# “Predictive Modeling For Heart Disease”

A Project Report Submitted to

Department Of Statistics,

Shivaji University,

Kolhapur.

Submitted by,

Mr. Patil Omkar Deelip

M.Sc.II

Department Of Statistics,

Shivaji University,

Kolhapur.

Under The Guidance

Dr. D.M.Sakate

2018/19

## CERTIFICATE

This is to certify that the project entitled "*Predictive Modeling For Heart Disease",* being submitted by Mr. Patil Omkar Deelip, as a partial fulfillment for the award of degree of M.Sc. in statistics of Shivaji University, Kolhapur, is a record of bonafide work carried out by him under my supervision and guidance. To the best my knowledge the matter presented in the project has not been submitted earlier.

| | | |
|---|---|---|
| Place : Kolhapur | Dr. D.M.Sakate | Dr. H.V.Kulkarni |
| Date : | Project Guide | Head of Department, |
| | | Department of Statistics, |
| | | Shivaji University, |
| | | Kolhapur. |

## Acknowledgement

I wish to thank Department of Statistics (Shivaji University, Kolhapur) for giving me an opportunity to do a project.

This report has been prepared under the guidance of Mr. D.M.Sakate. I would like to express my profound gratitude towards him for his guidance throughout this project.

Also I would like to thank head of department Dr. H. V. Kulkarni and all faculty members.

## ❖ Contents

## 1. Introduction:-

We know that, day by day the competition has been increasing. Everybody is chasing to get more and more. Today’s life is not a bed of rose. It has become a race. There are a number of lifestyle choices that can increase the risk of heart disease. People do not have time to maintain their health. So many people has been getting victims of heart disease. The main reason behind it is that they do not take seriously the causes of heart disease. So, the main objective of this project is to predict the person will suffer from heart disease or not.

Heart disease is the leading cause of death for both men and women. Likewise there are so many factors which affect on heart. In this project there are 13 factors included. Those are, age, sex, chest pain (cp), resting blood pressure (trestbps), serum cholesterol (chol), fasting blood sugar (fbs), resting electrocardiographic (rest ecg), maximum heart rate achieved, exercise induced angina , ST depression induced by exercise relative to rest , the slope of the peak exercise ST segment, number of major vessels colored by fluoroscopy, thalassemia. There is one response variable as target which indicates that the person is diseased by heart or not (i.e. 1 or 0 respectively). Having any of these risk factors greatly increases the risk of heart disease.

In this project, historical data on 303 instances is analyzed. I have implemented various classification algorithms namely logistic regression, artificial neural network, decision tree and random forest to predict the probability of person diseased by heart.

## 2.About data:

### 2.1 Data source:

The data set used for this project is obtained from https://www.kaggle.com/ronitf/heart-disease-uci.

### 2.2 Data description:

In this data there are 303 instances and the continuous, categorical as well as binary variables coded as 0 or 1 and one of the binary variable is our response variable named as target. The list of variable is as follows,

| Sr. No. | Variable Name | Type |
|---|---|---|
| 1 | Age (year) | Continuous |
| 2 | Sex (1=male, 0=female) | Binary |
| 3 | Chest Pain Type (cp)<br>i. Typical Anginas<br>ii. Atypical Anginas<br>iii. Non Anginal Pain<br>iv. Asymptomatic | Categorical |
| 4 | Resting Blood Pressure (trestbps) | Continuous |
| 5 | Serum cholesterol (chol) | Continuous |
| 6 | Fasting Blood Sugar (fbs, 1=true, 0= false) | Binary |
| 7 | Resting Electrocardiographical Results(restecg)<br>i. 0:Normal<br>ii. 1:Having STT wave abnormality<br>iii. 2:Showing probable or definite left ventricular hypertrophy by Estes' criteria 20 ekgmo. | Categorical |
| 8 | Maximum Heart Rate Achieved (thalach) | Continuous |
| 9 | Exercise Induced Angina (exang, 1=yes, 0=no) | Binary |
| 10 | ST depression induced by exercise relative to rest(Oldpeak) | Continuous |
| 11 | Slope(The lope of the peak exercise ST segment)<br>i. Upsloping<br>ii. Flat<br>iii. Downsloping | Categorical |

|  |  |  |
|---|---|---|
| 12 | Number of Major Vessels (ca) | Categorical |
| 13 | Thalassemia (thal)<br>i. Normal<br>ii. Fixed Defect<br>iii. Reversable defect | Categorical |
| 14 | Target (1=Diseased, 0=Non diseased) | Binary |

### 2.3 Statistical tools:-

- Logistic Regression
- Data mining classifiers:
    - Artificial Neural Network
    - K-nearest neighbor
    - Decision Tree
    - Random Forest classifier

## 3. Objective:-

To classify whether the person is suffering from heart disease or not.

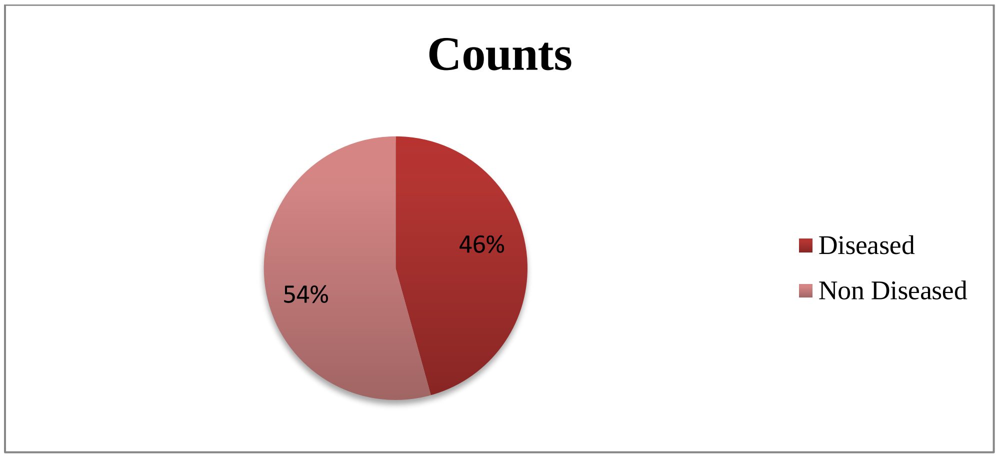
## 4. Statistical and data mining software:-

- R
- Python
- MS-Excel

## 5. Analysis:

Before doing analysis we have to check the data is balanced or not.

- **Distribution of diseased and non diseased person:**

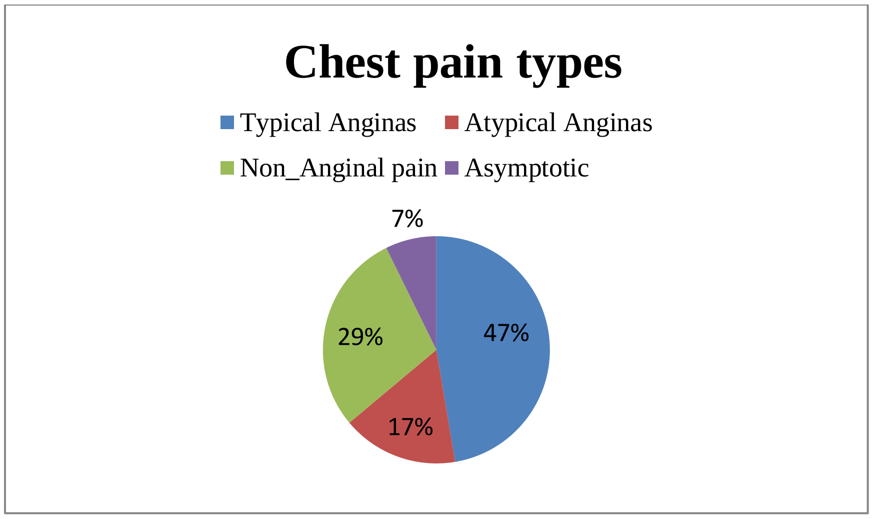


From this pie chart it is clear that, In this data set there are 54% persons are non diseased and 46% persons are diseased by heart.

- **Distribution of chest pain types**

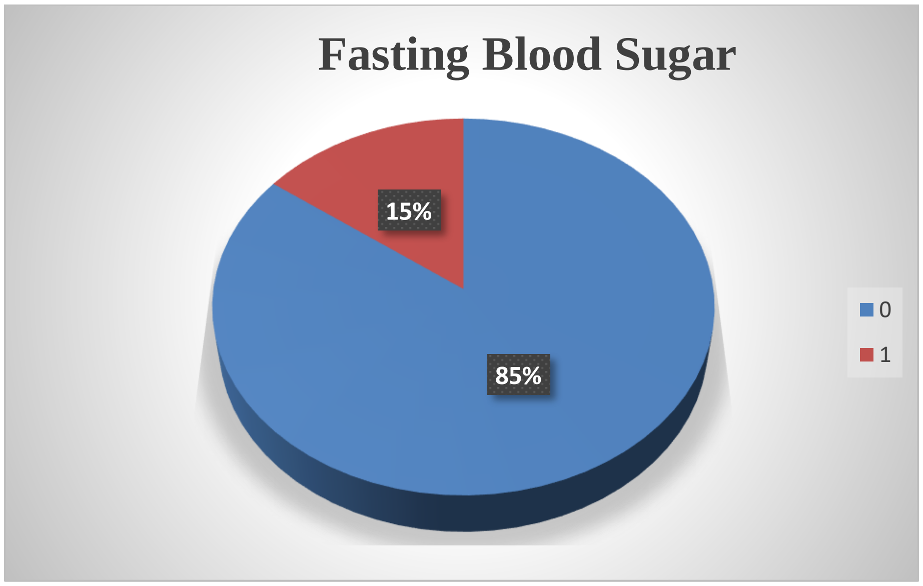
We have to check which chest pain type more occur the by diagrammatically perform as follows;

The following figure shows which chest pain type mostly occur.



