiency of the overall classes to rightly classify the observation into their respective classes is 0.8041 which is a great improvement over LDA and other linear classifiers.

CONFUSION MATRIX

		Acutal Class	
		Arrhythmia	Normal
Predicted Class	Arrhythmia	45	6
	Normal	17	45

TEST PERFORMANCE MEASURES

Accuracy	79.65%
Sensitivity	0.7258
Specificity	0.8824

Model – 2

We were not able to apply QDA to a multiclass classification problem because we have more variables than the class having the minimum number of observations. Thus, an alternative was LDA or reducing the number of features. Therefore we tried other algorithms.

Comparison:

The below Table and Plot shows the accuracy obtained by different approaches used by the team and the improvement gained by used different methods.

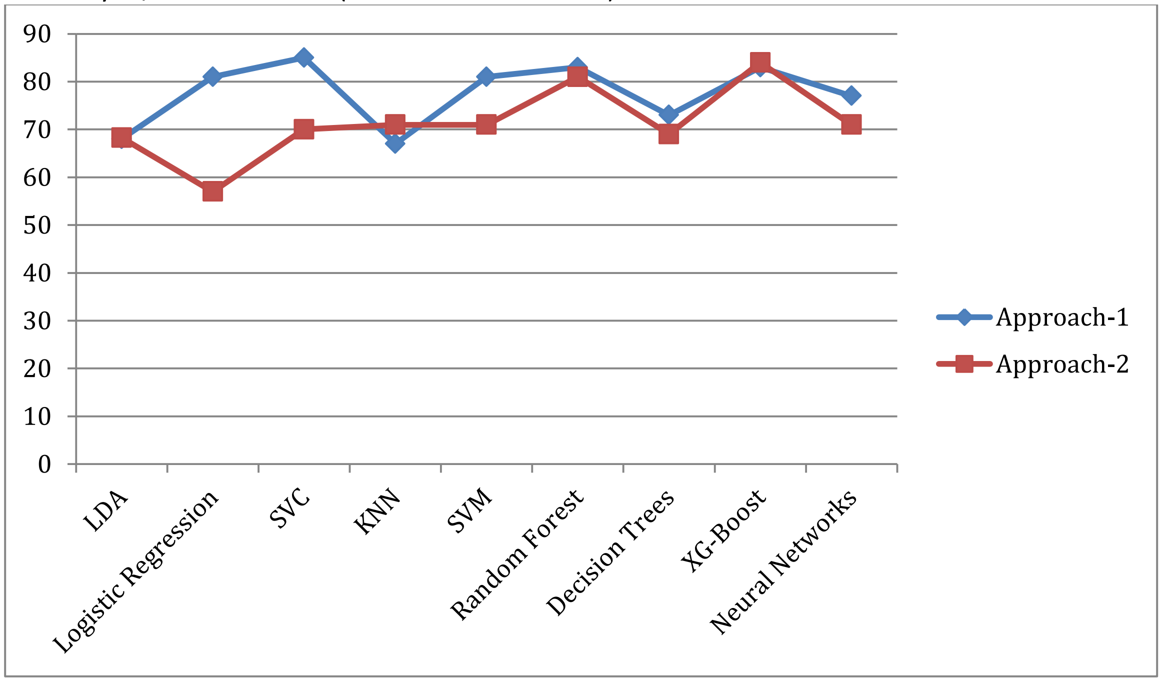
Table:

Here we show the table for showing the accuracy and test MSE for each method

Sr. No.	Algorithm	Accuracy(Model-1)	Accuracy(Model-2)	Remarks
1	LDA	68	68.26	
2	Logistic Regression	81,79.64	57	Ridge, Lasso
3	SVC	85	70	
4	KNN	67	71	
5	SVM	81	71	
6	Random Forest	83	81	
7	Decision Trees	73	69	
8	XG-Boost	83	84	
9	Neural Networks	77	71	
10	QDA	80		

Plot:

Accuracy V/s Method used (Model-1 and Model-2)



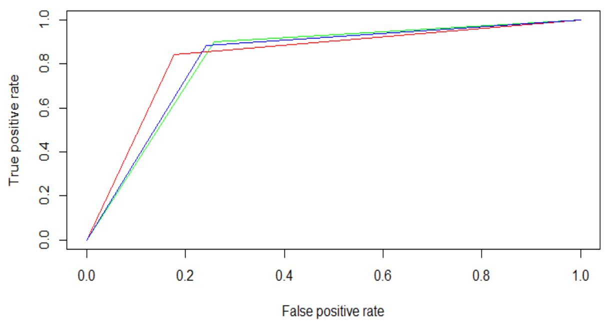
Interpretation:

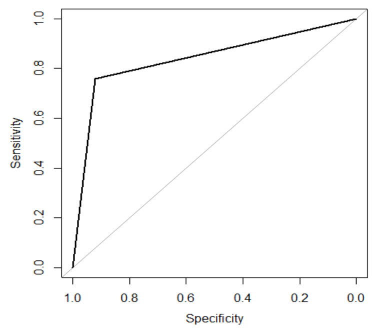
- In general, model 1(model for detection) performs better in terms of accuracy and sensitivity, the critical parameters. This could be because the initial data set upon which we built the classification model was heavily skewed towards class 1. By merging the classes, the distribution of classes becomes almost even making it easier to separate the classes with the available information. Certain classes in model 2(model for classification) have very few instances making it difficult to predict them.

