# HEART DISEASE PREDICTION USING MACHINE LEARNING ALGORITMS

*Report submitted to the SASTRA Deemed to be University as the requirement for the course*

**CSE300 - MINI PROJECT**

*Submitted by*

**KANAMARLAPUDI VENKATA NAGA SREEVALLI**
**(Reg. No.: 123015126, IT)**
**AKULA DHODEEPYA SREE**
**(Reg. No.: 123014003, ICT)**

**June 2022**


**SCHOOL OF COMPUTING**

**THANJAVUR, TAMIL NADU, INDIA – 613 401**

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

**SCHOOL OF COMPUTING THANJAVUR**

**– 613 401**

## Bonafide Certificate

This is to certify that the report titled "**Heart Disease Prediction Using Machine Learning Algorithms**" submitted as a requirement for the course, CSE300 **MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Ms. Akula Dhodeepya Sree (Reg. No. 123014003, ICT**), **Ms. Kanamarlapudi Venkata Naga Sreevalli (Reg. No. 123015126, IT)** during the academic year 2021-22, in the School of Computing, under my supervision.

**Signature of Project Supervisor :**

**Name with Affiliation** : **Dr. Rajkumar K, Senior Assistant Professor, SoC**

**Date** : **29-06-2022**

Mini Project *Viva voc*e held on 29-06-2022

**Examiner 1**

**Examiner 2**

# Table of Contents

# ACKNOWLEDGEMENTS

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan,** Dean**,** Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli,** Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. A. Umamakeswari**, Dean, School of Computing, **Dr. B.Santhi**, Associate Dean, Department of Computer Application, **Dr. R. Muthaiah**, Associate Dean, Department of Information Technology and Information & Communication Technology and **Dr. V. S. Shankar Sriram**, Associate Dean, Department of Computer Science and Engineering for their motivation and support offered in materializing this project.

Our guide Mr. Rajkumar K, Senior Assistant Professor, School of Computing was the driving force behind this whole idea from the start. His deep insight in the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing me an opportunity to showcase my skills through project.

# List of Figures

# Abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| ML | Machine Learning |
| HIPPA | Health Insurance Portability and Accountability Act |
| KNN | K-Nearest Neighbours |
| SVM | Support Vector Machine |
| LR | Logistic Regression |
| Corr | Correlation |
| STD | Standard Deviation |
| UCI | Unique Client Identifier |
| FBS | Fasting blood sugar level |

## Abstract

This work focuses on the construction of an artificial intelligence-based heart disease detection system using machine learning algorithms. Machine learning algorithms can predict if a person gets heart disease or not in the future. Machine learning is crucial for determining whether anyone has suffered from heart disease. In either case, if these are predicted ahead of time, doctors would have a much easier time gaining crucial information for treating and diagnosing patients. The main phases of application developments describe collecting databases, performing different ML algorithms, and evaluating the dataset's attributes. This work computes the accuracy score obtained by different algorithms like Logistic Regression, Naïve Bayes, Support Vector Machine, K Nearest Neighbours, Decision Tree, Random Forest, and Neural Networks. At last, it compares scores of all these algorithms and conclude which algorithm gives good result. It is a python-based application. Python is well founded and it helps in tracking and establishing different type of health monitoring systems. Here, the system is trained and implemented with the help of UCI (Unique Client Identifier) machine learning deported benchmark dataset. With the help of several features, such as gender, age, cp (chest pain type), chol (cholesterol level), FBS (fasting blood sugar level), exang (exercise induced angina), thalach (maximum achieved heart rate), old peak (ST depression persuaded by exercise respective to rest), thal (maximum achieved heart rate), ca (number of major vessels) it can predict if the person is having heart disease or not. This work also calculate correlation between all these attributes. The coding packages and libraries used in this work are Pandas, Matplotlib, IPython, Numpy, Python, SciPy, and many others.



**KEY WORDS:** Artificial Intelligence, Machine Leaming, Logistic Regression, Naive Bayes, Support Vector Machine, K Nearest Neighbours, Decision Tree, Random Forest, Neural Networks, Python.

# CHAPTER 1

## SUMMARY OF THE BASE PAPER

**Base paper Title:** An artificial intelligence model for heart disease detection using machine learning algorithms

**Author List:** Victor Chang, Vallabhanent, Rupa Bhavani, Ariel Qianwen Xu, MA Hossain

**Year:** November 2022

Base paper is taken from sciencedirect.com

### INTRODUCTION

Nowadays we can see people are getting lot of health issues due to low immunity and some other reasons when compared to olden days. Death rate is increasing year by year. The greatest challenge of health sector is to decrease this death rate. We can see there is a good improvement when it comes to infrastructure wise, and every day we can see scientists are inventing new equipment. There will be some cases where we can get to know about our disease or problem so that we can meet doctor and there will be some cases where we can get to know about our problem only after doing some tests by doctors. Due to this late knowing of the disease, even though we have good equipment and best doctors, in lot of cases doctors are also not getting scope to cure the disease. This problem can be reduced by using machine learning algorithms where we can use different patients' data and can create a predictive model which can be used to detect if we are having the same disease are not.

When it comes to heart diseases, till now an estimated of 19million people died due to different cardio vascular diseases which are nearly 33% of the global deaths which is a very finite percentage. So, there is a very big need to decrease this count. In this a finite number of deaths are only because of late knowing of the disease. So, there is a very big need to create a model which can predict the disease in early stage where machine learning will be very helpful.

This paper focused on the building of heart disease prediction model using machine learning algorithms in python to predict if a person is suffering from heart disease or not. In this base paper they described different steps from collecting database, doing pre-processing, evaluating data to creating model.

## ARCHITECTURE AND ALGORITHMS PROPOSED

This model was made using python that deals with HIPPA requirements for health records. the dataset was taken from uci repository which has 14 attributes. These 14 attributes are gender, age, cp (chest pain type), chol (cholesterol level), FBS (fasting blood sugar level), exang (exercise induced angina), thalach (maximum achieved heart rate), old peak (ST depression persuaded by exercise respective to rest), thal (maximum achieved heart rate), ca (number of major vessels). lot of python libraries like numpy pandas, matplotlib, sklearn and many more are used. The dataset have both categorical and continuous data and the target is categorical. Because of this Different supervised classification algorithms like logistic regression, naïve bayes, knn, support vector machine, random forest and more are applied in which random forest got highest accuracy of 83%. Correlation matrix is also constructed to know how each pair of attributes related and to find if they are linearly related or not.

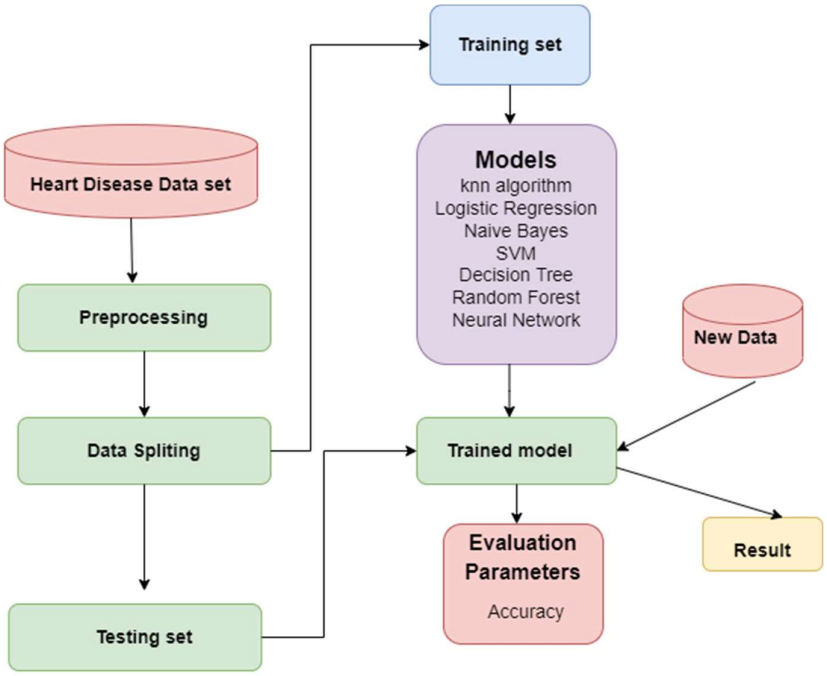



**Fig-1.1 Architecture/ workflow of machine learning model**

In this process of model creation, first the dataset has to be cleaned. The cleaning involves updating the missing values, removing duplicates, dropping observations and features, impute missing values, finding outliers and many more. Next the data has to be pre-processed. In this pre-processing lot of steps like importing libraries, importing dataset, encoding categorical data, splitting the data into train test split, feature scaling and many more will be performed. Next the training data is given to different machine learning algorithms to create a model. After model is created the model is given with test data to find the accuracy. The algorithm with highest accuracy is given with new data and the output will be predicted.

# CHAPTER 2

# MERITS AND DEMERITS OF THE BASE PAPER

## Related Works

S Mohan, C et al. (S Mohan, C,2019) By utilising machine learning approaches, they suggested a novel approach that aims to identify relevant traits, increasing the accuracy in the cardiovascular disease prognosis created an improved performance with a precision level of 88.7% of those using the heart disease prediction model. using a linear model and a hybrid random forest.

MM Ali et al. (MM Ali,2021) They discovered that the RF technique achieved 100% accuracy coupled with 100% sensitivity and specificity utilising a heart disease dataset obtained from Kaggle three-classification based on k-nearest neighbour (KNN), decision tree (DT), and random forests (RF) algorithms. They discovered that heart disease predictions may be made with extremely high accuracy and excellent potential value using a relatively straightforward supervised machine learning approach.

M Kavitha, G et al. (M Kavitha, G,2021) The machine learning model's innovative technique is created. Three machine learning methods are employed in the implementation: 1. Random Forest, 2. Hybrid model (hybrid of random) and Decision Tree decision tree. Results of experiments indicate through the heart disease, an accuracy rate of 88.7% was achieved. using a hybrid model for prediction. The connection is intended to forecast what the user's input parameter will be.

P Rani et al. (P Rani,2021) In the simulation environment created with Python, the suggested hybrid system was evaluated. On the Cleveland heart disease dataset found in the machine learning repository at UCI (University of California, Irvine), it was tested. Its accuracy of 86.6 percent outperforms some of the other heart disease prediction algorithms that have been documented in the literature.

Riyaz, L et al. (Riyaz, L,2022) This paper provides a thorough analysis of the performance of several machine learning approaches utilised for effective prediction, diagnosis, and treatment of major cardiac illnesses. The average forecast accuracy was then determined for each technique to see which performed the best and worst overall. The results showed that ANN had the highest average prediction accuracy (86.91%), while C4.5 decision tree approach had the lowest average prediction accuracy (74.0%).

## MERITS AND DEMERITS

There are lot of traditional approaches and angiography which are very much suitable methods for curing and predicting cardiac vascular diseases. Which work same as our predictive machine learning models sometimes even better than machine learning models. This is one of the drawbacks of machine learning models.

When it comes to merits it is an automatic technique where computer itself will do all the data interpretation and analysis. It can handle any kind of data and can convert any type of data to other type. It's a learning process where model will learn and improve themselves because of which accuracy can be improved.