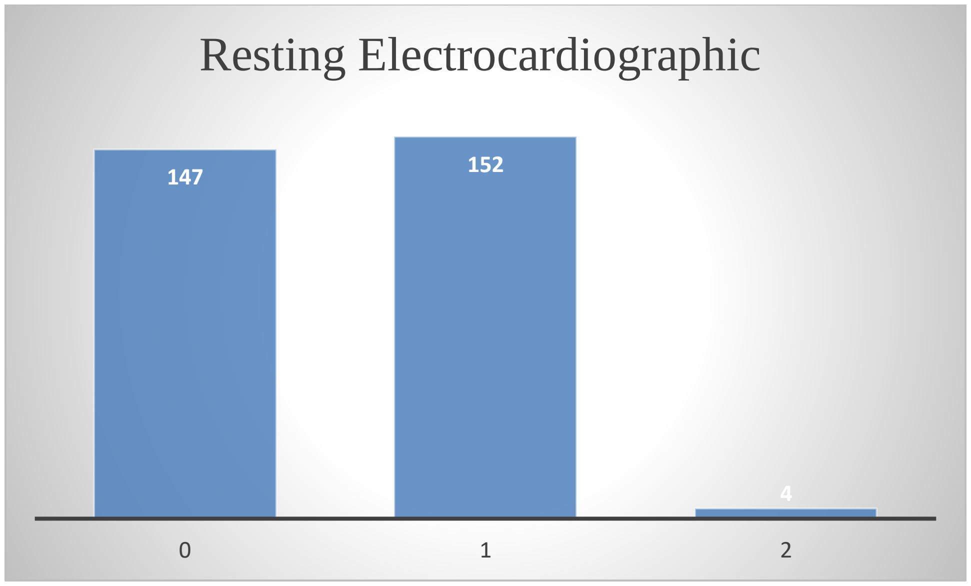
From the above pie chart it is clear that the chest pain type "Typical Anginas" is highly obtained as compare to the chest pain types.

- **Fasting Blood Sugar**



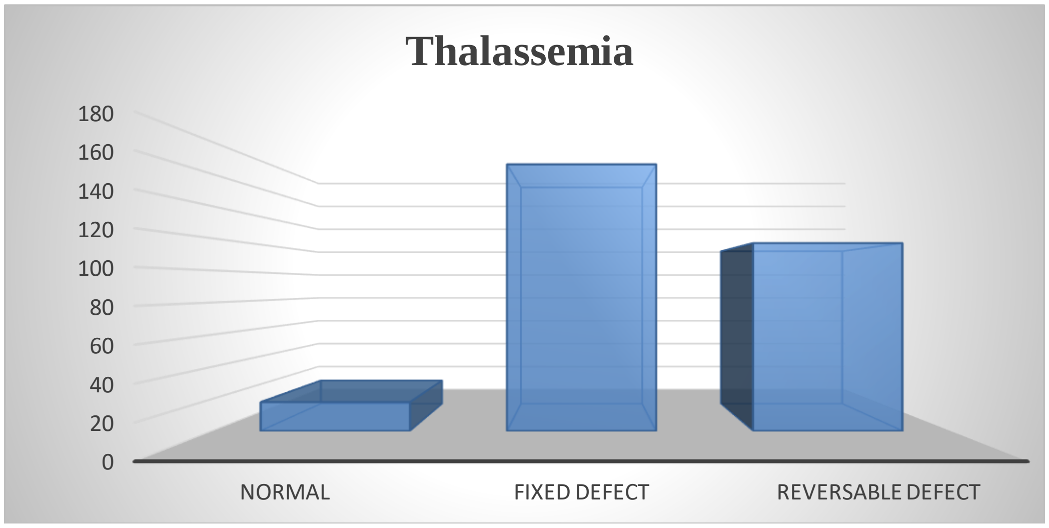
From above graph, we conclude that only 15% persons suffering from fasting blood sugar problem.

- **Resting Electrocardiographic**



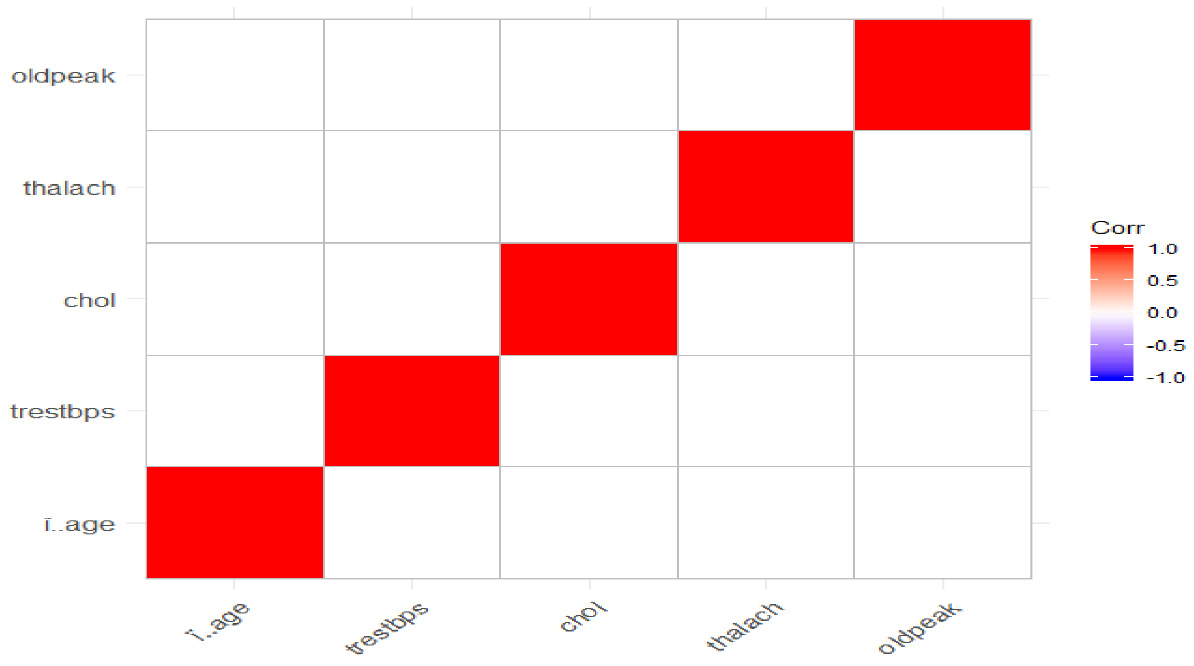
From above bar diagram, we conclude that value 0 and 1 are highly occurred. Value 0 represents the restecg is normal and value 1 represents the restecg having ST-T wave abnormality.

- **Thalassemia:**



From the above graph it is clear that fixed defects and reversal defect are more occurs.

Correlation Matrix:



The above correlation plot which can show that there is no correlation between continuous predictors.

## ➢ Some Evaluation Measures:

- **Confusion matrix:**

| | **Predicted** | | |
|---|---|---|---|
| **Actual** | Class Label | Positive Class | Negative Class |
| | Positive Class | TRUE Positive (TP) | FALSE Negative (FN) |
| | Negative Class | False Positive( FP) | True Negative (TN) |

1. Accuracy: (TP+TN)/ (TP+TN+FP+FN)
2. Sensitivity: True Positive Rate = TP/ (TP+FN)
3. Specificity: True Negative Rate = TN/ (TN+FP)
4. ROC Curve: Plot of sensitivity against (1-specificity)

## 6. Data preprocessing and model building techniques:-

### Logistic Regression:

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). It is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. Logistic regression does not try to predict the value of a numeric variable given a set of inputs. Instead, the output is a probability that the given input point belongs to a certain class. While logistic regression is a powerful modeling tool, it assumes that the response variable is linear in the coefficients of the predictor variable.

In this project the response is binary that is the person is diseased or not diseased. So, the logistic regression is used in this project. By using "Minitab" the summary as,

| **Term** | **Coef** | **SE Coef** | **95% CI** | **Z-Value** | **P-Value** |
|---|---|---|---|---|---|
| **Constant** | 0.18 | 3.7 | ( -7.08, 7.44) | 0.05 | 0.962 |
| **age** | 0.0278 | 0.0254 | ( -0.0220, 0.0776) | 1.09 | 0.274 |
| **trestbps** | -0.0261 | 0.0119 | ( -0.0495, -0.0027) | -2.19 | 0.029 |
| **chol** | -0.00428 | 0.00424 | (-0.01260, 0.00403) | -1.01 | 0.313 |
| **thalach** | 0.02 | 0.0119 | ( -0.0032, 0.0433) | 1.69 | 0.091 |
| **oldpeak** | -0.397 | 0.242 | ( -0.872, 0.078) | -1.64 | 0.101 |
| **sex** | | | | | |
| 1 | -1.861 | 0.571 | ( -2.980, -0.741) | -3.26 | 0.001 |
| **cp** | | | | | |
| 1 | 0.865 | 0.578 | ( -0.268, 1.997) | 1.5 | 0.135 |
| 2 | 2.003 | 0.529 | ( 0.965, 3.040) | 3.78 | 0 |
| 3 | 2.416 | 0.719 | ( 1.006, 3.826) | 3.36 | 0.001 |
| **fbs** | | | | | |
| 1 | 0.445 | 0.588 | ( -0.708, 1.597) | 0.76 | 0.45 |
| **restecg** | | | | | |
| 1 | 0.46 | 0.4 | ( -0.323, 1.243) | 1.15 | 0.249 |
| 2 | -0.72 | 2.77 | ( -6.14, 4.71) | -0.26 | 0.796 |
| **exang** | | | | | |
| 1 | -0.778 | 0.452 | ( -1.664, 0.107) | -1.72 | 0.085 |
| **slope** | | | | | |
| 1 | -0.775 | 0.88 | ( -2.500, 0.951) | -0.88 | 0.379 |
| 2 | 0.69 | 0.947 | ( -1.167, 2.546) | 0.73 | 0.467 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| **ca** |  |  |  |  |  |
| 1 | -2.342 | 0.527 | ( -3.375, -1.308) | -4.44 | 0 |
| 2 | -3.482 | 0.812 | ( -5.073, -1.891) | -4.29 | 0 |
| 3 | -2.242 | 0.939 | ( -4.082, -0.403) | -2.39 | 0.017 |
| 4 | 1.27 | 1.72 | ( -2.10, 4.64) | 0.74 | 0.461 |
| **thal** |  |  |  |  |  |
| 1 | 2.64 | 2.68 | ( -2.62, 7.90) | 0.98 | 0.326 |
| 2 | 2.37 | 2.6 | ( -2.72, 7.45) | 0.91 | 0.362 |
| 3 | 0.91 | 2.6 | ( -4.18, 6.01) | 0.35 | 0.725 |