- All the models classify class 1 (normal) with reasonable accuracy. This could be because more than 50% of the instances belong to class 1 and hence sufficient information is available for the model to learn its features.
- Majority of the misclassified instances of class 2 in many models have been classified to class 1(normal) which is highly undesirable. This could be because there may not be strong features that could distinguish this class from Class 1.
- We have observed that many cases of class 2 arrhythmia have been misclassified into the normal class. Hence, it may be noted that there is only a small margin separating the normal class from class 2 arrhythmia.
- Class 3 has the least number of instances. But almost all the models classify those instances with comparatively high sensitivity. So it is reasonable to conclude that class 3 (Myocardial Infarction), has very distinct features which make it easier for the model to identify them.
- In the classification model (model 2), it is observed that only Random forest and XgBoost are able to classify class 4 with high accuracy (75% for both). This could be because, unlike the other algorithms, the random forest and Xgboost reduces the variance by averaging the output from ensemble models.
- Since, a large number of predictors are used to fit a model on a very small number of observations, the models suffer from high variance. Random Forest and XgBoost are exceptions.

Model 1:

In the end, we see that the highest accuracy is 85% for Support Vector Classifier followed by Tree models: Extreme Gradient Boosting Tree and Random Forest (83% and 82% resp.). We get the highest sensitivity of 0.82 for Random Forest which increases to 0.90 by lowering the threshold to 0.3.





ROC 1 ROC 2

The ROC curve for the best performing algorithms is shown where:
ROC 1: Red <- Logistic Regression with Ridge Regression (AUC=0.8219),
Green <- Random Forest (AUC=0.8320),
Blue <- SVM-Radial (AUC=0.8202),
ROC 2: Black <- Extreme Gradient Boosting Trees (AUC=0.8398)

Model 2:

Here, the highest accuracy is 84% for Extreme Gradient Boosting Tree and 81% for Random Forest. We average the sensitivity for the arrhythmic classes where the highest value is 0.69 for Extreme Gradient Boosting Tree followed by 0.82 for Random Forest. We also observe that in tree-based models, work well for model 1 and model 2.

Model Comparison:

Sr .No	Model	Accuracy	Sensitivity	
1	**Logistic regression**			
1.a	Model 1	0.81	0.74	
1.b	Model 2	0.57	Class 1	0.64
			Class 2	0.46
			Class 3	0.83
			Class 4	0.37
			Class 5	0.44
2	**LDA**			
2.a	Model 1	0.6814	0.5806	
2.b	Model 2	0.6826	Class 1	0.86
			Class 2	0.38
			Class 3	0.83
			Class 4	0.25
			Class 5	0.44
3	**QDA**			
3.a	Model 1	0.80	0.73	
3.b	Model 2	NOT APPLICABLE		
4	**KNN**			
4.a	Model 1	0.67	0.51	
4.b	Model 2	0.71	Class 1	0.95
			Class 2	0.46
			Class 3	0.83
			Class 4	0.00
			Class 5	0.39
5	**Decision Tree**			
5.a	Model 1	0.73	0.62	
5.b	Model 2	0.69	Class 1	0.84
			Class 2	0.23
			Class 3	0.5
			Class 4	0.75
			Class 5	0.55

Sr .No	Model	Accuracy	Sensitivity	
6	**Random Forest**			
6.a	Model 1	0.83	0.82	
6.b	Model 2	0.81	Class 1	0.88
			Class 2	0.53
			Class 3	0.67
			Class 4	0.75
			Class 5	0.67
7	**SVM**			
7.a	Model 1	0.81	0.76	
7.b	Model 2	0.71	Class 1	0.85
			Class 2	0.46
			Class 3	0.83
			Class 4	0.12
			Class 5	0.67
8	**SVC**			
8.a	Model 1	0.85	0.7581	
8.b	Model 2	0.7	Class 1	0.86
			Class 2	0.46
			Class 3	0.66
			Class 4	0.12
			Class 5	0.61
9	**XG-Boost**			
9.a	Model 1	0.83	0.76	
9.b	Model 2	0.84	Class 1	0.89
			Class 2	0.62
			Class 3	0.67
			Class 4	0.75
			Class 5	0.72
10	**Neural Network**			
10.a	Model 1	0.77	0.72	
10.b	Model 2	0.71	Class 1	0.88
			Class 2	0.46
			Class 3	0.83
			Class 4	0.12
			Class 5	0.55

Executive Summary

Objective

To objective of this project was to analyze the Cardiac Arrhythmia dataset obtained from University of California Irvine Machine Learning Repository, so as to detect if the patient is suffering from Arrhythmia or not (Model-1) and to correctly classify the patient in 1 of 5 classes of Cardiovascular Arrhythmias (Model-2) on basis of their ECG readings.