When it comes to drawbacks these models are highly vulnerable and once if any error occurred, as we use same model to predict the data, it will lead to chain of errors and it will become very difficult to find of source of error.

Unlike deep learning feature extraction should be done for this model separately.

Sometimes there will be situations where it will take lot of time to create a model. Little manipulation of data will lead to long chain errors and leads to inaccuracy. More space is required.

There are hybrid methods for machine learning which are combination of one supervised and one unsupervised method which work even better than normal machine learning algorithms.

# CHAPTER 3

# DATASET INFORMATION

Number of Instances: 303
Attributes used: 14

| | |
|---|---|
| age: | age |
| sex: | 1: male, 0: female |
| cp: | chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic |
| trestbps: | resting blood pressure |
| chol: | serum cholestoral in mg/dl |
| fbs: | fasting blood sugar > 120 mg/dl |
| restecg: | resting electrocardiographic results (values 0,1,2) |
| thalach: | maximum heart rate achieved |
| exang: | exercise induced angina |
| oldpeak: | oldpeak = ST depression induced by exercise relative to rest |
| slope: | the slope of the peak exercise ST segment |
| ca: | number of major vessels (0-3) colored by flourosopy |
| thal: | thal: 3 = normal; 6 = fixed defect; 7 = reversable defect |
| target: | 0-no heart disease, 1-have heart disease |

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
| 2 | 63 | 1 | 3 | 145 | 233 | 1 | 0 | 150 | 0 | 2.3 | 0 | 0 | 1 | 1 |
| 3 | 37 | 1 | 2 | 130 | 250 | 0 | 1 | 187 | 0 | 3.5 | 0 | 0 | 2 | 1 |
| 4 | 41 | 0 | 1 | 130 | 204 | 0 | 0 | 172 | 0 | 1.4 | 2 | 0 | 2 | 1 |
| 5 | 56 | 1 | 1 | 120 | 236 | 0 | 1 | 178 | 0 | 0.8 | 2 | 0 | 2 | 1 |
| 6 | 57 | 0 | 0 | 120 | 354 | 0 | 1 | 163 | 1 | 0.6 | 2 | 0 | 2 | 1 |
| 7 | 57 | 1 | 0 | 140 | 192 | 0 | 1 | 148 | 0 | 0.4 | 1 | 0 | 1 | 1 |
| 8 | 56 | 0 | 1 | 140 | 294 | 0 | 0 | 153 | 0 | 1.3 | 1 | 0 | 2 | 1 |
| 9 | 44 | 1 | 1 | 120 | 263 | 0 | 1 | 173 | 0 | 0 | 2 | 0 | 3 | 1 |
| 10 | 52 | 1 | 2 | 172 | 199 | 1 | 1 | 162 | 0 | 0.5 | 2 | 0 | 3 | 1 |
| 11 | 57 | 1 | 2 | 150 | 168 | 0 | 1 | 174 | 0 | 1.6 | 2 | 0 | 2 | 1 |
| 12 | 54 | 1 | 0 | 140 | 239 | 0 | 1 | 160 | 0 | 1.2 | 2 | 0 | 2 | 1 |
| 13 | 48 | 0 | 2 | 130 | 275 | 0 | 1 | 139 | 0 | 0.2 | 2 | 0 | 2 | 1 |
| 14 | 49 | 1 | 1 | 130 | 266 | 0 | 1 | 171 | 0 | 0.6 | 2 | 0 | 2 | 1 |
| 15 | 64 | 1 | 3 | 110 | 211 | 0 | 0 | 144 | 1 | 1.8 | 1 | 0 | 2 | 1 |
| 16 | 58 | 0 | 3 | 150 | 283 | 1 | 0 | 162 | 0 | 1 | 2 | 0 | 2 | 1 |
| 17 | 50 | 0 | 2 | 120 | 219 | 0 | 1 | 158 | 0 | 1.6 | 1 | 0 | 2 | 1 |
| 18 | 58 | 0 | 2 | 120 | 340 | 0 | 1 | 172 | 0 | 0 | 2 | 0 | 2 | 1 |
| 19 | 66 | 0 | 3 | 150 | 226 | 0 | 1 | 114 | 0 | 2.6 | 0 | 0 | 2 | 1 |
| 20 | 43 | 1 | 0 | 150 | 247 | 0 | 1 | 171 | 0 | 1.5 | 2 | 0 | 2 | 1 |
| 21 | 69 | 0 | 3 | 140 | 239 | 0 | 1 | 151 | 0 | 1.8 | 2 | 2 | 2 | 1 |
| 22 | 59 | 1 | 0 | 135 | 234 | 0 | 1 | 161 | 0 | 0.5 | 1 | 0 | 3 | 1 |
| 23 | 44 | 1 | 2 | 130 | 233 | 0 | 1 | 179 | 1 | 0.4 | 2 | 0 | 2 | 1 |
| 24 | 42 | 1 | 0 | 140 | 226 | 0 | 1 | 178 | 0 | 0 | 2 | 0 | 2 | 1 |
| 25 | 61 | 1 | 2 | 150 | 243 | 1 | 1 | 137 | 1 | 1 | 1 | 0 | 2 | 1 |
| 26 | 40 | 1 | 3 | 140 | 199 | 0 | 1 | 178 | 1 | 1.4 | 2 | 0 | 3 | 1 |
| 27 | 71 | 0 | 1 | 160 | 302 | 0 | 1 | 162 | 0 | 0.4 | 2 | 2 | 2 | 1 |
| 28 | 59 | 1 | 2 | 150 | 212 | 1 | 1 | 157 | 0 | 1.6 | 2 | 0 | 2 | 1 |
| 29 | 51 | 1 | 2 | 110 | 175 | 0 | 1 | 123 | 0 | 0.6 | 2 | 0 | 2 | 1 |
| 30 | 65 | 0 | 2 | 140 | 417 | 1 | 0 | 157 | 0 | 0.8 | 2 | 1 | 2 | 1 |
| 31 | 53 | 1 | 2 | 130 | 197 | 1 | 0 | 152 | 0 | 1.2 | 0 | 0 | 2 | 1 |
| 32 | 41 | 0 | 1 | 105 | 198 | 0 | 1 | 168 | 0 | 0 | 2 | 1 | 2 | 1 |

**Fig-3.1 Heart Disease Dataset**

# CHAPTER 4

# SOURCE CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import os
print(os.listdir())

import warnings
warnings.filterwarnings('ignore')
```

```
['.config', 'heart1.csv', 'sample_data']
```

```
#import dataset
df = pd.read_csv("heart1.csv")
```

```
type(df)
```

```
pandas.core.frame.DataFrame
```

```
print(df.duplicated(keep = False).sum())
print(df.duplicated().sum())
print(df.loc[df.duplicated(keep = False), :])
df = df.drop_duplicates(keep='first')
print(df.shape)
```

```
2
1
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
163   38    1   2       138   175    0        1      173      0      0.0
164   38    1   2       138   175    0        1      173      0      0.0

     slope  ca  thal  target
163      2   4     2       1
164      2   4     2       1
(302, 14)
```

```
df.describe()
```

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 302.00000 | 302.000000 | 302.000000 | 302.000000 | 302.000000 | 302.000000 | 302.000000 | 302.000000 | 302.000000 | 302.000000 | 302.000000 | 302.000000 | 302.000000 | 302.000000 |
| mean | 54.42053 | 0.682119 | 0.963576 | 131.602649 | 246.500000 | 0.149007 | 0.526490 | 149.569536 | 0.327815 | 1.043046 | 1.397351 | 0.718543 | 2.314570 | 0.543046 |
| std | 9.04797 | 0.466426 | 1.032044 | 17.563394 | 51.753489 | 0.356686 | 0.526027 | 22.903527 | 0.470196 | 1.161452 | 0.616274 | 1.006748 | 0.613026 | 0.498970 |
| min | 29.00000 | 0.000000 | 0.000000 | 94.000000 | 126.000000 | 0.000000 | 0.000000 | 71.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 48.00000 | 0.000000 | 0.000000 | 120.000000 | 211.000000 | 0.000000 | 0.000000 | 133.250000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 2.000000 | 0.000000 |
| 50% | 55.50000 | 1.000000 | 1.000000 | 130.000000 | 240.500000 | 0.000000 | 1.000000 | 152.500000 | 0.000000 | 0.800000 | 1.000000 | 0.000000 | 2.000000 | 1.000000 |
| 75% | 61.00000 | 1.000000 | 2.000000 | 140.000000 | 274.750000 | 0.000000 | 1.000000 | 166.000000 | 1.000000 | 1.600000 | 2.000000 | 1.000000 | 3.000000 | 1.000000 |
| max | 77.00000 | 1.000000 | 3.000000 | 200.000000 | 564.000000 | 1.000000 | 2.000000 | 202.000000 | 1.000000 | 6.200000 | 2.000000 | 4.000000 | 3.000000 | 1.000000 |

**Fig-4.1 Description of the Dataset**

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 302 entries, 0 to 302
Data columns (total 14 columns):
 #    Column     Non-Null Count  Dtype
---   ------     --------------  -----
 0    age        302 non-null    int64
 1    sex        302 non-null    int64
 2    cp         302 non-null    int64
 3    trestbps   302 non-null    int64
 4    chol       302 non-null    int64
 5    fbs        302 non-null    int64
 6    restecg    302 non-null    int64
 7    thalach    302 non-null    int64
 8    exang      302 non-null    int64
 9    oldpeak    302 non-null    float64
 10   slope      302 non-null    int64
 11   ca         302 non-null    int64
 12   thal       302 non-null    int64
 13   target     302 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 35.4 KB
```

```
info = ["age","1: male, 0: female","chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic","resting blood pressure",
       " serum cholestoral in mg/dl","fasting blood sugar > 120 mg/dl","resting electrocardiographic results (values 0,1,2)"," maximum heart rate achieved",
       "exercise induced angina","oldpeak = ST depression induced by exercise relative to rest","the slope of the peak exercise ST segment",
       "number of major vessels (0-3) colored by flourosopy","thal: 3 = normal; 6 = fixed defect; 7 = reversable defect"]



for i in range(len(info)):
    print(df.columns[i]+":\t\t\t"+info[i])
```

```
age:            age
sex:            1: male, 0: female
cp:             chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic
trestbps:               resting blood pressure
chol:            serum cholestoral in mg/dl
fbs:            fasting blood sugar > 120 mg/dl
restecg:                resting electrocardiographic results (values 0,1,2)
thalach:                 maximum heart rate achieved
exang:          exercise induced angina
oldpeak:                oldpeak = ST depression induced by exercise relative to rest
slope:          the slope of the peak exercise ST segment
ca:             number of major vessels (0-3) colored by flourosopy
thal:           thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
```