## Regression Equation

P(Y'=1) = P(Getting Heart Disease) = exp(Y')/(1 + exp(Y'))

Y' = 0.18 + 0.0278 age - 0.0261 trestbps - 0.00428 chol + 0.0200 thalach
- 0.397 oldpeak+ 0.0 sex_0- 1.861 sex_1+ 0.0 cp_0+ 0.865 cp_1
+ 2.003 cp_2+ 2.416 cp_3+ 0.0 fbs_0+ 0.445 fbs_1+ 0.0 restecg_0
+ 0.460 restecg_1- 0.72 restecg_2+ 0.0 exang_0- 0.778 exang_1 + 0.0 slope_0
- 0.775 slope_1 + 0.690 slope_2 + 0.0 ca_0 - 2.342 ca_1
- 3.482 ca_2 - 2.242 ca_3 + 1.27 ca_4 + 0.0 thal_0 + 2.64 thal_1 + 2.37 thal_2
+ 0.91 thal_3

## Goodness of fit test:-

| Test | DF | Chi-Square | P-Value |
|---|---|---|---|
| Deviance | 279 | 179.62 | 1.00 |
| Hosmer-Lemeshow | 8 | 14.25 | 0.075 |

Here it is clear that the P-Value corresponding to the Deviance test and Hosmer-Lemeshow is >0.05, thus both tests is appropriate fit to the given problem.

## Odds Ratios for Continuous Predictors

| | Odds Ratio | 95% CI |
|---|---|---|
| **age** | 1.0282 | (0.9782, 1.0807) |
| **trestbps** | 0.9742 | (0.9517, 0.9973) |
| **chol** | 0.9957 | (0.9875, 1.0040) |
| **thalach** | 1.0202 | (0.9968, 1.0442) |
| **oldpeak** | 0.6722 | (0.4181, 1.0808) |

## Odds Ratios for Categorical Predictors

| Factor | Level A | Level B | Odds Ratio | 95% CI |
|---|---|---|---|---|
| | | | | |
| **sex** | 1 | 0 | 0.1556 | (0.0508, 0.4765) |
| **cp** | 1 | 0 | 2.3744 | (0.7650, 7.3699) |
| | 2 | 0 | 7.4076 | (2.6250, 20.9043) |
| | 3 | 0 | 11.2017 | (2.7358, 45.8654) |
| | 2 | 1 | 3.1198 | (0.8555, 11.3776) |
| | 3 | 1 | 4.7178 | (0.9403, 23.6703) |
| | 3 | 2 | 1.5122 | (0.3782, 6.0469) |
| **fbs** | 1 | 0 | 1.5599 | (0.4928, 4.9378) |
| **restecg** | 1 | 0 | 1.5846 | (0.7241, 3.4676) |
| | 2 | 0 | 0.4889 | (0.0022, 110.7385) |
| | 2 | 1 | 0.3085 | (0.0014, 70.0622) |
| **exang** | 1 | 0 | 0.4592 | (0.1894, 1.1134) |
| **slope** | 1 | 0 | 0.4609 | (0.0821, 2.5873) |
| | 2 | 0 | 1.9927 | (0.3112, 12.7623) |
| | 2 | 1 | 4.324 | (1.6284, 11.4821) |
| **ca** | 1 | 0 | 0.0962 | (0.0342, 0.2703) |
| | 2 | 0 | 0.0307 | (0.0063, 0.1509) |
| | 3 | 0 | 0.1062 | (0.0169, 0.6686) |
| | 4 | 0 | 3.5521 | (0.1221, 103.3662) |
| | 2 | 1 | 0.3198 | (0.0639, 1.5993) |
| | 3 | 1 | 1.1044 | (0.1686, 7.2358) |
| | 4 | 1 | 36.937 | (1.1173, 1221.0992) |
| | 3 | 2 | 3.4538 | (0.3689, 32.3329) |
| | 4 | 2 | 115.5173 | (2.5745, 5183.1433) |
| | 4 | 3 | 33.446 | (0.7665, 1459.4247) |

|  |  |  |  |  |
|---|---|---|---|---|
| **thal** | 1 | 0 | 13.9531 | (0.0725, 2684.7999) |
|  | 2 | 0 | 10.6688 | (0.0659, 1726.8719) |
|  | 3 | 0 | 2.4941 | (0.0153, 407.0827) |
|  | 2 | 1 | 0.7646 | (0.1532, 3.8165) |
|  | 3 | 1 | 0.1788 | (0.0366, 0.8718) |
|  | 3 | 2 | 0.2338 | (0.0978, 0.5590) |

➢ Age:

- If the unit change corresponding to the continuous variable age then the chance of getting heart disease is increased by 0.0282%.

➢ Resting Blood Pressure:

- If the unit change corresponding to the continuous variable tresbps then the chance of getting heart disease is decreased by 0.0258%.

➢ ST depression induced by exercise relative to rest:

- If the unit change corresponding to the oldpeak then the chance of getting heart disease is decreased by 0.33%.

➢ Gender:

- Chance of occurring heart diseases in male 84.44% lesser than the female.

➢ Chest Pain:

- The chance of occurring heart disease in a person who has atypical angina chest pain is 2.37 fold more than the person having typical angina chest pain.
- The chance of occurring heart disease in a person who has non-atypical angina chest pain is 7.41 fold more than the person having typical angina chest pain.
- The chance of occurring heart disease in a person who has asymptotical angina chest pain is 11.20 fold more than the person having typical angina chest pain.

- The chance of occurring heart disease in a person who has non-atypical chest pain is 3.12 fold more than the person having atypical angina chest pain.
- The chance of occurring heart disease in a person who has asymptotical angina chest pain is 4.72 fold more than the person having atypical angina chest pain.
- The chance of occurring heart disease in a person who has asymptotical angina chest pain is 51% more than the person having non-atypical angina chest pain.

➢ Fasting blood sugar:

- The chance of occurring heart disease in a person who has fasting blood sugar is 55% more than the person do not having fasting blood sugar.

➢ Resting electrocardiographic results:

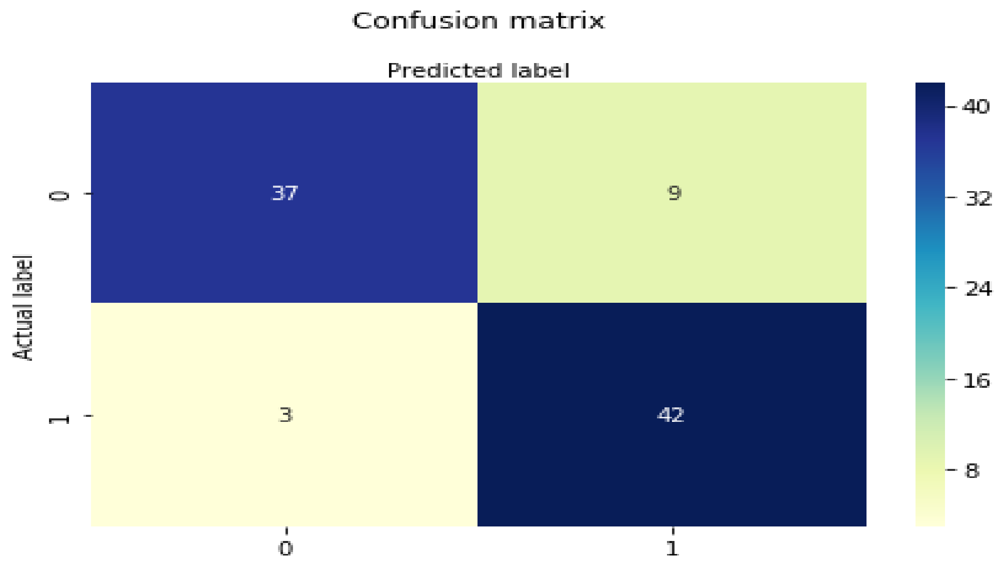
- The chance of occurring heart disease in a person whose resting electrocardiographic results contain STT wave abnormality is 58% more than the person whose result is normal.
- The chance of occurring heart disease in a person whose resting electrocardiographic results showing probable or definite left ventricala hypertrophy by Esters Criterial 20 ekgmo is 51.11% lesser than the person whose result is normal.
- The chance of occurring heart disease in a person whose resting electrocardiographic results showing probable or definite left ventricala hypertrophy by Esters Criterial 20 ekgmo is 69.15% lesser than the person whose result contain STT wave abnormality.

➢ Exercise induced angina:

- The chance of occurring heart disease in a person who do exercise is decreased by 54% than the person who don’t.