Importance

Heart failure is a chronic, progressive condition in which the heart muscle is unable to pump enough blood through to meet the body's needs for blood and oxygen. Basically, the heart can't keep up with its workload. About 5.7 million Americans are living with it today. An electrocardiogram (ECG) is used to detect arrhythmia. It may happen that if wrongly classified, the results may lead to a fatality.
As team 12, we have tried to train the dataset using various statistical learning techniques like Logistic Regression, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Random Forest, K-Nearest Neighbours, Support Vector Machines and ensemble of methods like and Neural Networks. For each of the technique used, plots and table are shown in the report and their diagnosis is made.

Gap in Literature

We have developed two models, one for detection of Arrhythmia (model 1) and another for classification of Arrhythmia into 5 classes (model 2). This can potentially be used as an iterative method to eliminate normal cases (class 1) first and then using the classification model (model 2) strictly for classifying only the Arrythmia cases into its specific class. However, with the limited number of data instances, we have developed the two models independently without eliminating the normal class. We have not encountered such a two-model decision system to the best of our knowledge.
Feature selection was done using a random forest based wrapper algorithm called Boruta which gives the exact number of significant features. This is a relatively new approach for feature selection.
We have also used XgBoost (Regularized Gradient Boosted trees) and neural networks for our classification.

Implications of Results in Terms of Objectives we had set

We have observed that upon adjusting the threshold probability for Random Forest model for detection (model 1), we were able to detect the presence of Arrhythmia with a considerable high sensitivity (>0.9) with the overall accuracy remaining more than 0.8. However, in classification model (model 2), we were unable to find a model that could predict all the classes with a high sensitivity. The detection model can be translated to a decision support system to assist medical professionals as it is sensitive to arrhythmic conditions. But, the classification model does not have good sensitivity in detecting different classes of Arrhythmia. Since, a wrong classification can prove to be fatal, only the detection model can be adopted for daily application.

Conclusion:

Summary of Analysis:

- Through this project, we aimed to detect and classify cardiac arrhythmia into classes. We have obtained a very high accuracy with the arrhythmia detection model (model 1). Also, by reducing the threshold, the sensitivity of the detection models can be increased further.
- For the purpose of detection, the models can be utilized as a decision support system and serves the objective of the project. However, as far as classification (model 2) to the different classes of arrhythmia is concerned, with the given imbalanced data set that is heavily skewed towards class 1, obtaining a very accurate classification of all arrhythmia classes is not feasible. But since we have identified a set of strong predictors that make sense from a physiological point of view, it can be concluded that the models will definitely perform better if trained on a bigger data set that has more number of rare class instances.
- A key result we obtained from the binary classification is that a heart rate less than 51 is a clear indication of arrhythmia. So, for patients with a heart rate less than 51, if the model classifies a patient to normal class, it could potentially be a case of misclassification and should be looked into.
- Also, as we expected, personal characteristics like height, age, sex etc. did not make any considerable impact.
- As stated throughout this report, the main intention of this project was to develop an auxiliary decision support system to assist medical professionals in detecting and classifying arrhythmia. As far as detection is concerned, the model 1 can be used as a decision support system that may guide the professionals in confirming the presence of arrhythmia with a fairly high certainty. So as to have an efficient classification model (model 2) in place, more data points of the rare classes maybe needed.
- Since we have utilized only the readings of ECG as the features, adding other features that a medical professional may utilize in his diagnosis may improve the performance of the model.

Future Scope:

- Arrhythmia class 4 (Sinus arrhythmia) has low sensitivity for all the algorithms. One reasonable conclusion for such results is comparatively low number of observations. Class 4 instances can be oversampled either by bootstrapping or synthetic minority oversampling technique(SMOT).
- Model 1 and model 2 are run independently in our project. Instead, model 1 can be performed first to filter out normal instances and the remaining instances can be subjected to model 2. This will reduce the noise created in the data due to normal instances (class 1)
- Ensembles of different algorithms, for example (Neural network and random forest) can be created to classify arrhythmia by taking weighted sum of their respective probabilities to improve sensitivity or accuracy. The basic idea is to combine the best unique ability of individual algorithms and translate them into our ensemble model.
- Features can be grouped by their attributes (for example biographical features can be grouped, wave duration features can be grouped etc.) and normalized weights can be assigned to every group. More weight should be given to a group of features which have more predictive power.
- Similarly, class weighting can also be done. This can improve the power of predictive models such as SVM and KNN since more weight to classes which are in minority will improve their predictions and sensitivity.
- Stratified sampling can be performed to ensure adequate representation of rare classes in both training and testing sets. In our approach, random sampling was adopted which may not be appropriate for multi class predictions.