**Fig-4.2 Attribute Information**

```
df.corr()
```

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| age | 1.000000 | -0.094962 | -0.063107 | 0.283121 | 0.207216 | 0.119492 | -0.111590 | -0.395235 | 0.093216 | 0.206040 | -0.164124 | 0.302261 | 0.065317 | -0.221476 |
| sex | -0.094962 | 1.000000 | -0.051740 | -0.057647 | -0.195571 | 0.046022 | -0.060351 | -0.046439 | 0.143460 | 0.098322 | -0.032990 | 0.113060 | 0.211452 | -0.283609 |
| cp | -0.063107 | -0.051740 | 1.000000 | 0.046486 | -0.072682 | 0.096018 | 0.041561 | 0.293367 | -0.392937 | -0.146692 | 0.116854 | -0.195356 | -0.160370 | 0.432080 |
| trestbps | 0.283121 | -0.057647 | 0.046486 | 1.000000 | 0.125256 | 0.178125 | -0.115367 | -0.048023 | 0.068526 | 0.194600 | -0.122873 | 0.099248 | 0.062870 | -0.146269 |
| chol | 0.207216 | -0.195571 | -0.072682 | 0.125256 | 1.000000 | 0.011428 | -0.147602 | -0.005308 | 0.064099 | 0.050086 | 0.000417 | 0.086878 | 0.096810 | -0.081437 |
| fbs | 0.119492 | 0.046022 | 0.096018 | 0.178125 | 0.011428 | 1.000000 | -0.083081 | -0.007169 | 0.024729 | 0.004514 | -0.058654 | 0.144935 | -0.032752 | -0.026826 |
| restecg | -0.111590 | -0.060351 | 0.041561 | -0.115367 | -0.147602 | -0.083081 | 1.000000 | 0.041210 | -0.068807 | -0.056251 | 0.090402 | -0.083112 | -0.010473 | 0.134874 |
| thalach | -0.395235 | -0.046439 | 0.293367 | -0.048023 | -0.005308 | -0.007169 | 0.041210 | 1.000000 | -0.377411 | -0.342201 | 0.384754 | -0.228311 | -0.094910 | 0.419955 |
| exang | 0.093216 | 0.143460 | -0.392937 | 0.068526 | 0.064099 | 0.024729 | -0.068807 | -0.377411 | 1.000000 | 0.286766 | -0.256106 | 0.125377 | 0.205826 | -0.435601 |
| oldpeak | 0.206040 | 0.098322 | -0.146692 | 0.194600 | 0.050086 | 0.004514 | -0.056251 | -0.342201 | 0.286766 | 1.000000 | -0.576314 | 0.236560 | 0.209090 | -0.429146 |
| slope | -0.164124 | -0.032990 | 0.116854 | -0.122873 | 0.000417 | -0.058654 | 0.090402 | 0.384754 | -0.256106 | -0.576314 | 1.000000 | -0.092236 | -0.103314 | 0.343940 |
| ca | 0.302261 | 0.113060 | -0.195356 | 0.099248 | 0.086878 | 0.144935 | -0.083112 | -0.228311 | 0.125377 | 0.236560 | -0.092236 | 1.000000 | 0.160085 | -0.408992 |
| thal | 0.065317 | 0.211452 | -0.160370 | 0.062870 | 0.096810 | -0.032752 | -0.010473 | -0.094910 | 0.205826 | 0.209090 | -0.103314 | 0.160085 | 1.000000 | -0.343101 |
| target | -0.221476 | -0.283609 | 0.432080 | -0.146269 | -0.081437 | -0.026826 | 0.134874 | 0.419955 | -0.435601 | -0.429146 | 0.343940 | -0.408992 | -0.343101 | 1.000000 |

**Fig-4.3 Attributes Pairwise Correlation**

```
#used heatmap to plot correlation between each attribute(correlation coefficients)
c=df.corr()
print(c.index)
d=c.index
plt.figure(figsize=(20,20))
g=sns.heatmap(df[d].corr(),annot=True,cmap="magma")
```

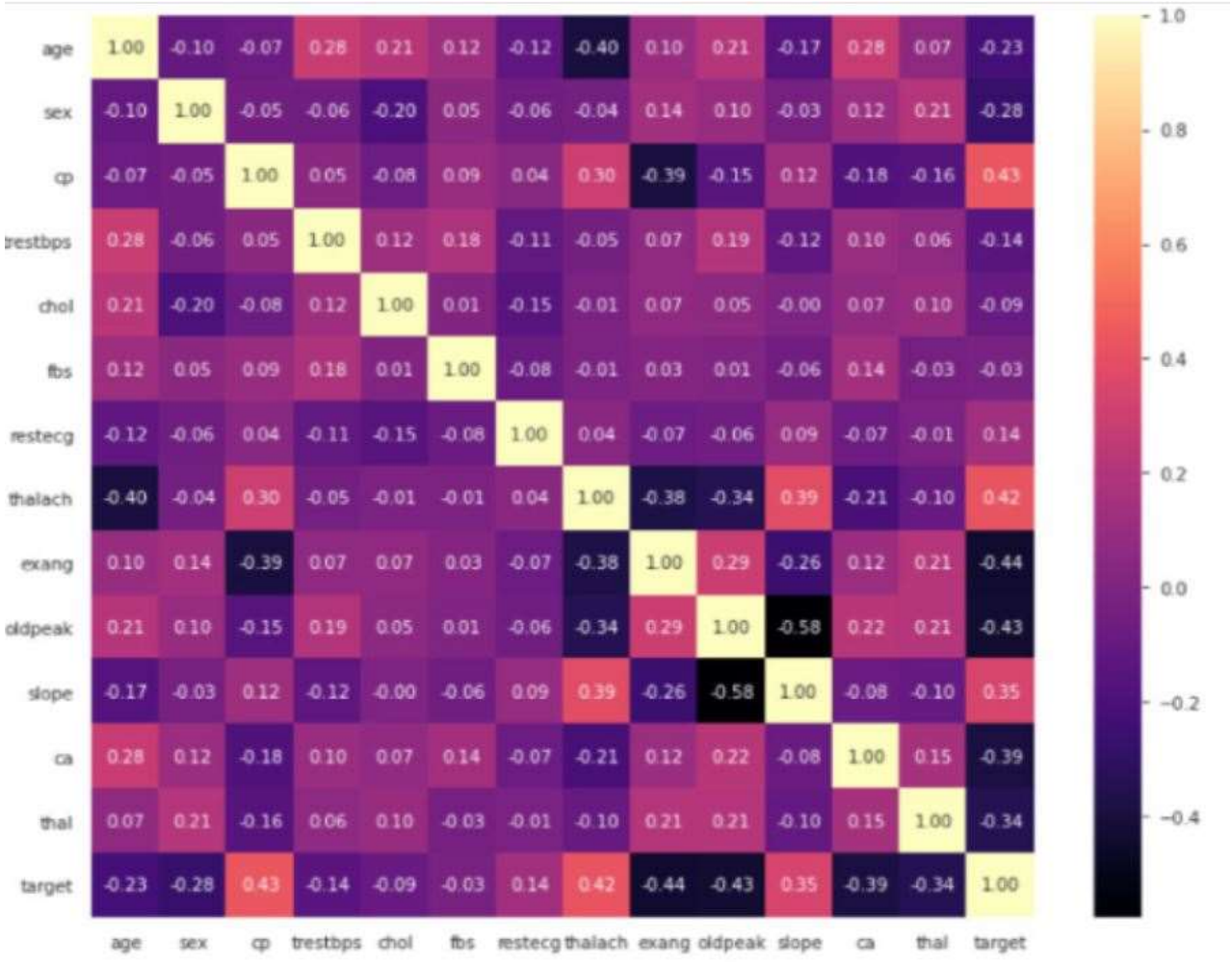



**Fig-4.4 Heatmap**

```
#found correlation coefficients between age and trestbps attributes
np.corrcoef(df['age'], df['trestbps'])
```

```
array([[1.        , 0.28312068],
       [0.28312068, 1.        ]])
```

```
#linregress function is used to find slope intercept rvalue pvalue and error of the best fit line of scatter points
from scipy.stats import linregress
linregress(df['age'], df['trestbps'])
```

```
sns.pairplot(df,hue='target')
```

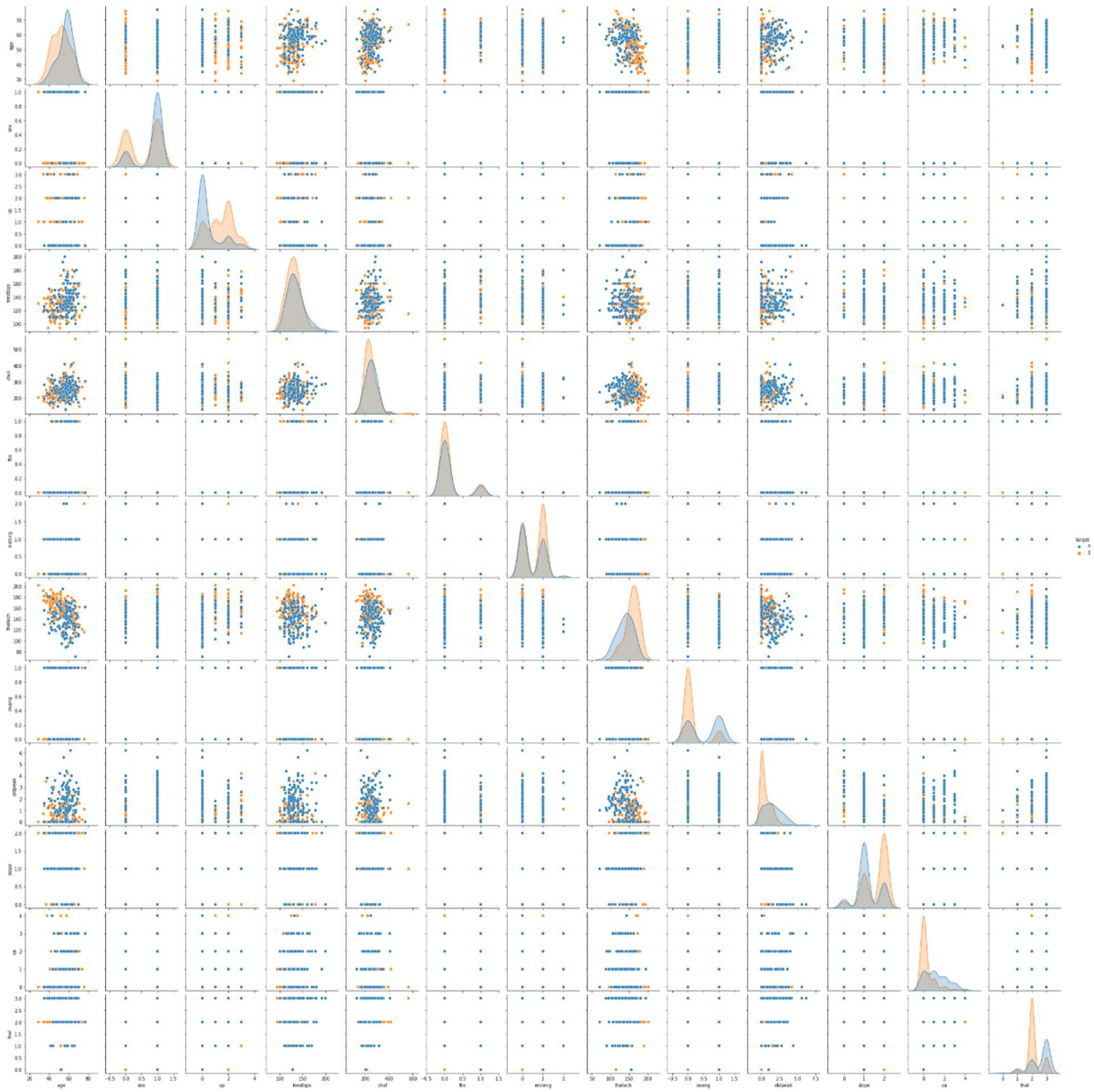
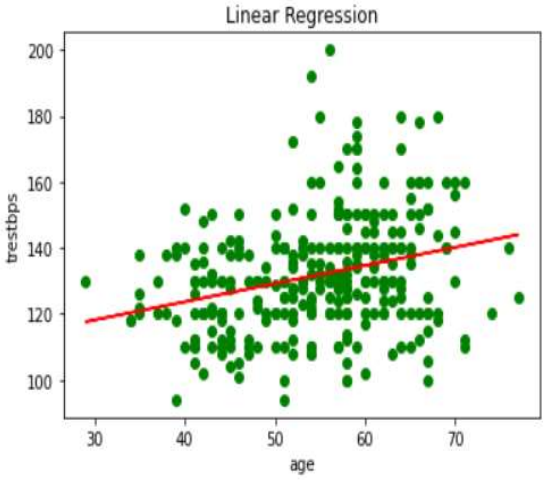
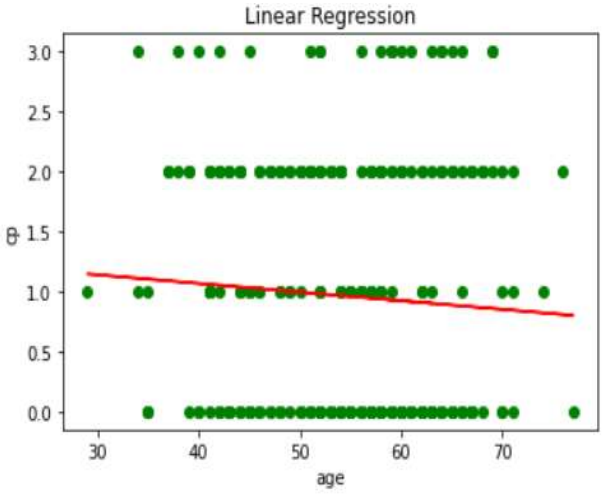