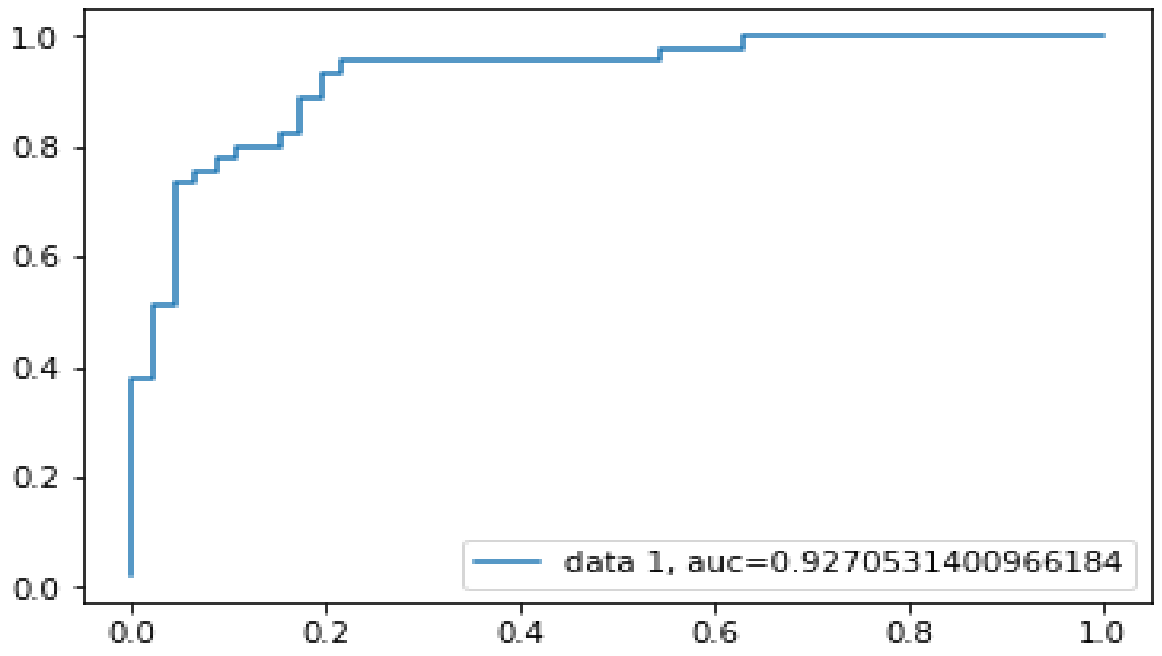
**Confusion matrix in logistic regression**

| Measures | Test Data in % |
|---|---|
| Accuracy | 86.81 |
| Sensitivity | 93.33 |
| Specificity | 80.43 |



## ROC curve for logistic regression

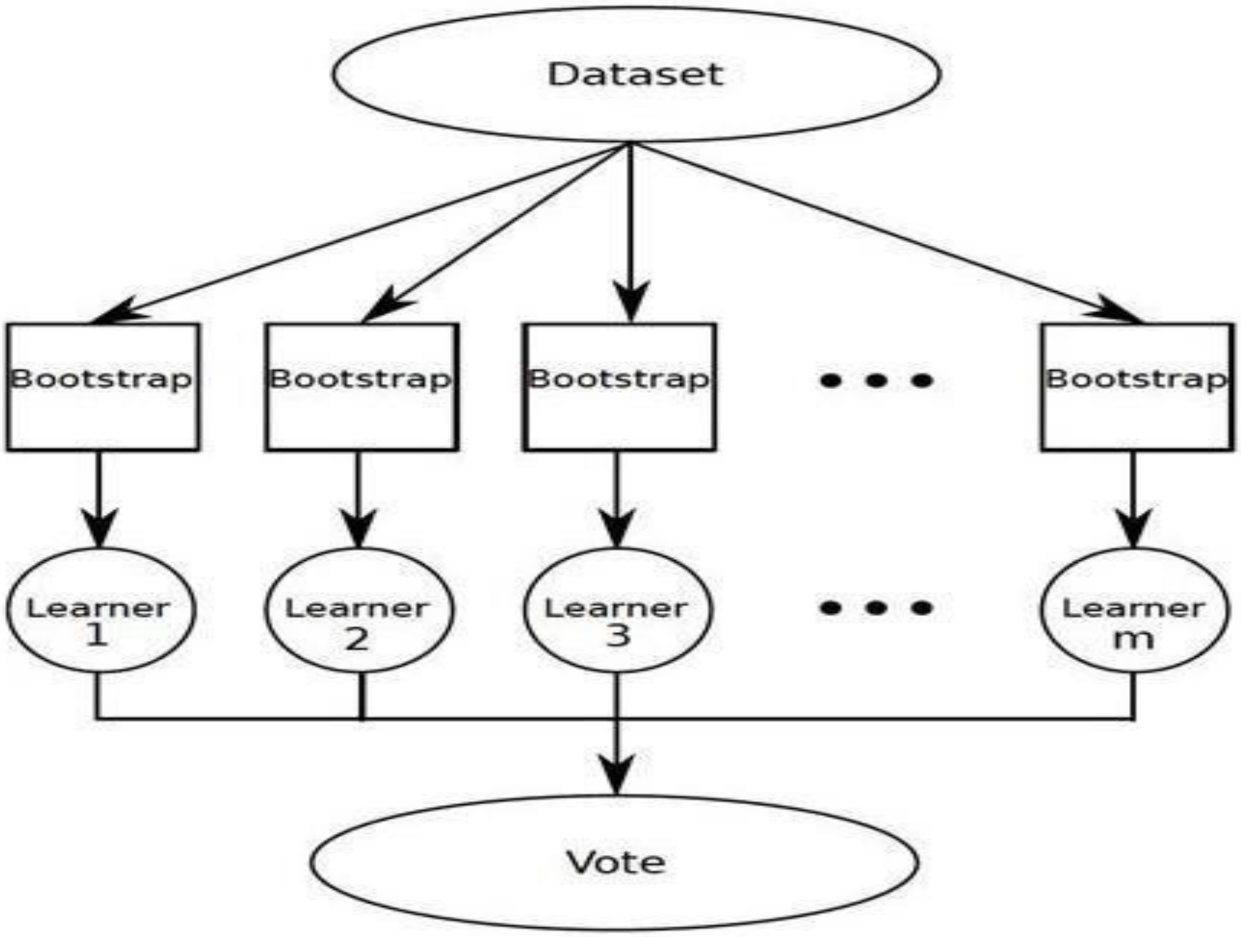
ROC stands for Receiver Operating Characteristic. ROC curves are a nice way to see how any predictive model can distinguish between the true positives and negatives. ROC curve does this by plotting the true positive rate (Sensitivity) in function of the false positive rate (100-Specificity) for different cut-off points of a parameter. The area under the ROC curve ( AUC ) is a measure of how well a parameter can distinguish between two diagnostic groups (diseased/ non-diseased).



Fitted logistic regression classifier has accuracy 86.81%, sensitivity 93.33 and specificity 80.43 with ROC curve having area under curve 0.9246. From all these accuracy measures we conclude that model fits good for train data and also works well for test data.

The accuracy of the fitted logistic model was 86.81% which is good enough, but we want to improve the accuracy, thus we move to bagging ensemble method.

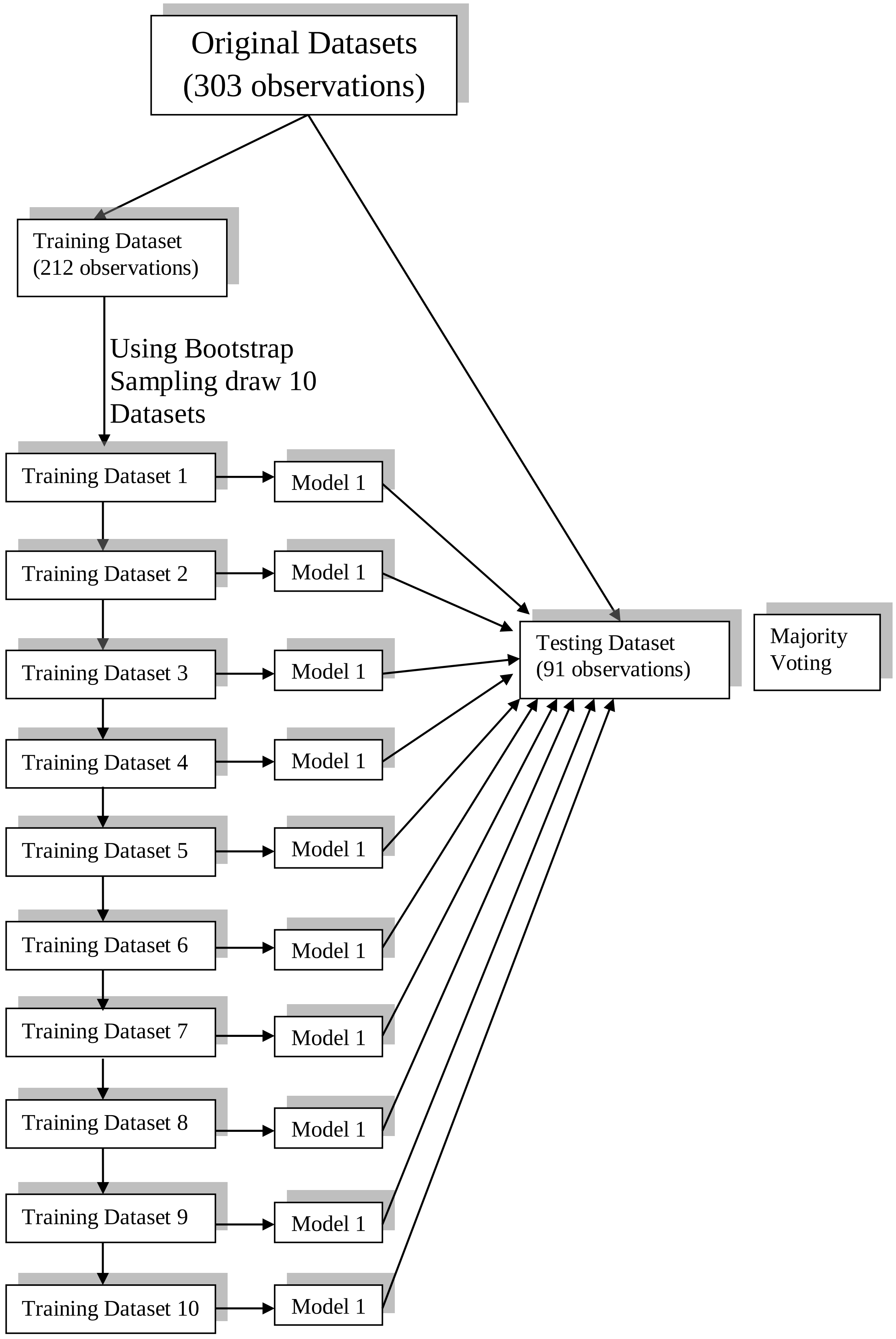
**Bagging method:**
Bagging is a simple but powerful ensemble algorithm that facilitates the increased stability & accuracy of classification models. The Bagging process works by generating multiple training datasets via random sampling with replacement, applying the same model to each training dataset, and then taking the majority vote amongst the models to determine data classifications. Bagging is a particularly popular method because it reduces variance, helps to prevent over fitting (i.e. forced applicability of random irrelevant data), and it can be easily parallelized for application to large datasets.