References:

1. http://www.heart.org/HEARTORG/Conditions/HeartFailure/AboutHeartFailure/What-is-Heart-Failure_UCM_002044_Article.jsp#.WtuFdy7wbIU
2. https://www.heart.org/idc/groups/heart-public/@wcm/@hcm/documents/downloadable/ucm_300315.pdf
3. https://pdfs.semanticscholar.org/51d4/9636fb88f5b515190dd6ab9f934e5d504b8d.pdf
4. https://www.jstatsoft.org/article/view/v036i11/v36i11.pdf
5. https://ieeexplore.ieee.org/document/7936635/
6. https://www.sciencedaily.com/releases/2009/11/091116103435.htm
7. http://www.seattlemalpracticelawyers.com/legal-services/medical-malpractice-lawyer/heart-attack-delayed-misdiagnosis/
8. http://www.ohioinjurylaw.com/columbus-medical-negligence-lawyer/heart-attack-misdiagnosis
9. https://www.ncbi.nlm.nih.gov/pubmed/1506943
10. https://onlinelibrary.wiley.com/doi/pdf/10.1002/jhrm.5600280205
11. http://epmonthly.com/blog/medical-malpractice-and-access-to-care/
12. https://en.wikipedia.org/wiki/Electrocardiography
13. https://en.wikipedia.org/wiki/Linear_discriminant_analysis
14. https://en.wikipedia.org/wiki/Multinomial_logistic_regression
15. https://en.wikipedia.org/wiki/Support_vector_machine
16. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
17. https://en.wikipedia.org/wiki/Random_forest
18. https://en.wikipedia.org/wiki/Decision_tree_learning
19. https://en.wikipedia.org/wiki/Gradient_boosting
20. https://datascience.stackexchange.com/questions/16904/gbm-vs-xgboost-key-differences
21. https://www.otexts.org/fpp/9/3
22. https://en.wikipedia.org/wiki/Quadratic_classifier#Quadratic_discriminant_analysis
23. https://machinelearningmastery.com/machine-learning-ensembles-with-r/
24. https://www.listendata.com/2017/05/feature-selection-boruta-package.html

Appendix:

Model – 1 (Binary Classes)

Reading the data file:

```
setwd("C:/Users/siddh/Desktop/ISEN 613/Cardiac
arrhythmia")
getwd()
df <- read.csv("arrhythmia.csv")
```

Data Cleaning:

```
df[df == "?"] <- NA
df$Y[df$Y!=1] <- "arrhythmia"
df$Y[df$Y==1] <- "normal"
table(df$Y)      ## 245 - normal cases, 206 - cardiac
arrhythmia conditions
## identifying dataframe columns with just one value and
removing them
unique_vect <- sapply(df, function(x) length(unique(x)))
col_rm <- sort(unique_vect, decreasing = FALSE)
## Removing all columns with just one unique value
ind_lim <- sum(col_rm<2)
list_rm <- names(col_rm)[1:ind_lim]
ix <- which(colnames(df) %in% list_rm)
df <- df[,-ix]
## identifying missing values from dataframe
na_count <-sapply(df, function(y) sum(length(which(is.na(y)))))
miss_count = sort(na_count,decreasing = TRUE)
## If there are more than 10 missing values, remove the
columns
ind_rmv <- sum(miss_count>10)
list_miss <- names(miss_count)[1:ind_rmv]
iz <- which(colnames(df) %in% list_miss)
df <- df[,-iz]
## converting factor variables to numeric
type_list <- sapply(df,class)
## converting factor variables to numeric
type_list <- sapply(df,class)
bool_convert <- (type_list!="numeric")+(type_list!="integer")
## Convert if not numeric or integer
ind_convert <- which(bool_convert==2)
ind_convert <- ind_convert[1:(length(ind_convert)-1)]
df[,ind_convert] <- sapply(df[,ind_convert],function(x)
as.numeric(x))
## Imputing missing values
k <- sapply(df,function(x) which(is.na(x)))
for (i in 1:length(k)){
  if (length(k[[i]]) < 1) {
    k[[i]] <- FALSE
  } else {
    k[[i]] <- TRUE
  } }
miss_colind <- which(unlist(k)==TRUE) ## Finding index of
missing values
print(miss_colind)   ##Tells which columns have missing
values
miss_index_qrst <- which(is.na(df$QRST_angle))
miss_index_t <- which(is.na(df$T_angle))
miss_index_heart <- which(is.na(df$Heart))
## Imputing missing value with class mean
df$QRST_angle[is.na(df$QRST_angle)] = mean(df$QRST_angle,
na.rm=TRUE)
df$T_angle[is.na(df$T_angle)] = mean(df$T_angle,
na.rm=TRUE)
df$Heart[is.na(df$Heart)] = mean(df$T_angle, na.rm=TRUE)
## converting Y into factor
df$Y <- as.factor(df$Y)
## Check sum(is.na(df))
## Scaling the dataframe
df.scaled <- as.data.frame(scale(df[1:(ncol(df)-1)]))
df.scaled['Y'] <- df$Y
```

Feature selection

```
library(caret)
set.seed(100)
test <- sample(nrow(df),floor(0.75*nrow(df)),replace = FALSE)
df_train <- df.scaled[test,]
train <- setdiff(c(1:nrow(df)),test)
df_test <- df.scaled[train,]
## Using boruta
library(Boruta)
set.seed(100)
boruta.train <- Boruta(Y~., data = df_train, doTrace =
2,maxRuns = 120)
print(boruta.train)
var_boruta <-
getSelectedAttributes(boruta.train,withTentative = TRUE)
```