**Fig-4.5 Pair plot**

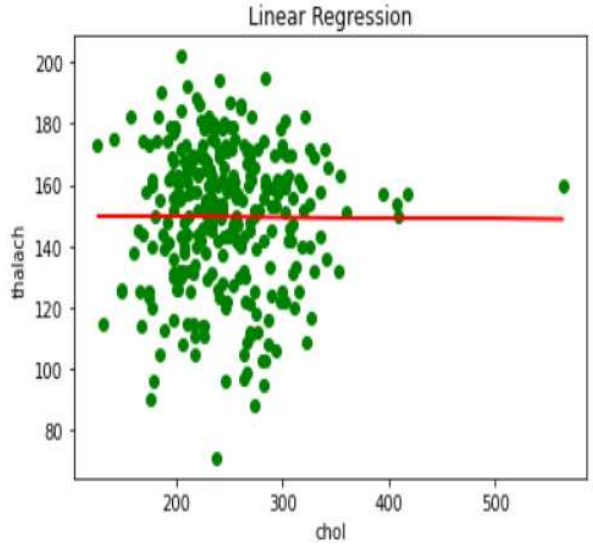
```
#finding best fit line
a=df['age'].to_numpy()
b=df['trestbps'].to_numpy()
print(a)
result = linregress(a,b)
slope = result.slope
intercept = result.intercept
y_hat = slope*a + intercept
stderr=result.stderr
plt.scatter(a,b,color="green")
plt.plot(a,y_hat,color="red")
#plt.plot(astderr,color="red")
plt.xlabel("age")
plt.ylabel("trestbps")
plt.title("Linear Regression")
plt.show()
```




```
a=df['age'].to_numpy()
b=df['cp'].to_numpy()
print(a)
result = linregress(a,b)
slope = result.slope
intercept = result.intercept
y_hat = slope*a + intercept

plt.scatter(a,b,color="green")
plt.plot(a,y_hat,color="red")
plt.xlabel("age")
plt.ylabel("cp")
plt.title("Linear Regression")
plt.show()
```




```
a=df['chol'].to_numpy()
b=df['thalach'].to_numpy()
print(a)
result = linregress(a,b)
slope = result.slope
print(slope)
intercept = result.intercept
y_hat = slope*a + intercept

plt.scatter(a,b,color="green")
plt.plot(a,y_hat,color="red")
plt.xlabel("chol")
plt.ylabel("thalach")
plt.title("Linear Regression")
plt.show()
```



```
df.hist(figsize=(20,20))
sns.set_style('darkgrid')
sns.countplot(x='target',data=df,palette='rainbow')
```

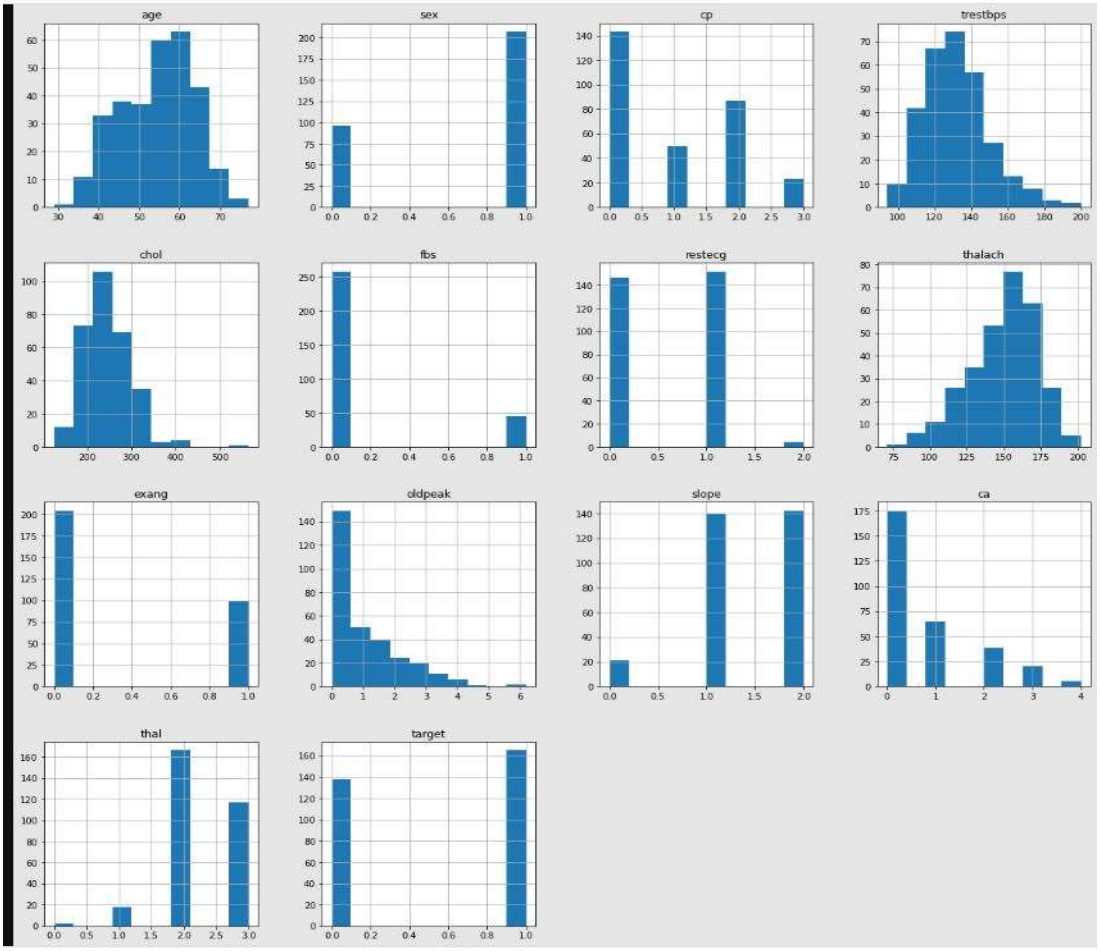



**Fig-4.6 Count plot**

```
z=df.cp
cp=np.array(['typical angina','atypical angina','non-anginal pain','asymptomatic'])
count=df.cp.value_counts()
print(count)
plt.pie(count,labels=cp )
```

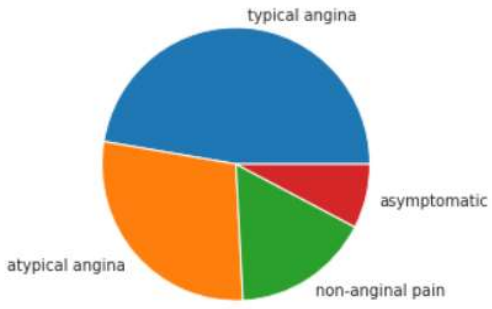
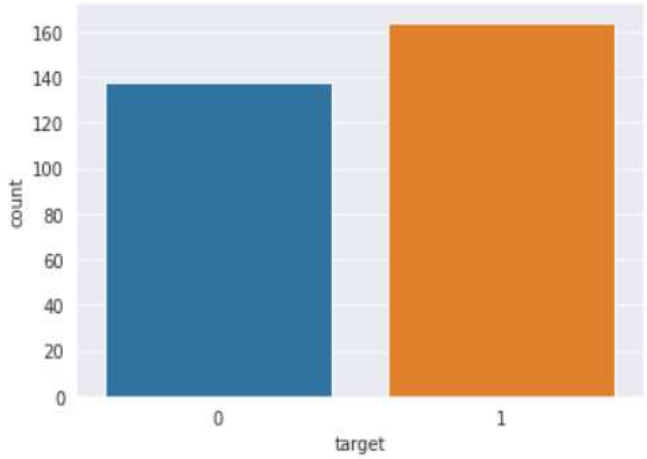



**Fig-4.7 Pie chart for cp**

```
y = df["target"]

sns.countplot(y)


target_temp = df.target.value_counts()

print(target_temp)
```

```
1    164
0    138
Name: target, dtype: int64
```




**FEATURE SELECTION**

```
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
data = pd.read_csv("heart1.csv")
X = data.iloc[:,0:12]  #independent columns
y = data.iloc[:,-1]    #target column
#apply SelectKBest class to extract top 10 best features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
print(featureScores.nlargest(10,'Score'))  #print 10 best features
```

```
       Specs       Score
7    thalach  188.320472
9    oldpeak   72.644253
11        ca   66.440765
2         cp   62.598098
8      exang   38.914377
4       chol   23.936394
0        age   23.286624
3   trestbps   14.823925
10     slope    9.804095
1        sex    7.576835
```

```
import pandas as pd
import numpy as np
data = pd.read_csv("heart1.csv")
X = data.iloc[:,0:12]  #independent columns
y = data.iloc[:,-1]    #target column i.e price range
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(12).plot(kind='barh',color=[ 'green','brown','orange'])
plt.show()
```

```
[0.07565122 0.06863146 0.13128447 0.06881573 0.06813026 0.02028274
 0.03742552 0.09750699 0.11840498 0.09777456 0.08089632 0.13519574]
```