Dataset
Bootstrap
Bootstrap
Bootstrap
• • •
Bootstrap
Learner 1
Learner 2
Learner 3
• • •
Learner m
Vote

## 7. Model Building:-

Fortunately in our data there are no missing values are present. Firstly we find majority class and minority class counts of the response variable which is the particular person is suffering from heart disease or not. In the data we have 303 person under study. Out of which 165 are heart disease patients and remaining 138 are not a heart disease patients. Thus, we have a approximately balanced data. Now first we have to partition the original dataset into 70%(training dataset) and 30%(testing dataset). Now generating 10 multiple training datasets via random sampling with replacement from the training data set, applying the same logistic regression model to each training dataset. Then these 10 models are test on the test data which gives classification for the each tuple of test dataset and then taking the majority vote amongst the models to determine classification of the each tuples of the test data. When we use these models on the test data then for each tuples gives a prediction probability and we use 0.5 cut off point for the classifying the tuples. By using ROC curve we find optimal cut off point and calculate the accuracy of the model for which classification error is minimum and which is nearly same for the 0.5 cut off point. Similarly these 10 models are used for the whole data which gives classification of the each tuples of the whole data and then taking majority vote amongst the models to determine classification for each tuples of whole dataset here also use 0.5 cut off value for each model for classifying the tuples of the whole data. For both test and whole data the confusion matrix and evaluation measures are calculated.

The diagrammatical representation of model building is as follows;



The 10 fitted model to the 10trained data
The model is
Y(target) ~ Intercept + age + sex + cp + tresbps + chol + fbs + restecg + thalach + exang + oldpeak + slope + ca + thal

## Model 1:

| | Estimate Std | . Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | 6.582479 | 3.732707 | 1.763 | 0.07782 | . |
| ï..age | -0.064336 | 0.035754 | -1.799 | 0.07195 | . |
| sex | -4.274488 | 0.901966 | -4.739 | 2.15E-06 | *** |
| cp | 1.278542 | 0.262769 | 4.866 | 1.14E-06 | *** |
| trestbps | -0.037008 | 0.012388 | -2.987 | 0.00281 | ** |
| chol | -0.008085 | 0.006449 | -1.254 | 0.20996 | |
| fbs | -0.493522 | 0.721314 | -0.684 | 0.49385 | |
| restecg | 0.072506 | 0.487145 | 0.149 | 0.88168 | |
| thalach | 0.049841 | 0.018305 | 2.723 | 0.00647 | ** |
| exang | -1.499422 | 0.583551 | -2.569 | 0.01019 | * |
| oldpeak | -0.247114 | 0.274474 | -0.9 | 0.36795 | |
| slope | 0.976874 | 0.451236 | 2.165 | 0.0304 | * |
| ca | -0.471144 | 0.291119 | -1.618 | 0.10558 | |
| thal | -1.03432 | 0.354432 | -2.918 | 0.00352 | ** |

## Model 2:

| | Estimate | Std. Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | -0.88718 | 3.652858 | -0.243 | 0.8081 | |
| ï..age | 0.011157 | 0.029358 | 0.38 | 0.70392 | |
| sex | -2.62794 | 0.667521 | -3.937 | 8.26E-05 | *** |
| cp | 0.880773 | 0.243015 | 3.624 | 0.00029 | *** |
| trestbps | -0.0196 | 0.012421 | -1.578 | 0.11454 | |
| chol | -0.00656 | 0.00576 | -1.14 | 0.25448 | |
| fbs | -0.09183 | 0.685933 | -0.134 | 0.8935 | |
| restecg | 0.338404 | 0.467252 | 0.724 | 0.46892 | |
| thalach | 0.047014 | 0.016613 | 2.83 | 0.00466 | ** |
| exang | -0.50914 | 0.555091 | -0.917 | 0.35902 | |
| oldpeak | -0.54651 | 0.261108 | -2.093 | 0.03634 | * |
| slope | 0.620424 | 0.475871 | 1.304 | 0.19231 | |
| ca | -0.76845 | 0.231895 | -3.314 | 0.00092 | *** |
| thal | -0.60351 | 0.355782 | -1.696 | 0.08983 | . |

## Model 3:

| | Estimate | Std. Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | 4.832807 | 3.393402 | 1.424 | 0.154395 | |
| ï..age | 0.000155 | 0.02864 | 0.005 | 0.995671 | |
| sex | -0.69942 | 0.504951 | -1.385 | 0.166013 | |
| cp | 0.805177 | 0.247102 | 3.258 | 0.00112 | ** |
| trestbps | -0.0029 | 0.012371 | -0.234 | 0.814728 | |
| chol | -0.00728 | 0.005548 | -1.312 | 0.189436 | |
| fbs | -0.30431 | 0.597841 | -0.509 | 0.610738 | |
| restecg | 0.924971 | 0.435999 | 2.121 | 0.03388 | * |
| thalach | -0.00085 | 0.014943 | -0.057 | 0.954727 | |
| exang | -1.23043 | 0.50775 | -2.423 | 0.01538 | * |
| oldpeak | -0.9457 | 0.287062 | -3.294 | 0.000986 | *** |
| slope | 0.954642 | 0.443769 | 2.151 | 0.031459 | * |
| ca | -0.98104 | 0.234063 | -4.191 | 2.77E-05 | *** |
| thal | -1.1671 | 0.386096 | -3.023 | 0.002504 | ** |

## Model 4:

| | Estimate Std | . Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | 4.327551 | 4.059874 | 1.066 | 0.28645 | |
| ï..age | -0.02295 | 0.034362 | -0.668 | 0.50418 | |
| sex | -1.12639 | 0.549813 | -2.049 | 0.04049 | * |
| cp | 1.684479 | 0.319024 | 5.28 | 1.29E-07 | *** |
| trestbps | -0.02834 | 0.015625 | -1.814 | 0.06972 | . |
| chol | -0.00638 | 0.005983 | -1.065 | 0.28667 | |
| fbs | 0.008341 | 0.664493 | 0.013 | 0.98998 | |
| restecg | 1.654339 | 0.513931 | 3.219 | 0.00129 | ** |
| thalach | 0.015352 | 0.016231 | 0.946 | 0.34423 | |
| exang | -1.28485 | 0.580062 | -2.215 | 0.02676 | * |
| oldpeak | -0.47095 | 0.310887 | -1.515 | 0.1298 | |
| slope | 0.640052 | 0.502246 | 1.274 | 0.20253 | |
| ca | -0.41699 | 0.22237 | -1.875 | 0.06076 | . |
| thal | -0.51674 | 0.362035 | -1.427 | 0.15349 | |

## Model 5:

| | Estimate | Std. Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | 7.034091 | 3.729894 | 1.886 | 0.059313 | . |
| ï..age | -0.04051 | 0.035128 | -1.153 | 0.248846 | |
| sex | -2.44931 | 0.65649 | -3.731 | 0.000191 | *** |
| cp | 0.626871 | 0.237475 | 2.64 | 0.008297 | ** |
| trestbps | -0.02309 | 0.014661 | -1.575 | 0.115208 | |
| chol | -0.01184 | 0.005495 | -2.154 | 0.031208 | * |
| fbs | 0.331173 | 0.686544 | 0.482 | 0.629539 | |
| restecg | 0.495468 | 0.429371 | 1.154 | 0.248525 | |
| thalach | 0.038231 | 0.014405 | 2.654 | 0.007955 | ** |
| exang | -1.98239 | 0.651799 | -3.041 | 0.002355 | ** |
| oldpeak | -0.64266 | 0.267145 | -2.406 | 0.016143 | * |
| slope | 0.458838 | 0.426635 | 1.075 | 0.282159 | |
| ca | -0.04267 | 0.244453 | -0.175 | 0.861435 | |
| thal | -1.41175 | 0.39361 | -3.587 | 0.000335 | *** |

## Model 6:

| | Estimate | Std. Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | 3.695416 | 3.401484 | 1.086 | 0.277296 | |
| ï..age | -0.00607 | 0.031611 | -0.192 | 0.847825 | |
| sex | -1.68651 | 0.51151 | -3.297 | 0.000977 | *** |
| cp | 0.615086 | 0.219874 | 2.797 | 0.005151 | ** |
| trestbps | -0.01505 | 0.011958 | -1.259 | 0.208184 | |
| chol | -0.0029 | 0.005726 | -0.507 | 0.612322 | |
| fbs | -0.41837 | 0.598814 | -0.699 | 0.484763 | |
| restecg | 0.444597 | 0.430098 | 1.034 | 0.301271 | |
| thalach | 0.017077 | 0.013622 | 1.254 | 0.209969 | |
| exang | -1.85134 | 0.536282 | -3.452 | 0.000556 | *** |
| oldpeak | -0.58627 | 0.25304 | -2.317 | 0.020509 | * |
| slope | 0.38392 | 0.363377 | 1.057 | 0.290725 | |
| ca | -0.38286 | 0.217469 | -1.761 | 0.078321 | . |
| thal | -0.92235 | 0.3441 | -2.68 | 0.007352 | ** |

## Model 7:

| | Estimate | Std. Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | -0.72922 | 4.132885 | -0.176 | 0.85995 | |
| ï..age | 0.002954 | 0.031439 | 0.094 | 0.92514 | |
| sex | -1.45811 | 0.699587 | -2.084 | 0.03714 | * |
| cp | 0.931154 | 0.309415 | 3.009 | 0.00262 | ** |
| trestbps | -0.00433 | 0.016323 | -0.265 | 0.79068 | |
| chol | -0.00947 | 0.006621 | -1.431 | 0.15246 | |
| fbs | -1.31458 | 0.748869 | -1.755 | 0.07919 | . |
| restecg | 0.984569 | 0.51722 | 1.904 | 0.05697 | . |
| thalach | 0.044324 | 0.017825 | 2.487 | 0.0129 | * |
| exang | -1.14401 | 0.674495 | -1.696 | 0.08987 | . |
| oldpeak | -0.41313 | 0.337818 | -1.223 | 0.22135 | |
| slope | 1.001533 | 0.453825 | 2.207 | 0.02732 | * |
| ca | -1.26729 | 0.278085 | -4.557 | 5.18E-06 | *** |
| thal | -1.38135 | 0.432228 | -3.196 | 0.00139 | ** |