Sampling:

```
#Creating training and test sets
df_train.boruta = df_train[,var_boruta]
df_train.boruta['Y']=df_train$Y
df_test.boruta  = df_test[,var_boruta]
df_test.boruta['Y']=df_test$Y
```

Logistic Regression with Ridge Regularization:

```
library(glmnet)
library(boot)
library(glmnetUtils)
set.seed(10)
glm.ridge <-  cv.glmnet(Y~.,data = df_train.boruta, family =
"binomial", type.measure = "auc", alpha = 0)
plot(glm.ridge)
print(max(glm.ridge$cvm)) ## Highest value for AUC
lambda.ridge <- glm.ridge$lambda.min
## Refitting the model with the best lambda
ridge.mod <- glmnet(Y~.,data = df_train.boruta,family =
"binomial",lambda = lambda.ridge, alpha = 0)
ridge.predict <- predict(ridge.mod,df_test.boruta,type =
"response")
ridge.class <- ifelse(ridge.predict>0.5,2,1)
## computing accuracy
```

```
confusionMatrix(as.numeric(ridge.class),as.numeric((df_test.boruta$Y)))
```

Logistic Regression with Lasso Regularization:

```
set.seed(10)
glm.lasso <-  cv.glmnet(Y~.,data = df_train.boruta, family = "binomial", type.measure = "auc", alpha = 1)
plot(glm.lasso)
print(max(glm.lasso$cvm)) ## Highest value for AUC 0.8234
lambda.lasso <- glm.lasso$lambda.min  ## Lambda value - 0.01819
## Refitting the model with the best lambda
lasso.mod <- glmnet(Y~.,data = df_train.boruta,family = "binomial",lambda = lambda.lasso, alpha = 1)
lasso.predict <- predict(lasso.mod,s = lambda.lasso,df_test.boruta,type = "response") ## Probability
lasso.class <- ifelse(lasso.predict>0.5,2,1)        ## actual classes
## computing accuracy
confusionMatrix(as.numeric(lasso.class),as.numeric(df_test.boruta$Y))
```

K-Nearest Neighbour:

```
set.seed(1)
KNN.Control <- trainControl(method  = "cv", number  = 5,classProbs = TRUE,summaryFunction = twoClassSummary)
fit <- train(Y ~ .,method = "knn",tuneGrid  = expand.grid(k = 1:50),trControl  = KNN.Control,
metric  = "ROC",data  = df_train.boruta)
print(fit)       ## The highest ROC value was observed at k = 12
trctrl_Knn=trainControl(method='repeatedcv',number=5,repeats = 5)
knn.prob <- predict(fit,df_test.boruta,type = "prob")
knn.predict <- predict(fit,df_test.boruta)
confusionMatrix(knn.predict,df_test.boruta$Y)
```

Random Forest:

```
library(randomForest)
library(pROC)
# Fitting model
y = list()
roc = list()
#for (i in seq(from = 15,to = length(var_boruta),by = 2)){
set.seed(123)
fit <-
rfcv(df_train.boruta[,var_boruta],df_train.boruta[,'Y'],step = 0.95,cv.fold = 10)
print(min(fit$error.cv))  ## The min value of error.cv is for 21 predictors
plot(fit$n.var,fit$error.cv)
## Checking best value by using test dataset
for (i in seq(from = 15,to = length(var_boruta),by = 2)){
set.seed(123)
fit <-
randomForest(df_train.boruta[,var_boruta],df_train.boruta[,'Y'],mtry = i,importance = TRUE)
predicted= predict(fit,df_test.boruta)
y[[i]] <- mean(as.numeric(df_test$Y)==as.numeric(predicted))
roc[[i]] <- roc(as.numeric(df_test$Y),as.numeric(predicted))
print(auc(roc))
print(y[[i]])
print(i)}
## Best accuracy
print(max(unlist(y)))## For 17 and 25 predictors
set.seed(123)
rf.fit <-
randomForest(df_train.boruta[,var_boruta],df_train.boruta[,'Y'],mtry = 25,importance = TRUE)
predicted.rf= predict(rf.fit,df_test.boruta)
print("The best model accuracy is")
print(mean(as.numeric(df_test.boruta$Y)==as.numeric(predicted.rf))) ## accuracy
rf.prob <-  predict(rf.fit,df_test.boruta,type = "prob")
confusionMatrix(predicted.rf,df_test.boruta$Y)
```

Gradient Boosting:

```
fitControl <- trainControl(method = "repeatedcv",number = 5, classProbs = TRUE, repeats = 2,search='random')
gbmGrid <-  expand.grid(interaction.depth = c(1,2,4), n.trees = c(500,2500,5000), shrinkage = c(0.001,0.01,0.1,1,4), n.minobsinnode = 10)
gbmFit <- train(Y~., data = df_train.boruta, method = "gbm", trControl = fitControl, verbose = TRUE, tuneGrid = gbmGrid)
predict.gbm <- predict(gbmFit,df_test.boruta,type = "raw")
gbm.probs <- predict(gbmFit,df_test.boruta,type = "prob")[2]
confusionMatrix(predict.gbm,df_test.boruta$Y)
```