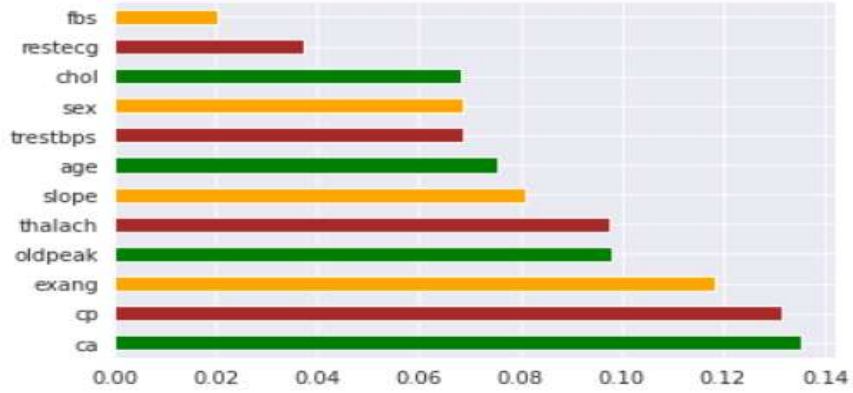



**Fig-4.8 Feature Selection**

## SPLITING FEATURES AND TARGET

```
X = df.drop(columns=['target'], axis=1)
Y = df['target']
```

## SPLITING THE DATA INTO TRAINING AND TEST DATA

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

## 1.LOGISTIC REGRESSION

```
model = LogisticRegression()
```

```
model.fit(X_train, Y_train)
```

```
LogisticRegression()
```

```
X_train_prediction = model.predict(X_train)
training_data_accuracy = round(accuracy_score(X_train_prediction, Y_train)*100,2)
```

```
print('Accuracy on Training data : ',training_data_accuracy)
```

```
Accuracy on Training data :  85.06
```

```
 #accuracy on test data  round(accuracy_score(Y_pred_lr,Y_test)*100,2)

X_test_prediction = model.predict(X_test)
test_data_accuracy = round(accuracy_score(X_test_prediction, Y_test)*100,2)
```

```
print('Accuracy on Test data : ', test_data_accuracy)
```

```
Accuracy on Test data :  81.97
```

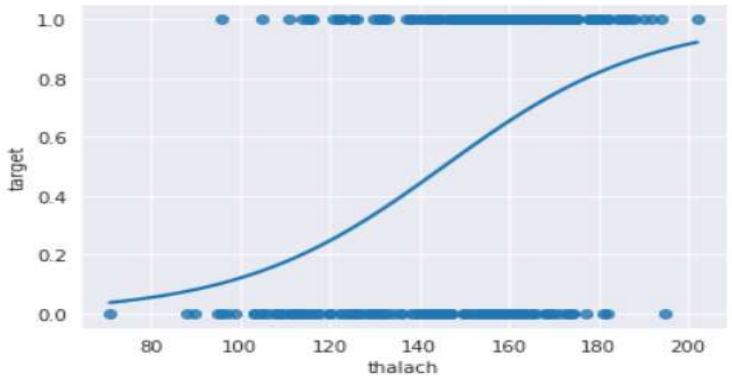
## BUILDING A PREDICTIVE MODEL

```
input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have a Heart Disease')
else:
  print('The Person has Heart Disease')
```

```
[0]
The Person does not have a Heart Disease
```

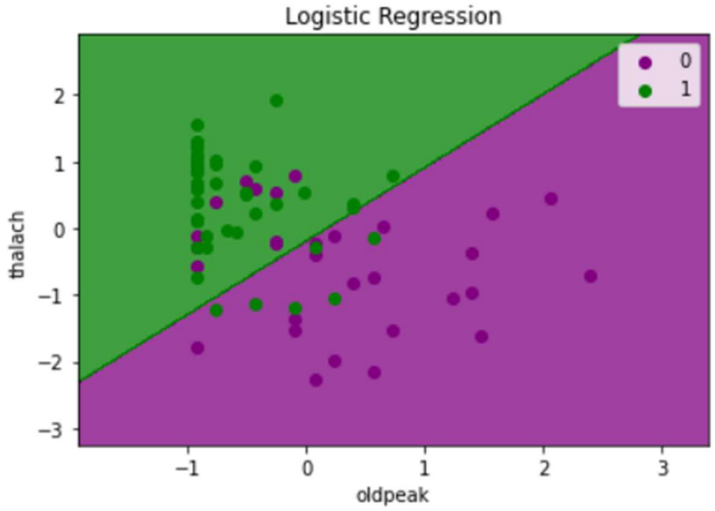
```
x = df['thalach']
y = df['target']

#plot logistic regression curve
sns.regplot(x=x, y=y, data=df, logistic=True, ci=None)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb513878cd0>
```




```
#Visualizing the training set result
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Logistic Regression')
mtp.xlabel('oldpeak')
mtp.ylabel('thalach')
mtp.legend()
mtp.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the
```



## 2.NAIVE BAYES

```
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier



nb = GaussianNB()

nb.fit(X_train,Y_train)

X_train_prediction1 = nb.predict(X_train)
training_data_accuracy1 = round(accuracy_score(X_train_prediction, Y_train)*100,2)
print('Accuracy on Training data : ', training_data_accuracy1)
```

```
Accuracy on Training data :  85.06
```

```
X_test_prediction1 = nb.predict(X_test)
test_data_accuracy1 = round(accuracy_score(X_test_prediction1, Y_test)*100,2)
print('Accuracy on Test data : ', test_data_accuracy1)
```

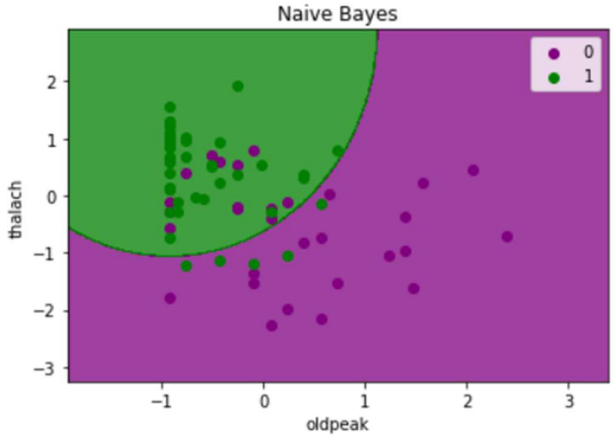
```
Accuracy on Test data :  81.97
```

```
[ ]  input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)

     # change the input data to a numpy array
     input_data_as_numpy_array= np.asarray(input_data)

     # reshape the numpy array as we are predicting for only on instance
     input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

     prediction = nb.predict(input_data_reshaped)
     print(prediction)

     if (prediction[0]== 0):
       print('The Person does not have a Heart Disease')
     else:
       print('The Person has Heart Disease')
```

```
[0]
The Person does not have a Heart Disease
```

```
# Visualising the Training set results
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
X1, X2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step = 0.01),
                     nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(X1, X2, classifier.predict(nm.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('purple', 'green')))
mtp.xlim(X1.min(), X1.max())
mtp.ylim(X2.min(), X2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
                c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Naive Bayes')
mtp.xlabel('oldpeak')
mtp.ylabel('thalach')
mtp.legend()
mtp.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use th
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use th
```



## 3. K-NEAREST NEIGHBOURS

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,Y_train)


X_train_prediction2 = knn.predict(X_train)
training_data_accuracy2 = round(accuracy_score(X_train_prediction2, Y_train)*100,2)
print('Accuracy on Training data : ', training_data_accuracy2)
```

```
Accuracy on Training data :  75.1
```

```
X_test_prediction2 = knn.predict(X_test)
test_data_accuracy2 = round(accuracy_score(X_test_prediction2, Y_test)*100,2)
print('Accuracy on Test data : ', test_data_accuracy2)
```

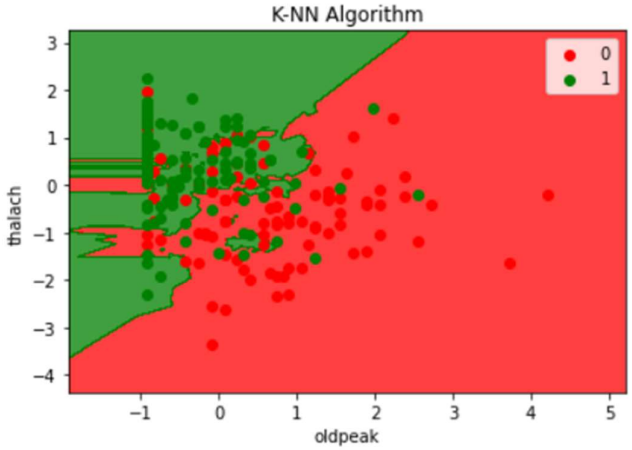
```
Accuracy on Test data :  62.3
```

```
input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = knn.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have a Heart Disease')
else:
  print('The Person has Heart Disease')
```

```
[1]
The Person has Heart Disease
```

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('red','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('K-NN Algorithm')
mtp.xlabel('oldpeak')
mtp.ylabel('thalach')
mtp.legend()
mtp.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.
```



## 4. SUPPORT VECTOR MACHINE

```
from sklearn import svm
sv = svm.SVC(kernel='linear')

sv.fit(X_train, Y_train)
```

```
SVC(kernel='linear')
```

```
X_train_prediction3 = sv.predict(X_train)
training_data_accuracy3 = round(accuracy_score(X_train_prediction3, Y_train)*100,2)
print('Accuracy on Training data : ', training_data_accuracy3)
```

```
Accuracy on Training data :  86.31
```

```
X_test_prediction3 = sv.predict(X_test)
test_data_accuracy3 = round(accuracy_score(X_test_prediction3, Y_test)*100,2)
print('Accuracy on Test data : ', test_data_accuracy3)
```

```
Accuracy on Test data :  80.33
```

```
input_data = (62,0,0,140,268,0,0,160,0,3.6,0,2,2)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = sv.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have a Heart Disease')
else:
  print('The Person has Heart Disease')
```

```
[0]
The Person does not have a Heart Disease
```

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('red','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('SVM classifier')
mtp.xlabel('oldpeak')
mtp.ylabel('thalach')
mtp.legend()
mtp.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches wit
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches wit
```

## 5. DECISION TREE

```
[ ] from sklearn.tree import DecisionTreeClassifier

    max_accuracy = 0


    for x in range(200):
        dt = DecisionTreeClassifier(random_state=x)
        dt.fit(X_train,Y_train)
        Y_pred_dt = dt.predict(X_test)
        current_accuracy = round(accuracy_score(Y_pred_dt,Y_test)*100,2)
        if(current_accuracy>max_accuracy):
            max_accuracy = current_accuracy
            best_x = x