## Model 8:

| | Estimate | Std. Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | -2.7912 | 3.478643 | -0.802 | 0.422333 | |
| ï..age | -0.0005 | 0.027814 | -0.018 | 0.985754 | |
| sex | -1.41691 | 0.538808 | -2.63 | 0.008546 | ** |
| cp | 0.763821 | 0.226641 | 3.37 | 0.000751 | *** |
| trestbps | -0.01448 | 0.012235 | -1.183 | 0.2367 | |
| chol | -0.00702 | 0.005663 | -1.24 | 0.21501 | |
| fbs | 0.938163 | 0.707921 | 1.325 | 0.185093 | |
| restecg | 0.510158 | 0.444416 | 1.148 | 0.250998 | |
| thalach | 0.044785 | 0.014691 | 3.048 | 0.002301 | ** |
| exang | -0.36551 | 0.515248 | -0.709 | 0.478083 | |
| oldpeak | -0.32683 | 0.265414 | -1.231 | 0.218175 | |
| slope | 1.19898 | 0.44369 | 2.702 | 0.006886 | ** |
| ca | -0.87827 | 0.255576 | -3.436 | 0.000589 | *** |
| thal | -0.50684 | 0.354476 | -1.43 | 0.152765 | |

## Model 9:

| | Estimate | Std. Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | 4.457772 | 3.422573 | 1.302 | 0.19276 | |
| ï..age | -0.00128 | 0.030877 | -0.041 | 0.96697 | |
| sex | -1.53799 | 0.54838 | -2.805 | 0.00504 | ** |
| cp | 0.578928 | 0.212109 | 2.729 | 0.00635 | ** |
| trestbps | -0.04249 | 0.01373 | -3.095 | 0.00197 | ** |
| chol | -0.00568 | 0.005539 | -1.025 | 0.30537 | |
| fbs | 0.547923 | 0.705302 | 0.777 | 0.43724 | |
| restecg | 0.679168 | 0.440004 | 1.544 | 0.1227 | |
| thalach | 0.033438 | 0.014298 | 2.339 | 0.01936 | * |
| exang | -1.59403 | 0.54985 | -2.899 | 0.00374 | ** |
| oldpeak | -0.39629 | 0.260668 | -1.52 | 0.12844 | |
| slope | 0.782593 | 0.441204 | 1.774 | 0.0761 | . |
| ca | -0.84075 | 0.296382 | -2.837 | 0.00456 | ** |
| thal | -0.89338 | 0.360767 | -2.476 | 0.01327 | * |

## Model 10:

| | Estimate | Std. Error | z value | Pr(>\|z\|) | |
|---|---|---|---|---|---|
| (Intercept) | -0.86068 | 4.233468 | -0.203 | 0.8389 | |
| ï..age | 0.02721 | 0.033722 | 0.807 | 0.41973 | |
| sex | -2.17838 | 0.793632 | -2.745 | 0.00605 | ** |
| cp | 0.690605 | 0.261338 | 2.643 | 0.00823 | ** |
| trestbps | -0.03438 | 0.015394 | -2.233 | 0.02553 | * |
| chol | -0.00393 | 0.007524 | -0.523 | 0.60105 | |
| fbs | -0.32956 | 0.808129 | -0.408 | 0.68342 | |
| restecg | 0.232705 | 0.513874 | 0.453 | 0.65066 | |
| thalach | 0.05307 | 0.017968 | 2.954 | 0.00314 | ** |
| exang | -1.49412 | 0.644955 | -2.317 | 0.02052 | * |
| oldpeak | -0.52017 | 0.339665 | -1.531 | 0.12567 | |
| slope | 1.313905 | 0.492827 | 2.666 | 0.00767 | ** |
| ca | -0.7983 | 0.278139 | -2.87 | 0.0041 | ** |
| thal | -1.08822 | 0.397504 | -2.738 | 0.00619 | ** |

## Confusion matrix :

| For Test data | | Predicted | | |
|---|---|---|---|---|
| | | 0 | 1 | T |
| **Actual** | 0 | 38 | 6 | 44 |
| | 1 | 4 | 43 | 47 |
| | T | 42 | 49 | 91 |

| **Measures** | **Test Data in %** |
|---|---|
| Accuracy | 89.01 |
| Sensitivity | 91.49 |
| Specificity | 86.36 |

## Conclusion:

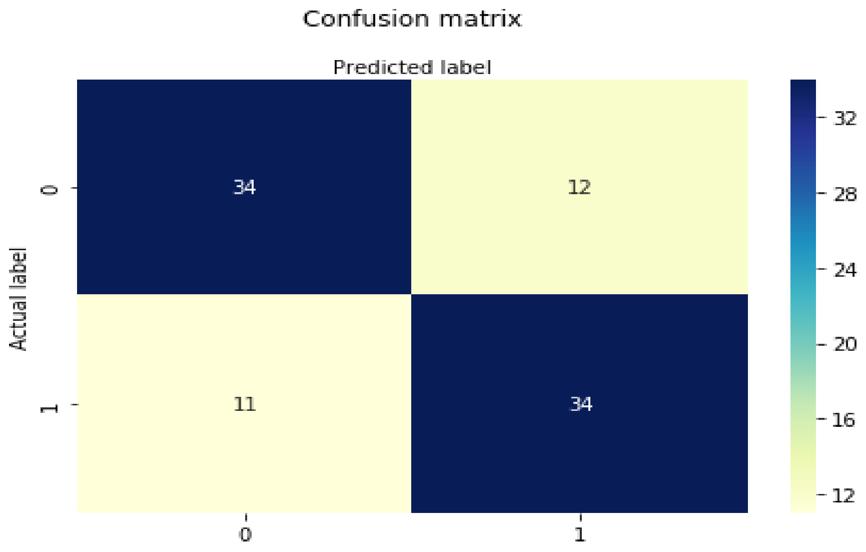
The model has accuracy 89.01, sensitivity 91.49 and specificity 86.36. From all these accuracy measures we conclude that model fits good for train data and also works well for test data.

## k- Nearest Neighbors

Nearest neighborhood classifier is based on learning by analogy i.e. by comparing a given test tuple with training tuples that are similar to it. The training tuples are described by n attribute. Each tuple represents a point in n-dimensional space. In this way all the training tuples are stored in an n-dimensional pattern space

When given an unknown tuple, a k-nearest neighborhood searches the pattern space for k-training tuples that closest to the unknown. These k training all the a k-nearest neighbors of the unknown tuple.

### Confusion matrix



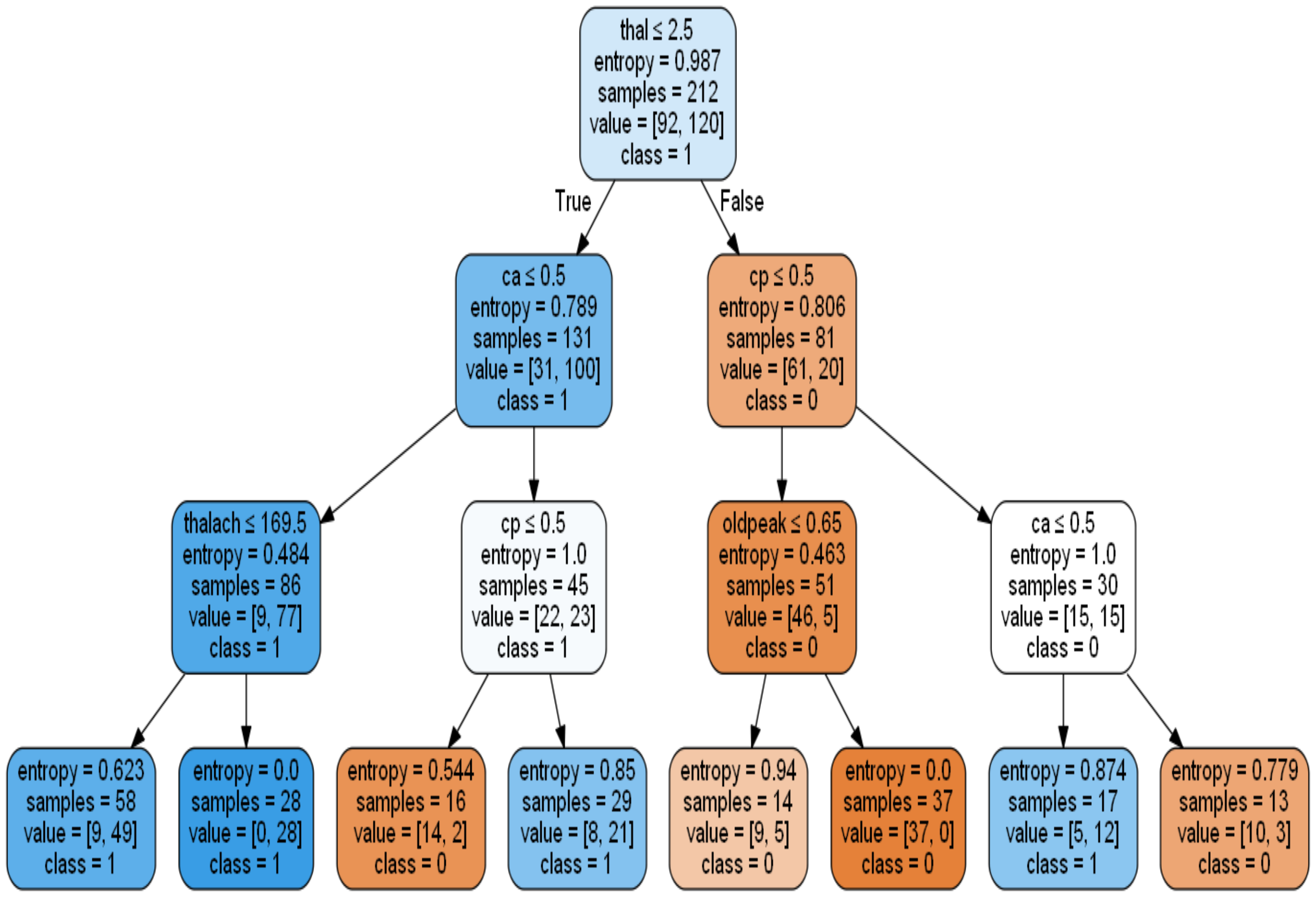
| Measures | Test Data in % |
|---|---|
| Accuracy | 74.72 |
| Sensitivity | 75.55 |
| Specificity | 73.91 |

Fitted k- Nearest Neighbor classifier has accuracy 74.72%, sensitivity 75.55 and specificity 73.91 with ROC curve having area under curve 0.9246. From all these accuracy measures we conclude that model fits good for train data and also works well for test data.