Decision Tree:

```
library(tree)
set.seed(1)
tree.train_boruta <- tree(Y~.,df_train.boruta)
cv.train_boruta <- cv.tree(tree.train_boruta,FUN = prune.misclass)
print(cv.train_boruta)
par(mfrow = c(1,2))
plot(cv.train_boruta$size,cv.train_boruta$dev,type = "b")
plot(cv.train_boruta$k,cv.train_boruta$dev,type = "b")
## The lowest misclassification is for depth = 9
prune.train_boruta <- prune.misclass(tree.train_boruta,best = 9)
plot(prune.train_boruta)
text(prune.train_boruta,pretty = 0)
tree.pred <- predict(prune.train_boruta,df_test.boruta,type = "class")
confusionMatrix(tree.pred,df_test.boruta$Y)
```

Support Vector Machine:

```
set.seed(123)
svm.model <- train(Y~., data=df_train.boruta, method = 'svmRadial', tuneGrid = expand.grid(C=2^(-5:15),sigma=2^(-15:3)), trControl = trainControl(method ='repeatedcv', number=5,repeats = 5,classProbs = TRUE))
svm.model
# From results,
sigma = 0.00390625
C = 0.5
set.seed(123)
svm.model.final <- train(Y~., data=df_train.boruta, method = 'svmRadial', tuneGrid = expand.grid(C=0.5,sigma=0.00390625),
trControl = trainControl(method ='repeatedcv',
number=5,repeats = 5,classProbs = TRUE))
pred_SVM <- predict(svm.model,df_test.boruta[,-62])
SVM.prob <- predict(svm.model,df)
pred_SVM_numeric <-
as.numeric(predict(svm.model,df_test.boruta[,-62]))
CM_SVM <- confusionMatrix(pred_SVM,df_test.boruta$Y)
```

Support Vector Classifier:

```
library(e1071)
set.seed(123)
svc <- svm(Y~.,data = df_train.boruta,kernel = "linear",cost = 1)
tune.out <- tune(svm,Y~.,data = df_train.boruta,ranges =
list(cost= c(0.001,0.05,0.01,0.1,1,5,10,100)), kernel = "linear")
bestsvc <- tune.out$best.model
summary(bestsvc)
svcpred <- predict(bestsvc,df_test.boruta)
print(mean(svcpred==df_test.boruta$Y))
confusionMatrix(svcpred,df_test.boruta$Y)
```

Neural Network:

```
library(nnet)
library(caret)
for (i in seq(from = 1,to = 12,by = 1))
nn <-
nnet(Y~.,data=df_train.boruta,size=3,rang=0.07,Hess=FALSE,d
ecay=15e-4,maxit=250)
my.grid <- expand.grid(.decay = c(0.5, 0.1), .size = c(3,4,5, 6, 7))
nn.fit <- train(Y~., data = df_train.boruta, method = "nnet",
maxit = 1000, tuneGrid = my.grid, trace = F,
trControl=trainControl(method='repeatedcv',number=5,repeat
s = 5,classProbs = TRUE))
nn.predict <- predict(nn.fit,df_test.boruta)
confusionMatrix(nn.predict,df_test.boruta$Y)
```

Linear Discriminant Analysis:

```
library(caret)
trctrl_lda.binary <-
trainControl(method='repeatedcv',number=5,repeats = 5)
set.seed(123)
lda.model.binary <- train(Y~.,data=df_train.boruta,
method='lda', trControl=trctrl_lda.binary,
tuneLength=10, preProcess = 'ignore')
pred_lda.binary <- predict(lda.model.binary,df_test.boruta)
confusionMatrix(pred_lda.binary,df_test.boruta$Y)
library(pROC)
pred_lda.binary_numeric <-
ifelse(pred_lda.binary=="arrhythmia", 1, 2)
true_lda.binary_numeric <-
ifelse(df_test.boruta$Y=="arrhythmia", 1, 2)
roc(pred_lda.binary_numeric,true_lda.binary_numeric,
plot=TRUE)
```

Quadratic Discriminant Analysis:

```
trctrl_qda.binary <-
trainControl(method='repeatedcv',number=5,repeats = 5)
set.seed(123)
qda.model.binary <- train(Y~., data=df_train.boruta,
method='qda', trControl=trctrl_qda.binary,
tuneLength=10, preProcess = 'ignore')
pred_qda.binary <- predict(qda.model.binary,df_test.boruta)
confusionMatrix(pred_qda.binary,df_test.boruta$Y)
library(pROC)
pred_qda.binary_numeric <-
ifelse(pred_qda.binary=="arrhythmia", 1, 2)
true_qda.binary_numeric <-
ifelse(df_test.boruta$Y=="arrhythmia", 1, 2)
roc(pred_qda.binary_numeric,true_qda.binary_numeric,
plot=TRUE)
```

Model -2 (Multiple Classes)

Reading the data file:

```
df <- read.csv("arrhythmia.data.csv")
df[df == "?"] <- NA
```