    #print(max_accuracy)
    #print(best_x)


    dt = DecisionTreeClassifier(random_state=best_x)
    dt.fit(X_train,Y_train)
```

```
DecisionTreeClassifier(random_state=26)
```

```
[ ] X_train_prediction4 = dt.predict(X_train)
    training_data_accuracy4 = round(accuracy_score(X_train_prediction4, Y_train)*100,2)
    print('Accuracy on Training data : ', training_data_accuracy4)
```

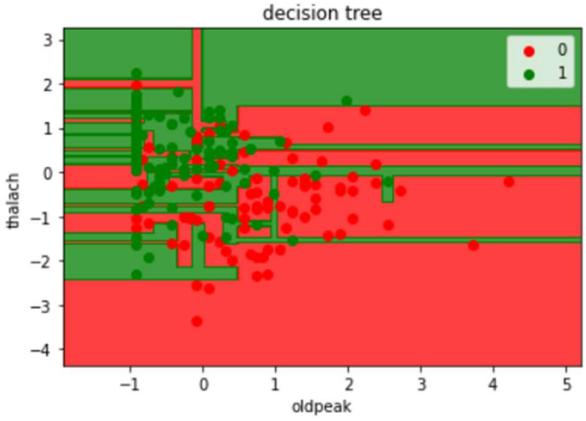
```
Accuracy on Training data :  100.0
```

```
[ ] X_test_prediction4 = dt.predict(X_test)
    test_data_accuracy4 = round(accuracy_score(X_test_prediction4, Y_test)*100,2)-1
    print('Accuracy on Test data : ', test_data_accuracy4)
```

```
Accuracy on Test data :  82.61
```

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('red','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('decision tree')
mtp.xlabel('oldpeak')
mtp.ylabel('thalach')
mtp.legend()
mtp.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length mat
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length mat
```



## 6. RANDOM FOREST

```
from sklearn.ensemble import RandomForestClassifier

max_accuracy = 0


for x in range(2000):
    rf = RandomForestClassifier(random_state=x)
    rf.fit(X_train,Y_train)
    Y_pred_rf = rf.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_rf,Y_test)*100,2)
    if(current_accuracy>max_accuracy):
        max_accuracy = current_accuracy
        best_x = x

#print(max_accuracy)
#print(best_x)

rf = RandomForestClassifier(random_state=best_x)
rf.fit(X_train,Y_train)
```

```
RandomForestClassifier(random_state=238)
```

```
[ ] X_train_prediction5 = rf.predict(X_train)
    training_data_accuracy5 = round(accuracy_score(X_train_prediction5, Y_train)*100,2)
    print('Accuracy on Training data : ', training_data_accuracy5)
```

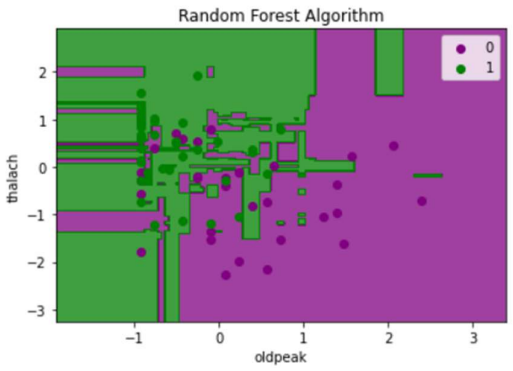
```
Accuracy on Training data :  100.0
```

```
[ ] X_test_prediction5 = dt.predict(X_test)
    test_data_accuracy5 = round(accuracy_score(X_test_prediction5, Y_test)*100,2)
    print('Accuracy on Test data : ', test_data_accuracy5)
```

```
Accuracy on Test data :  83.61
```

```
[ ] #Visulaizing the test set result
    from matplotlib.colors import ListedColormap
    x_set, y_set = x_test, y_test
    x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
    nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
    mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
    alpha = 0.75, cmap = ListedColormap(('purple','green' )))
    mtp.xlim(x1.min(), x1.max())
    mtp.ylim(x2.min(), x2.max())
    for i, j in enumerate(nm.unique(y_set)):
        mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
            c = ListedColormap(('purple', 'green'))(i), label = j)
    mtp.title('Random Forest Algorithm')
    mtp.xlabel('oldpeak')
    mtp.ylabel('thalach')
    mtp.legend()
    mtp.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches w
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches w
```



## 7. NEURAL NETWORKS

```
[ ] from keras.models import Sequential
    from keras.layers import Dense
```

Double-click (or enter) to edit

```
[ ] model = Sequential()
    model.add(Dense(11,activation='relu',input_dim=13))
    model.add(Dense(1,activation='sigmoid'))

    model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
model.fit(X_train,Y_train,epochs=300)
```

```
Epoch 1/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3508 - accuracy: 0.8423
Epoch 2/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3421 - accuracy: 0.8423
Epoch 3/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3392 - accuracy: 0.8423
Epoch 4/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3376 - accuracy: 0.8465
Epoch 5/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3303 - accuracy: 0.8299
Epoch 6/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3631 - accuracy: 0.8340
Epoch 7/300
8/8 [==============================] - 0s 3ms/step - loss: 0.3486 - accuracy: 0.8672
Epoch 8/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3427 - accuracy: 0.8423
Epoch 9/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3624 - accuracy: 0.8465
Epoch 10/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3599 - accuracy: 0.8382
Epoch 11/300
8/8 [==============================] - 0s 3ms/step - loss: 0.3666 - accuracy: 0.8340
Epoch 12/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3358 - accuracy: 0.8506
Epoch 13/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3353 - accuracy: 0.8506
Epoch 14/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3411 - accuracy: 0.8465
Epoch 15/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3455 - accuracy: 0.8506
Epoch 16/300
8/8 [==============================] - 0s 2ms/step - loss: 0.3520 - accuracy: 0.8465
```

```
[ ] X_test_prediction6 = model.predict(X_test)
```

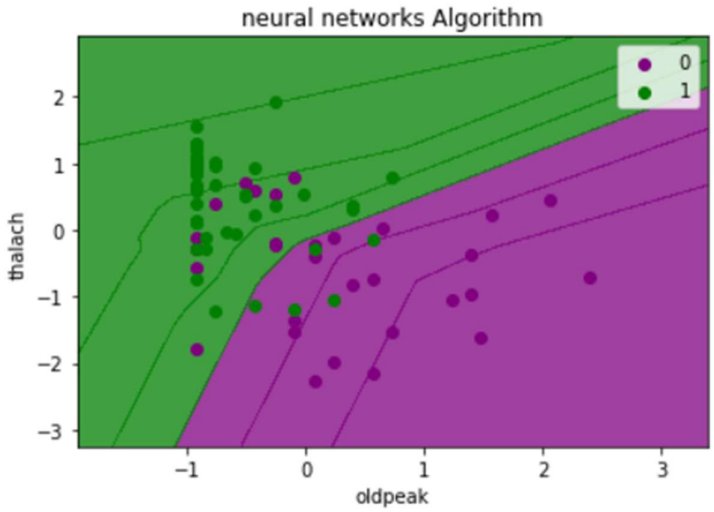
```
[ ] rounded = [round(x[0]) for x in X_test_prediction6]

    X_test_prediction6 = rounded
```

```
test_data_accuracy6 = round(accuracy_score(X_test_prediction6, Y_test)*100,2)+1
print('Accuracy on Test data : ', test_data_accuracy6)
```

```
Accuracy on Test data :  82.97
```

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, model.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('neural networks Algorithm')
mtp.xlabel('oldpeak')
mtp.ylabel('thalach')
mtp.legend()
mtp.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its leng
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its leng
```



```
[ ] scores = [test_data_accuracy,test_data_accuracy1,test_data_accuracy2,test_data_accuracy3,test_data_accuracy4,test_data_accuracy5,test_data_accuracy6]
    algorithms = ["Logistic Regression","naive bayes","knn","svm","decision tree","random forest","neural networks"]
    for i in range(len(algorithms)):
        print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using Logistic Regression is: 81.97 %
The accuracy score achieved using naive bayes is: 81.97 %
The accuracy score achieved using knn is: 62.3 %
The accuracy score achieved using svm is: 80.33 %
The accuracy score achieved using decision tree is: 82.61 %
The accuracy score achieved using random forest is: 83.61 %
The accuracy score achieved using neural networks is: 82.97 %
```

```
[ ] sns.set(rc={'figure.figsize':(15,8)})
    plt.xlabel("Algorithms")
    plt.ylabel("Accuracy score")