## Decision Tree

A decision tree is a classifier expressed as a recursive partition of the instance space. The decision tree consists of nodes that form a rooted tree, meaning it is a directed tree with a node called "root" that has no incoming edges. All other nodes have exactly one incoming edge. A node with outgoing edges is called an internal or test node. All other nodes are called leaves (also known as terminal or decision nodes).

Given a tuple, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traces from the root to the leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules

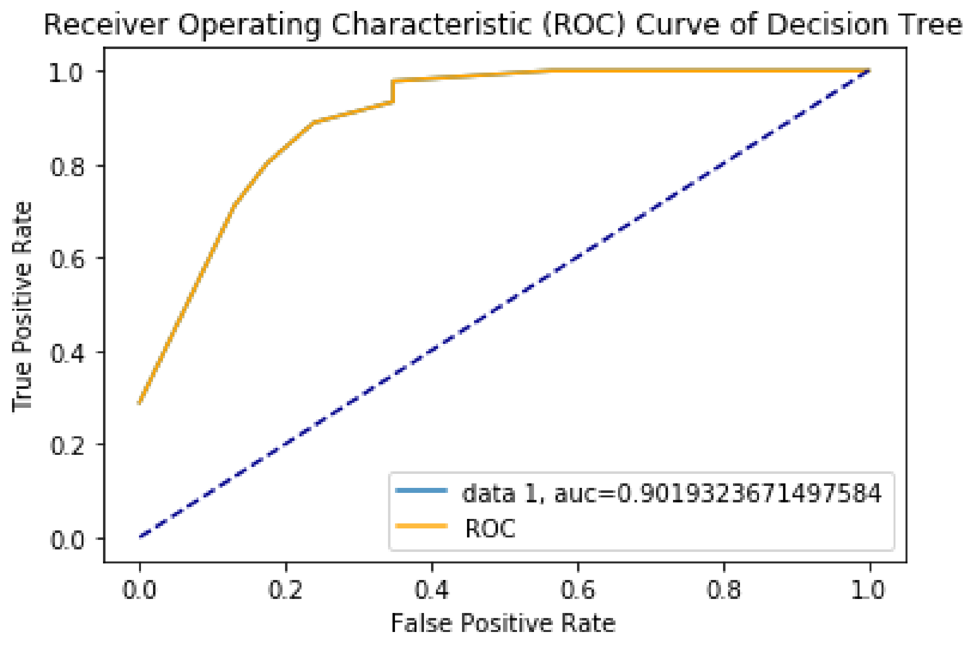


- **Confusion matrix:-**

| **For Test data** | | **Predicted** | | |
|---|---|---|---|---|
| | | 0 | 1 | T |
| **Actual** | 0 | 35 | 11 | 46 |
| | 1 | 5 | 40 | 45 |
| | T | 40 | 51 | 91 |

| **Measures** | **Test Data in %** |
|---|---|
| Accuracy | 82.42 |
| Sensitivity | 88.89 |
| Specificity | 76.08 |

**ROC curve for Decision Tree**



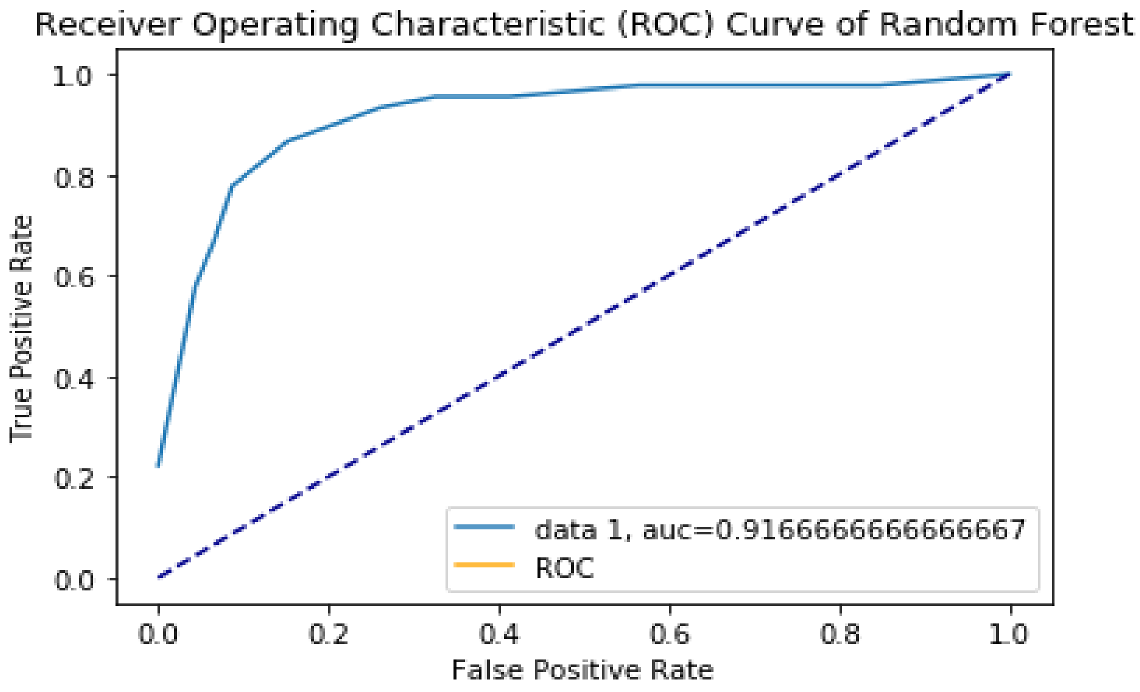
Fitted Decision Tree has accuracy 82.42%, sensitivity 88.89 and specificity 76.08 with ROC curve having area under curve 0.9020. From all these accuracy measures we conclude that model fits good for train data and also works well for test data.

## Random Forest

- **Confusion matrix:-**

| **For Test data** | | **Predicted** | | |
|---|---|---|---|---|
| | | 0 | 1 | T |
| **Actual** | 0 | 39 | 7 | 46 |
| | 1 | 6 | 39 | 45 |
| | T | 45 | 46 | 91 |

| **Measures** | **Test Data in %** |
|---|---|
| Accuracy | 85.71 |
| Sensitivity | 86.67 |
| Specificity | 84.78 |



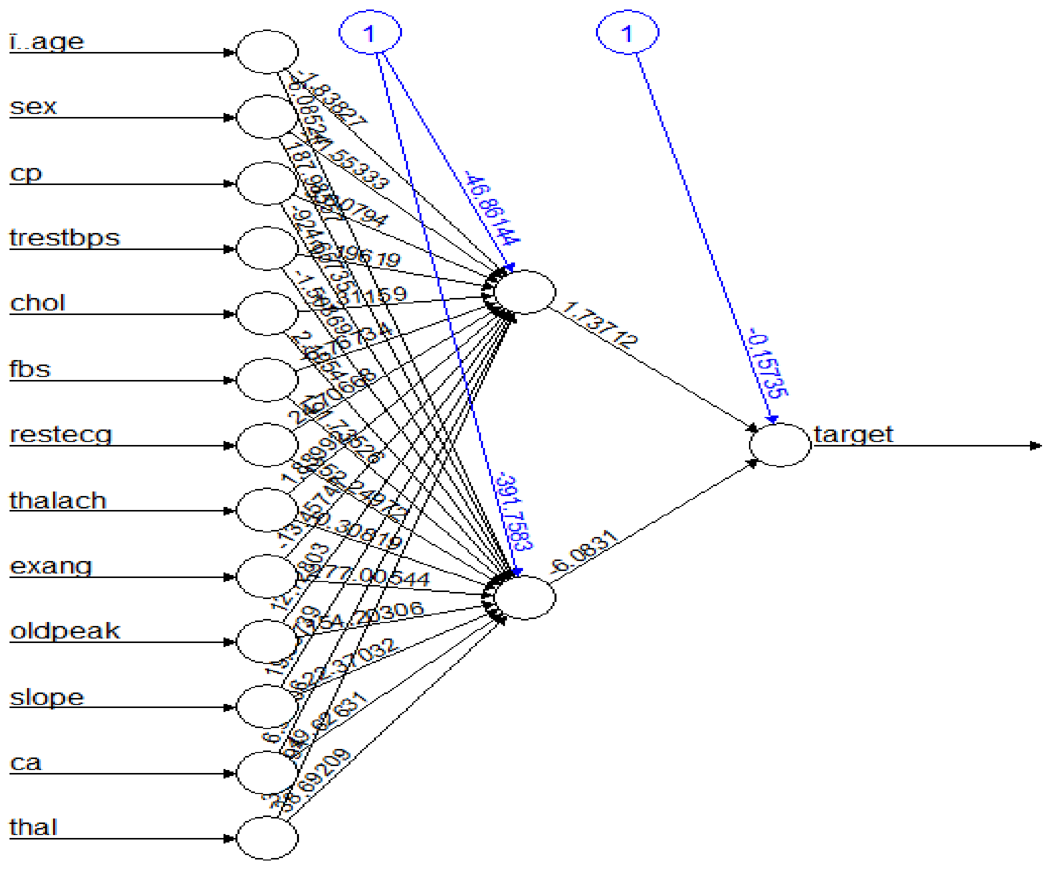
Fitted Random Forest classifier has accuracy 86.81%, sensitivity 86.67 and specificity 84.78 with ROC curve having area under curve 0.9167. From all these accuracy measures we conclude that model fits good for train data and also works well for test data.