Data Cleaning:

```
rare_lbl <- c(7,8,14,15,16)
iy <- which(df$Y %in% rare_lbl )
df <- df[-iy,]
## identifying dataframe columns with just one value and
removing them
unique_vect <- sapply(df, function(x) length(unique(x)))
col_rm <- sort(unique_vect, decreasing = FALSE)
## Removing all columns with just one unique value
ind_lim <- sum(col_rm<2)
list_rm <- names(col_rm)[1:ind_lim]
ix <- which(colnames(df) %in% list_rm)
df <- df[,-ix]
## identifying missing values from dataframe
na_count <-sapply(df, function(y) sum(length(which(is.na(y)))))
miss_count = sort(na_count,decreasing = TRUE)
## If there are more than 15 missing values, remove the
columns
ind_rmv <- sum(miss_count>15)
list_miss <- names(miss_count)[1:ind_rmv]
iz <- which(colnames(df) %in% list_miss)
df <- df[,-iz]
## converting factor variables to numeric
type_list <- sapply(df,class)
bool_convert <- (type_list!="numeric")+(type_list!="integer")
## Convert if not numeric or integer
ind_convert <- which(bool_convert==2)
df[,ind_convert] <- sapply(df[,ind_convert],function(x)
as.numeric(x))
## Imputing missing values  (change incase of low accuracy,
sub with mean)
k <- sapply(df,function(x) which(is.na(x)))
for (i in 1:length(k)){
 if (length(k[[i]]) < 1) {
  k[[i]] <- FALSE
 } else {
  k[[i]] <- TRUE
 } }
```

```
miss_colind <- which(unlist(k)==TRUE) ## Finding index of
missing values
print(miss_colind)   ##Tells which columns have missing
values
## Only use columns from miss_colind
miss_index_qrst <- which(is.na(df$QRST_angle))
miss_index_t <- which(is.na(df$T_angle))
miss_index_p <- which(is.na(df$P_angle))
#Merging classes 3-4
df$Y[which(df$Y==4)]<-3
#Merging classes 5-6
df$Y[which(df$Y==5)]<-4
df$Y[which(df$Y==6)]<-4
#Merging classes 9-10
df$Y[which(df$Y==9)]<-5
df$Y[which(df$Y==10)]<-5
## converting Y into factor
df$Y <- as.factor(df$Y)
## Imputing missing value with class mean
df$QRST_angle[is.na(df$QRST_angle)] = mean(df$QRST_angle,
na.rm=TRUE)
df$T_angle[is.na(df$T_angle)] = mean(df$T_angle,
na.rm=TRUE)
df$P_angle[is.na(df$P_angle)] = mean(df$P_angle,
na.rm=TRUE)
## Check sum(is.na(df))
## Scaling the dataframe
df.scaled <- as.data.frame(scale(df[1:(ncol(df)-1)]))
df.scaled['Y'] <- df$Y
```

Feature selection

```
library(caret)
#library(fscaret)
#levels(df.scaled$Y) <-
make.names(levels(factor(df.scaled$Y)))
set.seed(123)
test <- sample(nrow(df),floor(0.75*nrow(df)),replace = FALSE)
df_train <- df.scaled[test,]
train <- setdiff(c(1:nrow(df)),test)
df_test <- df.scaled[train,]
library(Boruta)
set.seed(123)
boruta.train <- Boruta(Y~., data = df_train, doTrace =
2,maxRuns = 120)
print(boruta.train)
var_boruta <-
getSelectedAttributes(boruta.train,withTentative = TRUE)
```

Sampling:

```
#Creating training and test sets
df_train.boruta = df_train[,var_boruta]
df_train.boruta['Y']=df_train$Y
df_test.boruta  = df_test[,var_boruta]
df_test.boruta['Y']=df_test$Y

## Scaled df with boruta features
df.scaled.boruta=df.scaled[,var_boruta]
```

Linear Discriminant Analysis:

```
set.seed(123)
lda.model = train(Y ~ ., data=df_train.boruta, method="lda",
trControl = trainControl(method = "cv"))
pred_lda=predict(lda.model,df_test.boruta)
pred_lda_numeric <-
as.numeric(predict(lda.model,df_test.boruta))
auc_lda=multiclass.roc(response = df_test.boruta$Y, predictor
= pred_lda_numeric)
```

Random Forest:

```
fitControl <- trainControl(method = "repeatedcv",number = 5,
## repeated 5 times
repeats = 5,  search='random')
grid_rf <- expand.grid(mtry = seq(4,16,4), ntree = c(700,
1000,2000) )
#Training Random Forest
set.seed(123)
RF=train(Y~., data=df_train.boruta, method='rf',
trControl=fitControl,tuneLength=20)
pred_RF=predict(RF,df_test.boruta)
mean(pred_RF==df_test.boruta$Y)
```