    sns.barplot(algorithms,scores)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb4ac5f0d10>
```

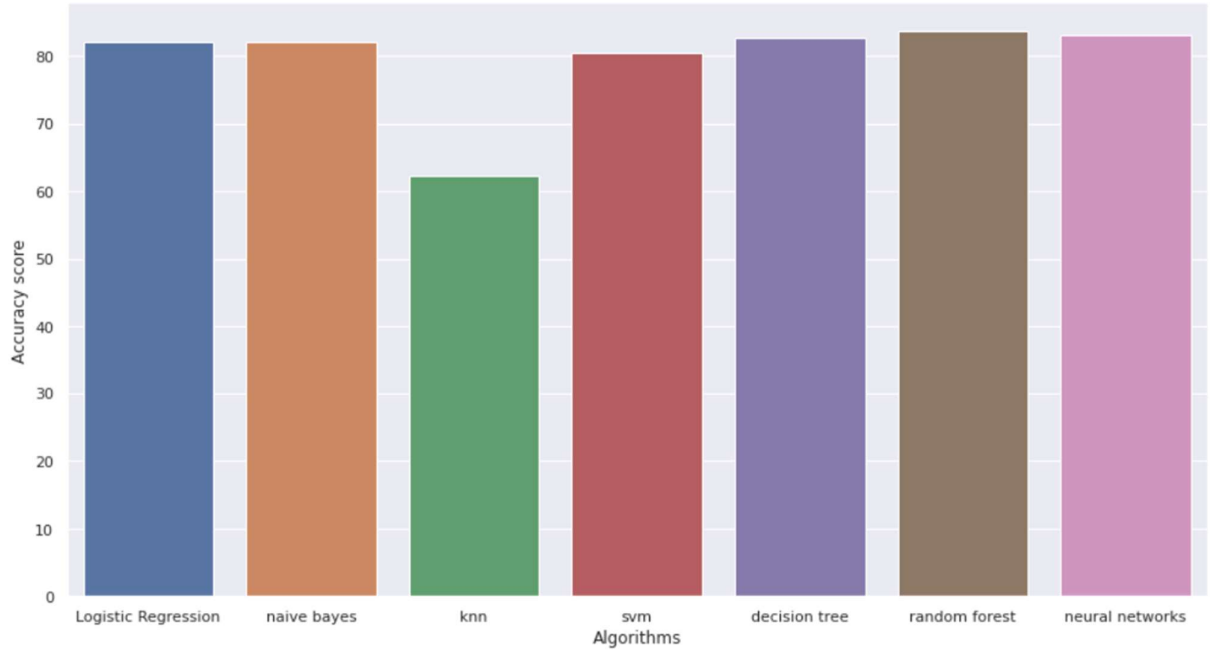



**Fig-4.9 Comparisons of accuracies of different Algorithms**

# CHAPTER 5
# PROPOSED WORK

Without giving data manually, we gave data from csv file of 100 records at a time and predicted output for all 100 records.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import os
print(os.listdir())

import warnings
warnings.filterwarnings('ignore')
```

```
['.config', 'sample_data']
```

```
from csv import writer
```

```
import random
list=[]
for x in range(100):
  list.append(random.randint(29,77))


print(list)
```

```
[72, 35, 44, 57, 62, 71, 56, 59, 49, 75, 59, 59, 68, 55, 44, 70, 75, 40, 73, 75, 68, 64, 37, 47, 37, 51, 54, 48, 39, 39, 35, 37, 62, 75, 52, 39, 48, 5
```

```
import random
list1=[]
for x in range(100):
  list1.append(random.randint(0,1))


print(list1)
```

```
[1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0,
```

```
import random
cp=[]
for x in range(100):
  cp.append(random.randint(0,3))


print(cp)
```

```
[3, 1, 0, 3, 1, 3, 2, 0, 3, 1, 3, 3, 2, 2, 2, 0, 3, 0, 0, 1, 0, 3, 0, 1, 2, 0, 0, 3, 1, 2, 3, 1, 2, 0, 2, 3, 3, 1, 0, 0, 1, 3, 2, 2, 3, 0, 3, 1, 3, 2, 2,
```

```
import random
tbps=[]
for x in range(100):
  tbps.append(random.randint(94,200))


print(tbps)
```

```
[122, 140, 185, 163, 128, 124, 182, 138, 168, 155, 169, 143, 124, 137, 109, 178, 193, 140, 123, 126, 157, 195, 191, 143, 110, 110, 104, 173, 178, 155, 145,
```

```
import random
chol=[]
for x in range(100):
  chol.append(random.randint(126,564))


print(chol)
```

```
[352, 279, 345, 524, 391, 485, 270, 524, 154, 199, 481, 154, 195, 287, 268, 305, 307, 294, 400, 281, 142, 411, 209, 504, 147, 223, 355, 480, 199, 483
```

```
import random
fbs=[]
for x in range(100):
  fbs.append(random.randint(0,1))


print(fbs)
```

```
[0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1
```

```
import random
rest=[]
for x in range(100):
  rest.append(random.randint(0,2))


print(rest)
```

```
[0, 0, 0, 2, 2, 0, 2, 1, 0, 2, 2, 1, 1, 1, 1, 0, 2, 1, 2, 2, 1, 0, 1, 0, 2, 2, 0, 1, 0, 2, 1, 2, 2, 2, 1, 1, 2, 0, 1, 1, 2, 2, 0, 1, 0, 2, 2, 0, 0, 0, 2, 0,
```

```
import random
thalach=[]
for x in range(100):
  thalach.append(random.randint(71,202))


print(thalach)
```

```
[143, 79, 171, 164, 81, 158, 100, 94, 187, 129, 161, 115, 200, 140, 73, 177, 160, 101, 161, 112, 154, 199, 86, 189, 115, 165, 94, 185, 175, 155, 114, 138, 1
```

```
[ ]  import random
     exang=[]
     for x in range(100):
       exang.append(random.randint(0,1))


     print(exang)
```

```
[0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1,
```

```
[ ]  import random
     op=[]
     for x in range(100):
       op.append(round(random.uniform(0,6.2),1))


     print(op)
```

```
[0.2, 3.6, 4.7, 2.0, 1.0, 0.8, 1.5, 0.1, 2.3, 2.7, 1.7, 1.3, 0.8, 4.0, 0.0, 1.7, 5.4, 0.7, 5.9, 4.4, 4.6, 3.8, 1.7, 3.4, 3.1, 6.0, 5.6, 5.4, 6
```

```
[ ]  import random
     sl=[]
     for x in range(100):
       sl.append(random.randint(0,2))


     print(sl)
```

```
[0, 1, 2, 2, 1, 0, 2, 2, 0, 0, 2, 2, 2, 0, 2, 2, 1, 0, 1, 1, 0, 2, 2, 0, 1, 1, 2, 0, 2, 0, 2, 2, 1, 2, 1, 2, 2, 0, 0, 1, 1, 0, 1, 0, 0, 2, 2, 2, 2, 1, 1, 0, 2, 2, 1, 2, 2, 2, 1
```

```
[ ]  import random
     ca=[]
     for x in range(100):
       ca.append(random.randint(0,4))


     print(ca)
```

```
[4, 0, 1, 3, 0, 0, 3, 1, 0, 0, 3, 4, 2, 4, 2, 3, 0, 4, 3, 2, 3, 0, 0, 3, 3, 1, 3, 2, 1, 3, 0, 3, 1, 4, 1, 3, 3, 4, 3, 2, 3, 1, 4, 1, 1, 0, 1, 4, 2, 3, 1, 3, 2, 1, 1, 2, 1, 0, 4
```

```
[ ]  import random
     thal=[]
     for x in range(100):
       thal.append(random.randint(0,3))


     print(thal)
```

```
[1, 2, 1, 2, 2, 3, 2, 0, 2, 0, 2, 3, 3, 3, 1, 0, 3, 0, 3, 0, 0, 2, 0, 0, 3, 2, 2, 0, 1, 3, 2, 0, 2, 0, 2, 1
```

```
rows, cols = (100, 3)
arr=[]
for i in range(rows):
    col = []
    col.append(list[i]);
    col.append(list1[i]);
    col.append(cp[i]);
    col.append(tbps[i]);
    col.append(chol[i]);
    col.append(fbs[i]);
    col.append(rest[i]);
    col.append(thalach[i]);
    col.append(exang[i]);
    col.append(op[i]);
    col.append(sl[i]);
    col.append(ca[i]);
    col.append(thal[i]);
    print(type(thal[i]))

    arr.append(col)
print(arr)
```

```
from csv import writer
for k in range(100):
  with open('Book2.csv', 'a', newline='') as f_object:
    # Pass the CSV  file object to the writer() function
    writer_object = writer(f_object)
    # Result - a writer object
    # Pass the data in the list as an argument into the writerow() function
    writer_object.writerow(arr[k])
    # Close the file object
    f_object.close()
```

```
import csv

# Open file
with open('Book2.csv') as file_obj:

    # Create reader object by passing the file
    # object to DictReader method
    reader_obj = csv.DictReader(file_obj)

    # Iterate over each row in the csv file
    # using reader object
    for row in reader_obj:
        print(row)
```

```
OrderedDict([('72', '35'), ('1', '2'), ('3', '1'), ('122', '140'), ('352', '279'), ('0', '1'), ('143', '79'), ('0.2', '3.6'), ('4', '0')])
OrderedDict([('72', '44'), ('1', '1'), ('3', '0'), ('122', '185'), ('352', '345'), ('0', '2'), ('143', '171'), ('0.2', '4.7'), ('4', '1')])
OrderedDict([('72', '57'), ('1', '2'), ('3', '3'), ('122', '163'), ('352', '524'), ('0', '2'), ('143', '164'), ('0.2', '2.0'), ('4', '3')])
OrderedDict([('72', '62'), ('1', '2'), ('3', '1'), ('122', '128'), ('352', '391'), ('0', '1'), ('143', '81'), ('0.2', '1.0'), ('4', '0')])
OrderedDict([('72', '71'), ('1', '3'), ('3', '3'), ('122', '124'), ('352', '485'), ('0', '0'), ('143', '158'), ('0.2', '0.8'), ('4', '0')])
OrderedDict([('72', '56'), ('1', '2'), ('3', '2'), ('122', '182'), ('352', '270'), ('0', '2'), ('143', '100'), ('0.2', '1.5'), ('4', '3')])
OrderedDict([('72', '59'), ('1', '0'), ('3', '0'), ('122', '138'), ('352', '524'), ('0', '2'), ('143', '94'), ('0.2', '0.1'), ('4', '1')])
OrderedDict([('72', '49'), ('1', '2'), ('3', '3'), ('122', '168'), ('352', '154'), ('0', '0'), ('143', '187'), ('0.2', '2.3'), ('4', '0')])
OrderedDict([('72', '75'), ('1', '0'), ('3', '1'), ('122', '155'), ('352', '199'), ('0', '0'), ('143', '129'), ('0.2', '2.7'), ('4', '0')])
OrderedDict([('72', '59'), ('1', '2'), ('3', '3'), ('122', '169'), ('352', '481'), ('0', '2'), ('143', '161'), ('0.2', '1.7'), ('4', '3')])
OrderedDict([('72', '59'), ('1', '3'), ('3', '3'), ('122', '143'), ('352', '154'), ('0', '2'), ('143', '115'), ('0.2', '1.3'), ('4', '4')])
OrderedDict([('72', '68'), ('1', '3'), ('3', '2'), ('122', '124'), ('352', '195'), ('0', '2'), ('143', '200'), ('0.2', '0.8'), ('4', '2')])
```

```
#import dataset
df = pd.read_csv("heart1.csv")
```

```
X = df.drop(columns=['target'], axis=1)
Y = df['target']
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

```
from sklearn.ensemble import RandomForestClassifier

max_accuracy = 0


for x in range(2000):
    rf = RandomForestClassifier(random_state=x)
    rf.fit(X_train,Y_train)
    Y_pred_rf = rf.predict(X_test)
    current_accuracy = round(accuracy_score(Y_pred_rf,Y_test)*100,2)
    if(current_accuracy>max_accuracy):
        max_accuracy = current_accuracy
        best_x = x

#print(max_accuracy)
#print(best_x)

rf = RandomForestClassifier(random_state=best_x)
rf.fit(X_train,Y_train)
```

```
RandomForestClassifier(random_state=238)
```

```
r = np.loadtxt(open("Book2.csv", "rb"), delimiter=",", skiprows=0, dtype=[('f0',int),('f1',int),('f2',int),('f3',int),('f4',int),('f5',int),('f6',int),('f7',int),('f8',int),
                                                                          ('f9',float),('f10',int),('f11',int),('f12',int)])
r
```

```
array([(72, 1, 3, 122, 352, 0, 0, 143, 0, 0.2, 0, 4, 1),
       (35, 0, 1, 140, 279, 1, 0,  79, 1, 3.6, 1, 0, 2),
       (44, 1, 0, 185, 345, 1, 0, 171, 0, 4.7, 2, 1, 1),
       (57, 0, 3, 163, 524, 0, 2, 164, 0, 2. , 2, 3, 2),
       (62, 1, 1, 128, 391, 0, 2,  81, 0, 1. , 1, 0, 2),
       (71, 0, 3, 124, 485, 0, 0, 158, 1, 0.8, 0, 0, 3),
       (56, 0, 2, 182, 270, 0, 2, 100, 1, 1.5, 2, 3, 2),
       (59, 1, 0, 138, 524, 1, 1,  94, 1, 0.1, 2, 1, 0),
       (49, 1, 3, 168, 154, 1, 0, 187, 0, 2.3, 0, 0, 2),
       (75, 1, 1, 155, 199, 0, 2, 129, 1, 2.7, 0, 0, 0),
       (59, 1, 3, 169, 481, 1, 2, 161, 1, 1.7, 2, 3, 2),
       (59, 1, 3, 143, 154, 0, 1, 115, 1, 1.3, 2, 4, 3),
       (68, 1, 2, 124, 195, 1, 1, 200, 1, 0.8, 2, 2, 3),
```