## Artificial Neural Network:

Neural network is a set of connected input/output units in which each connection has a weight associated with it. During the learning phase, the network learns by adjusting the weights so as to be able to predict the class label of the input tuples.

Neural network consist of an input layer, one or more hidden layers, threshold value and an output layer. It learns by iteratively processing a data set of training tuples, comparing the network's prediction for each tuple with the actual known target value (threshold value). The target value may be the known class label of the training tuple or a continuous value. For each training tuple, the weights are modified so as to minimize the mean-squared error between the network's prediction and the actual target value.

**Fitted model**



| **For Test data** | | **Predicted** | | |
|---|---|---|---|---|
| | | 0 | 1 | T |
| **Actual** | 0 | 32 | 9 | 41 |
| | 1 | 14 | 36 | 50 |
| | T | 46 | 45 | 91 |

| **Measures** | **Test Data in %** |
|---|---|
| Accuracy | 74.72 |
| Sensitivity | 72.00 |
| Specificity | 78.04 |

Fitted Artificial Neural Network model has accuracy 74.72%, sensitivity 72.00 and specificity 78.04. From all these accuracy measures we conclude that model fits good for train data and also works well for test data.

## 8.Benefits and Limitations:-

- Benefits: If a person who is going to suffer a heart attack is predicted earlier premintive measure can be taken to avoid heart attack.
- Limitations: This study is limited on this data only and can not be universal applicable.

## 9.Refference:-

- Data Mining: Concepts and Techniques,
  -Jiawei Han, Micheline Kamber, Jian Pei (2011). Third Edition.

## 10.Appendix:-

**Python Code: For Logistic, k-NN, Random Forest, Decision Tree**

```
import numpy as np
import pandas as pd
import pandas
data1 = pandas.read_csv("C:/Users/stats2/Desktop/heart.csv")
data1
import seaborn as sns
np.shape(data1)

### Train test splitting

from sklearn.model_selection import train_test_split
y=data1.target
x=data1.drop('target',axis=1)
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100)

print(x_train,x_test,y_train,y_test)

x_train.shape

x_test.shape
```

**## Logistic Regression**

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()

logreg.fit(x_train,y_train)

y_pred=logreg.predict(x_test)

y_pred

from sklearn import metrics
```

```
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)

cnf_matrix

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

class_names=[0,1] # name  of classes

fig, ax = plt.subplots()

tick_marks = np.arange(len(class_names))

plt.xticks(tick_marks, class_names)

plt.yticks(tick_marks, class_names)

# create heatmap

sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set_label_position("top")

plt.tight_layout()

plt.title('Confusion matrix', y=1.1)

plt.ylabel('Actual label')

plt.xlabel('Predicted label')

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

print("Precision:",metrics.precision_score(y_test, y_pred))

print("Recall:",metrics.recall_score(y_test, y_pred))

y_pred_proba = logreg.predict_proba(x_test)[::,1]

fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
```

```
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

## **Bagging in R.**

```
data=read.csv("C:/Users/stats2/Desktop/heart.csv",header=T)
n=nrow(data)
s=sample(1:n,round(0.7*n)) # training indices
ts=seq(1,n)[-s]    # testing indices
y1=rep(0,91)
for(i in 1:10)
{
train=sample(s,length(s),replace=T)
data1=data[train,]
model=glm(target~.,family="binomial",data=data1)
y=predict(model,data[ts,-14],type="response")
y.pred=rep(0,91)
y.pred[y>.5]=1

print(summary(model))
accuracy=mean(y.pred==data[ts,14])
print(accuracy)
y1=y1+y.pred
}

#creating y*

y_star=rep(0,91)

y_star[y1>5]=1

#creatingconfusion matrix

table(y_star,data[ts,14])

#storingaccuracy

accuracy1=mean(y_star==data[ts,14])

accuracy1
```

**## KNN**

```
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=6)
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
print(y_test,y_pred)
print(accuracy_score(y_test,y_pred))
y_test
y_pred
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
import scipy as sp
a=confusion_matrix(y_test, y_pred)
FP = a[0,1]
FN =a[1,0]
TP =  a[1,1]
TN = a[0,0]
print(a)
```

```
# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
print(TPR)
# Specificity or true negative rate
TNR = TN/(TN+FP)
print(TNR)
# Precision or positive predictive value
PPV = TP/(TP+FP)
# Negative predictive value
NPV = TN/(TN+FN)
# Fall out or false positive rate
FPR = FP/(FP+TN)
# False negative rate
FNR = FN/(TP+FN)
# False discovery rate
FDR = FP/(TP+FP)

# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(x_train, y_train)
y_pred=model.predict(x_test)
print(accuracy_score(y_test,y_pred))
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
def plot_roc_curve(FPR,TPR):
    plt.plot(FPR,TPR, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()
y_pred
probs = model.predict_proba(x_test)
probs = probs[:, 1]
FPR,TPR,thresholds = roc_curve(y_test, probs)
plot_roc_curve(FPR,TPR)
```

**## Random Forest**

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(x_train, y_train)
y_pred=model.predict(x_test)
print(accuracy_score(y_test,y_pred))
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
d=confusion_matrix(y_test, y_pred)
FP = d[0,1]
FN =d[1,0]
TP =  d[1,1]
TN = d[0,0]

# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
# Specificity or true negative rate
TNR = TN/(TN+FP)
# Precision or positive predictive value
PPV = TP/(TP+FP)
# Negative predictive value
NPV = TN/(TN+FN)
# Fall out or false positive rate
FPR = FP/(FP+TN)
# False negative rate
FNR = FN/(TP+FN)
```

```
# False discovery rate
FDR = FP/(TP+FP)

# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
y_pred
probs = model.predict_proba(x_test)
probs = probs[:, 1]
FPR,TPR,thresholds = roc_curve(y_test, probs)
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
def plot_roc_curve(FPR,TPR):
    plt.plot(FPR,TPR, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve of Random Forest ')
    plt.legend()
    plt.show()
plot_roc_curve(FPR,TPR)
```

**## Decision Tree**

```
###Import Required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
##class count of reponse variable
data1['target'].value_counts()

## Training
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
```

```
classifier.fit(x_train, y_train)
##Evaluating the Algorithm
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
## Now we have to install the Graphviz2.38 And then proceed
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'
##Visualize Decesion tree
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split
function
from sklearn import metrics #Import scikit-learn metrics module for accuracy
calculation

import graphviz
import graphviz

dot_data = export_graphviz(classifier, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("heart")
import os
os.getcwd()
x_train.columns
dot_data = export_graphviz(classifier, out_file=None,
                      feature_names=x_train.columns,
                 class_names=np.array([0,1],dtype='<U10'),
                    filled=True, rounded=True,
                    special_characters=True)
graph = graphviz.Source(dot_data)
graph
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
```

```
dot_data = StringIO()
export_graphviz(classifier, out_file=dot_data,feature_names=x_train.columns,
                 class_names=np.array([0,1],dtype='<U10'),
                filled=True, rounded=True,
                special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
classifier = DecisionTreeClassifier()

# Train Decision Tree Classifer
classifier = classifier.fit(x_train,y_train)
#Predict the response for test dataset
y_pred = classifier.predict(x_test)
y_pred
# Create Decision Tree classifer object
classifier = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
classifier = classifier.fit(x_train,y_train)

#Predict the response for test dataset
y_pred = classifier.predict(x_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
b=confusion_matrix(y_test, y_pred)
FP = b[0,1]
FN =b[1,0]
TP =  b[1,1]
TN = b[0,0]

# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
# Specificity or true negative rate
TNR = TN/(TN+FP)
# Precision or positive predictive value
PPV = TP/(TP+FP)
# Negative predictive value
NPV = TN/(TN+FN)
# Fall out or false positive rate
```

```
FPR = FP/(FP+TN)
# False negative rate
FNR = FN/(TP+FN)
# False discovery rate
FDR = FP/(TP+FP)

# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)

#x_test=x_test.drop('Q.N.',axis=1)
probs =classifier.predict_proba(x_test)
probs = probs[:, 1]
FPR,TPR,thresholds = roc_curve(y_test, probs)
FPR,TPR,thresholds
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
def plot_roc_curve(FPR,TPR):
    fpr, tpr, _ = metrics.roc_curve(y_test,probs)
    auc = metrics.roc_auc_score(y_test,probs)
    plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
    plt.plot(FPR,TPR, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve of Decision Tree')
    plt.legend()
    plt.show()
plot_roc_curve(FPR,TPR)
```

**The R Code For Artificial Neural Network:-**

```
# ANN_classifier

rm(list=ls())
install.packages('ISLR')
library(ISLR)
data=read.csv("C:/Users/stats2/Desktop/heart.csv",header=T)
head(data,2)
# Create Vector of Column Max and Min Values
maxs <- apply(data[,1:14], 2, max)
mins <- apply(data[,1:14], 2, min)

# Use scale() and convert the resulting matrix to a data frame
scaled.data <- as.data.frame(scale(data[,1:14],center = mins, scale = maxs - mins))

# Check out results
print(head(scaled.data,2))

data$target
library(caTools)
set.seed(101)

# Create Split (any column is fine)
split = sample.split(data$target, SplitRatio = 0.70)

# Split based off of split Boolean Vector
train = subset(data, split == TRUE)
test = subset(data, split == FALSE)
dim(train)
dim(test)
length(which(split == FALSE))

feats <- names(scaled.data)
feats <- feats[-14]

# Concatenate strings
f <- paste(feats,collapse=' + ')
f <- paste('target ~',f)

# Convert to formula
```

```
f <- as.formula(f)
f

#install.packages('neuralnet')
library(neuralnet)
nn <- neuralnet(f,train,hidden=2,linear.output=FALSE)
plot(nn)

# Compute Predictions off Test Set
predicted.nn.values <- compute(nn,test[1:14])

# Check out net.result
print(head(predicted.nn.values$net.result))
predicted.nn.values$net.result <-
sapply(predicted.nn.values$net.result,round,digits=0)


table(Actual=test$target,predicted=predicted.nn.values$net.result)
```