Support Vector Machines:

```
svm.model <- train(Y~., data=df_train.boruta, method =
'svmRadial', tuneGrid=expand.grid(C=2^(-5:5),  sigma=2^(-
7:4)), trControl=trainControl(method='repeatedcv', number=5,
repeats = 5))
pred_svt=predict(svm.model,df_test.boruta)
pred_svt_numeric <-
as.numeric(predict(svm.model,df_test.boruta))
library(pROC)
auc_svmrbf <- multiclass.roc(df_test.boruta$Y,
pred_svt_numeric)
print(auc_svmrbf$auc)
##############SVM LINEAR#####################
# Fit the model on the training set
set.seed(123)
model.svmLinear <- train(  Y ~., data = df_train.boruta, method
= "svmLinear", trControl = trainControl("repeatedcv", number
= 5,repeats=5), tuneGrid=expand.grid(C=2^(-5:15)) )
pred_svmlinear=predict(model.svmLinear,df_test.boruta)
```

Decision Trees:

```
trctrl <- trainControl(method = "repeatedcv", number = 5,
repeats = 5)
set.seed(123)
dtree_fit <- train(Y ~., data = df_train.boruta, method =
"rpart", parms = list(split = "information"), trControl=trctrl,
tuneLength = 40)
pred_dtree=predict(dtree_fit,df_test.boruta)
mean(pred_dtree==df_test.boruta$Y)
library(rpart.plot)
prp(dtree_fit$finalModel, box.palette = "Reds", tweak = 1.2)
```

XG-Boost:

```
library(mlr)
library(xgboost)
```

```
df_train.boruta.1 = df_train[,var_boruta]
ytrain.boruta.1 = as.numeric(df_train$Y)
df_test.boruta.1 = df_test[,var_boruta]
ytest.boruta.1 = as.numeric(df_test$Y)
#Change level to start from 0 for xgboost model
for (k in 1:6){
  ytrain.boruta.1[which(ytrain.boruta.1==k)] <- k-1
  ytest.boruta.1[which(ytest.boruta.1==k)] <- k-1
}
ytrain.boruta.1 = as.factor(ytrain.boruta.1)
ytest.boruta.1 = as.factor(ytest.boruta.1)
df_train.boruta.1[,'Y'] = ytrain.boruta.1
df_test.boruta.1[,'Y'] = ytest.boruta.1
#create tasks
train.task <- makeClassifTask(data = df_train.boruta.1,target =
"Y")
test.task <- makeClassifTask(data = df_test.boruta.1,target =
"Y")
xgb.learner <- makeLearner("classif.xgboost", predict.type =
"response")
# set of fixed parameters
xgb.learner$par.vals <- list(booster = "gbtree", objective =
"multi:softmax", eval_metric = "merror",
early_stopping_rounds = 50, verbose = 0, nthread = 4)
# paramaters to be tuned
params <- makeParamSet(
  makeNumericParam("eta", lower = 0.01, upper = 0.3),
  makeIntegerParam("max_depth",lower = 3L, upper = 10L),
  makeIntegerParam("nrounds", lower = 3L, upper = 20L),
  makeNumericParam("min_child_weight",lower = 1L, upper =
5L),
  makeNumericParam("subsample",lower = 0.5, upper = 1),
  makeNumericParam("colsample_bytree",lower = 0.5, upper =
1),
  makeNumericParam("lambda", lower = 0, upper = 2),
  makeDiscreteParam("gamma", values = c(0)))
# resampling
res.desc <- makeResampleDesc("CV", stratify = T, iters=5L)
# search
ctrl <- makeTuneControlRandom(maxit = 5L)
#parameter tuning
set.seed(123)
xgb.tune <- tuneParams(learner = xgb.learner, task =
train.task, resampling = res.desc, measures = mmce, par.set =
params, control = ctrl, show.info = T)
#set hyperparameters
set.seed(123)
xgb.learner_tune <- setHyperPars(xgb.learner, par.vals =
xgb.tune$x)
#train model
set.seed(125)
xgb.model <- mlr::train(learner = xgb.learner_tune, task =
train.task)
#predict model
xgb.pred <- predict(xgb.model,test.task)
confusionMatrix(xgb.pred$data$response,xgb.pred$data$trut
h)
```

KNN:

```
library(caret)
set.seed(123)
trctrl_Knn <-
trainControl(method='repeatedcv',number=5,repeats = 5)
knn.model <- train(Y~.,data=df_train.boruta,
method='knn',trControl=trctrl_Knn,tuneLength=20,preProcess
= 'ignore')
pred_knn <- predict(knn.model,df_test.boruta)
knn.model
plot(knn.model)
confusionMatrix(pred_knn, df_test.boruta$Y)
pred_knn_numeric <- as.numeric(pred_knn)
library(pROC)
auc_rf <- multiclass.roc(df_test.boruta$Y, pred_knn_numeric)
print(auc_rf$auc)
```

Neural Network:

```
library(nnet)
set.seed(123)
logistic = multinom(Y~.,data=df_train.boruta)
logistic.predict= predict(logistic, df_test.boruta)
CF_log=confusionMatrix(logistic.predict,df_test.boruta$Y)
CF_log
```

Multinomial Logsitic Regression:

```
library(nnet)
set.seed(123)
logistic = multinom(Y~.,data=df_train.boruta)
logistic.predict= predict(logistic, df_test.boruta)
CF_log=confusionMatrix(logistic.predict,df_test.boruta$Y)
CF_log
```