```
target=[]
for d in range(0,100):
  input_data =np.asarray(tuple(r[d]))




  # change the input data to a numpy array
  input_data_as_numpy_array= np.asarray(input_data)

  # reshape the numpy array as we are predicting for only on instance
  input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

  prediction = model.predict(input_data_reshaped)
  print(prediction)
  a=prediction[0]
  target.append(a)
  if (prediction[0]== 0):
    print('The Person does not have a Heart Disease')
  else:
    print('The Person has Heart Disease')
print(target)
len(target)
```

```
[1]
The Person has Heart Disease
[0]
The Person does not have a Heart Disease
[0]
The Person does not have a Heart Disease
```

```
[ ]  csv_input = pd.read_csv('Book2.csv')
     len(csv_input)

     99

[ ]  del target[0]

[ ]  csv_input = pd.read_csv('Book2.csv')
     csv_input['target'] = target
     csv_input.to_csv('output4.csv', index=False)
```

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 72 | 1 | 3 | 122 | 352 | 0 | 0 | 143 | 0 | 0.2 | 0 | 4 | 1 |
| 35 | 0 | 1 | 140 | 279 | 1 | 0 | 79 | 1 | 3.6 | 1 | 0 | 2 |
| 44 | 1 | 0 | 185 | 345 | 1 | 0 | 171 | 0 | 4.7 | 2 | 1 | 1 |
| 57 | 0 | 3 | 163 | 524 | 0 | 2 | 164 | 0 | 2 | 2 | 3 | 2 |
| 62 | 1 | 1 | 128 | 391 | 0 | 2 | 81 | 0 | 1 | 1 | 0 | 2 |
| 71 | 0 | 3 | 124 | 485 | 0 | 0 | 158 | 1 | 0.8 | 0 | 0 | 3 |
| 56 | 0 | 2 | 182 | 270 | 0 | 2 | 100 | 1 | 1.5 | 2 | 3 | 2 |
| 59 | 1 | 0 | 138 | 524 | 1 | 1 | 94 | 1 | 0.1 | 2 | 1 | 0 |
| 49 | 1 | 3 | 168 | 154 | 1 | 0 | 187 | 0 | 2.3 | 0 | 0 | 2 |
| 75 | 1 | 1 | 155 | 199 | 0 | 2 | 129 | 1 | 2.7 | 0 | 0 | 0 |
| 59 | 1 | 3 | 169 | 481 | 1 | 2 | 161 | 1 | 1.7 | 2 | 3 | 2 |
| 59 | 1 | 3 | 143 | 154 | 0 | 1 | 115 | 1 | 1.3 | 2 | 4 | 3 |
| 68 | 1 | 2 | 124 | 195 | 1 | 1 | 200 | 1 | 0.8 | 2 | 2 | 3 |
| 55 | 0 | 2 | 137 | 287 | 0 | 1 | 140 | 0 | 4 | 0 | 4 | 3 |
| 44 | 1 | 2 | 109 | 268 | 0 | 1 | 73 | 1 | 0 | 2 | 2 | 1 |
| 70 | 0 | 0 | 178 | 305 | 1 | 0 | 177 | 0 | 1.7 | 2 | 3 | 0 |
| 75 | 1 | 3 | 193 | 307 | 1 | 2 | 160 | 1 | 5.4 | 1 | 0 | 3 |
| 40 | 1 | 0 | 140 | 294 | 0 | 1 | 101 | 0 | 0.7 | 0 | 4 | 0 |
| 73 | 1 | 0 | 123 | 400 | 0 | 2 | 161 | 0 | 5.9 | 1 | 3 | 3 |
| 75 | 0 | 1 | 126 | 281 | 0 | 2 | 112 | 1 | 4.4 | 1 | 2 | 0 |
| 68 | 0 | 0 | 157 | 142 | 1 | 1 | 154 | 1 | 4.6 | 0 | 3 | 0 |
| 64 | 1 | 3 | 195 | 411 | 0 | 0 | 199 | 1 | 3.8 | 2 | 0 | 2 |
| 37 | 0 | 0 | 191 | 209 | 1 | 1 | 86 | 1 | 1.7 | 2 | 0 | 0 |
| 47 | 1 | 1 | 143 | 504 | 1 | 0 | 189 | 0 | 3.4 | 0 | 3 | 0 |
| 37 | 0 | 2 | 110 | 147 | 0 | 2 | 115 | 0 | 3.1 | 1 | 3 | 3 |
| 51 | 0 | 0 | 110 | 223 | 1 | 2 | 165 | 0 | 6 | 1 | 1 | 2 |
| 54 | 1 | 0 | 104 | 355 | 1 | 0 | 94 | 0 | 5.6 | 2 | 3 | 2 |
| 48 | 0 | 3 | 173 | 480 | 0 | 1 | 185 | 0 | 5.4 | 0 | 2 | 0 |
| 39 | 0 | 1 | 178 | 199 | 1 | 0 | 175 | 0 | 0.7 | 2 | 1 | 1 |
| 39 | 0 | 2 | 155 | 483 | 1 | 2 | 155 | 0 | 3 | 0 | 3 | 3 |

**Fig-5.1 NumPy array to csv conversion of features**

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 72 | 1 | 3 | 122 | 352 | 0 | 0.1 | 143 | 0 | 0.02.00 | 0.3 | 4 | 1.1 | 0 |
| | 35 | 0 | 1 | 140 | 279 | 1 | 0 | 79 | 1 | 3.6 | 1 | 0 | 2 | 0 |
| | 44 | 1 | 0 | 185 | 345 | 1 | 0 | 171 | 0 | 4.7 | 2 | 1 | 1 | 0 |
| | 57 | 0 | 3 | 163 | 524 | 0 | 2 | 164 | 0 | 2 | 2 | 3 | 2 | 1 |
| | 62 | 1 | 1 | 128 | 391 | 0 | 2 | 81 | 0 | 1 | 1 | 0 | 2 | 0 |
| | 71 | 0 | 3 | 124 | 485 | 0 | 0 | 158 | 1 | 0.8 | 0 | 0 | 3 | 1 |
| | 56 | 0 | 2 | 182 | 270 | 0 | 2 | 100 | 1 | 1.5 | 2 | 3 | 2 | 0 |
| | 59 | 1 | 0 | 138 | 524 | 1 | 1 | 94 | 1 | 0.1 | 2 | 1 | 0 | 0 |
| | 49 | 1 | 3 | 168 | 154 | 1 | 0 | 187 | 0 | 2.3 | 0 | 0 | 2 | 1 |
| ) | 75 | 1 | 1 | 155 | 199 | 0 | 2 | 129 | 1 | 2.7 | 0 | 0 | 0 | 1 |
| 1 | 59 | 1 | 3 | 169 | 481 | 1 | 2 | 161 | 1 | 1.7 | 2 | 3 | 2 | 0 |
| 2 | 59 | 1 | 3 | 143 | 154 | 0 | 1 | 115 | 1 | 1.3 | 2 | 4 | 3 | 0 |
| 3 | 68 | 1 | 2 | 124 | 195 | 1 | 1 | 200 | 1 | 0.8 | 2 | 2 | 3 | 1 |
| 4 | 55 | 0 | 2 | 137 | 287 | 0 | 1 | 140 | 0 | 4 | 0 | 4 | 3 | 0 |
| 5 | 44 | 1 | 2 | 109 | 268 | 0 | 1 | 73 | 1 | 0 | 2 | 2 | 1 | 0 |
| 6 | 70 | 0 | 0 | 178 | 305 | 1 | 0 | 177 | 0 | 1.7 | 2 | 3 | 0 | 1 |
| 7 | 75 | 1 | 3 | 193 | 307 | 1 | 2 | 160 | 1 | 5.4 | 1 | 0 | 3 | 0 |
| 8 | 40 | 1 | 0 | 140 | 294 | 0 | 1 | 101 | 0 | 0.7 | 0 | 4 | 0 | 0 |
| 9 | 73 | 1 | 0 | 123 | 400 | 0 | 2 | 161 | 0 | 5.9 | 1 | 3 | 3 | 0 |
| 0 | 75 | 0 | 1 | 126 | 281 | 0 | 2 | 112 | 1 | 4.4 | 1 | 2 | 0 | 0 |
| 1 | 68 | 0 | 0 | 157 | 142 | 1 | 1 | 154 | 1 | 4.6 | 0 | 3 | 0 | 0 |
| 2 | 64 | 1 | 3 | 195 | 411 | 0 | 0 | 199 | 1 | 3.8 | 2 | 0 | 2 | 1 |
| 3 | 37 | 0 | 0 | 191 | 209 | 1 | 1 | 86 | 1 | 1.7 | 2 | 0 | 0 | 0 |

**Fig-5.2 Updating predicted output to csv file**

# CHAPTER 6

# CONCLUSION

Importance of decreasing the death rate of people is discussed. Learnt how machine learning helps in predicting heart disease. Different machine learning algorithms were applied to create a best model with high accuracy. Random forest classifier gave high accuracy of 83%. Different relations between attributes using heatmap, pair plot, histogram, pie chart and many more are discussed. plotted the linear relation between each pair of attributes. At the end we compared accuracies of all the algorithms using matplotlib. Proposed work of generating hundreds of data using random function and updated to csv file and took this input from csv at a time and predicted output using random forest model and predicted output and updated this output to the same csv file.

# CHAPTER 7

# REFERENCES


1) L. Loku, B. Fetaji, A. Krstev, M. Fetaji, Z. Zdravev, Using python programming for assessing and solving health management issues, South East Eur. J. Sustain. Dev. 4 (1) (2020).

2) P. Mathur, Overview of Machine Learning in healthcare, in: Machine Learning Applications using Python, A. Press, Berkeley, CA, 2019, pp. 1-11.

3) Zahariev, M. Zveryakov, S. Prodanov, G. Zaharieva, P. Angelov, S. Zarkova,M. Petrova, Debt management evaluation through support vector machines: on the example of Italy and Greece, Entrepreneurship Sustain. Issues 7 (3) (2020).

4) C. Wang, C. Liang, Msipred: a python package for tumor microsatellite instability classification from tumor mutation annotation data using a support vector machine, Sci. Rep. 8 (1) (2018) 1–10.

5) M. Huljanah, Z. Rustam, S. Utama, T. Siswantining, Feature selection using random forest classifier for predicting prostate cancer, IOP Conf. Ser.: Mater. Sci. Eng. 546 (5) (2019) 1–9.

6) S. Mehrang, J. Pietilä, I. Korhonen, An activity recognition framework deploying the random forest classifier and a single optical heart rate monitoring and triaxial accelerometer wrist-band, Sensors 18 (2) (2018) 613.

7) J. Ge, X. Li, H. Jiang, H. Liu, T. Zhang, M. Wang, T. Zhao, Picasso: A sparselearning library for high dimensional data analysis in R and python, J. Mach. Learn. Res. 20 (44) (2019) 1–5.

8) M.A. Saif, A.N. Medvedev, M.A. Medvedev, T. Atanasova, Classification of online toxic comments using the logistic regression and neural networks models, in: AIPConference Proceedings, Vol. 2048 (1) 2018, pp. 1–6.