INT247: MACHINE LEARNING FOUNDATION

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

Submitted in partial fulfillment of the requirements for the award of degree of

BTech Computer Science Engineering

PROJECT REPORT

Heart Disease Prediction Webapp

SUBMITTED BY

Name of student : Aniket Jha
Registration Number : 11912130
Section &Roll number : RKM055A22

Submitted To:
Dr. Sagar Pande

Student Declaration

To whom so ever it may concern

I, Aniket Jha **, 11912130 ,** hereby declare that the project on "**Heart Disease Prediction Webapp**" is based on my own work and carried by my own
Fact and observation during the course of my study under the guidance of -
Dr. Sagar Pande sir

Name of the Student :- Aniket Jha

Registration Number :- 11912130

Roll Number :- RKM055A22

ACKNOWLEDGEMENT

Primarily I would like to thank you my college and my teacher **Dr. Sagar Pande sir** for guiding continuously throughout the project .Then I would like to express my special thanks for college who provided such an opportunity for students to gain hands on experience on machine learning by learning and building projects which helps to get career ready.

I would like to again thank my own college Lovely Professional University for offering such a opportunity which not only improve my programming skill but also taught me other new technology.

Then I would like to thank my parents and friends who have helped me with their valuable suggestions and guidance in completion of this project

Date: 27/09/2021 **Aniket Jha**

Reg no: 11912130

Table of Contents

ABSTRACT

As Heart acts a major role in corporeal body part. Heart diseases or Cardiovascular Diseases (CVDs) are some of the main reason for a huge number of death in the world and some of the reports said that over the last few decades heart disease or CVD has emerged as the most life-threatening disease, not only in India but in the whole world.
Therefore, the diseases of heart wants more perfection and exactness for diagnose and analyse. This disease occurs due to various problems such as over pressure, blood sugar,
Chest pain,high blood pressure, Cholesterol etc. in human body . Heart is the next major organ comparing to the brain which has more priority in the Human body. It pumps the blood and supplies it to all organs of the whole body.

In the health care sector, Machine Learning plays an important role in the health care Industry.So, there is a need fora reliable, accurate, and feasible system to diagnose such diseases in time for proper treatment. Machine Learning algorithms and techniques have been applied to various medical datasets to automate the analysis of large and complex data. A system model is capable of several data processing algorithms for the classification of heart disease. Prediction of occurrences of heart diseases in the medical field is significant work. Data analytics is useful for prediction from more information and it helps the medical center to predict various diseases. The correct prediction of heart disease can prevent life threats, and incorrect prediction can prove to be fatal at the same time.

The objective of this project is to detect whether patients have any chance of heart disease or not by giving number of features to patients with having maximum accuracy of above 97%. By Using Machine learning algorithms and deep learning are applied to compare the results and analysis of the UCI Machine Learning Heart Disease dataset. The dataset consists of 14 main attributes used for performing the analysis

t.

INTRODUCTION

Heart is one of the most indispensable organs in the human body. It is a organ that serves as a pump to circulate the blood. The heart is a muscular organ about the size of a fist, located just behind and slightly left of the breastbone. The heart pumps blood through the network of arteries and veins called the cardiovascular system . Oxygen is distributed through the circulatory system of the body in the blood, and if the heart does not function correctly, the entire circulatory system of the body will fail. So if the heart doesn't work properly, it could even lead to death.

According to the World Health Organization (WHO), in the last 15 years, an estimated 17 million people die each year from cardiovascular disease, particularly heart attacks and strokes [1]. Heart disease and stroke are the biggest killers. To predict heart disease, Machine Learning can be used for identifying unseen patterns and providing some clinical insights that will assist the physicians in planning and providing care.

According to World Health Organisation, heart related diseases are responsible for taking 17.7 million lives every year, 31% of all global deaths. In India too, heart-related diseases have become the leading cause of mortality . Heart diseases have killed 1.7 million Indians in 2016, according to the 2016 Global Burden of Disease Report, released on September 15,2017. Heart-related diseases increase the spending on health care and also reduce the productivity of an individual. Estimates made by the World Health Organisation (WHO), suggest that India has lost up to $237 billion, from 2005- 2015, due to heart-related or Cardiovascular diseases. Thus, feasible and accurate prediction of heart-related diseases is very important.

Heart disease is common among both men and women in most countries around the world. Therefore, people should consider heart disease risk factors. Although it plays a genetic role, some lifestyle factors significantly affect heart disease . The known risk factors for heart disease; radiation therapy for age, gender, family history, smoking, some chemotherapy drugs and cancer, malnutrition, high blood pressure, high blood cholesterol levels, diabetes, obesity, physical mobility, stress, and poor hygiene. These are the various risk factors in which the patient's exposure towards developing a CVD.

The most common type is coronary artery disease, which can cause a heart attack. Other types of heart disease may involve the valves in the heart, or the heart may not pump well and cause heart failure. Some people are born with heart disease. Anyone, including children, can develop heart disease. It happens when a substance called plaque builds up in your arteries. Smoking, unhealthy eating and lack of exercise increase your risk of heart disease. High cholesterol, high blood pressure or diabetes can also increase your risk of heart disease.

> In contrast, the removal or improvement of this factor decreases this risk. This interpretation suggests the causality between the factor and the illness, which means that the risk factor precedes the disease (the notion of anteriority). Correction of the factor will cause the disease (the idea of reversibility) to decrease its occurrence. Of course, it must be recognized in several different populations and offer a plausible physiopathological explanation of the disease. Strictly speaking, when there is no direct causal relationship, it is a “risk marker,” a witness to a process (e.g., the elevation of microalbuminuria, elevation of C-reactive protein CRP)
> .

the main heart disease risk factors such as physiological factors (age, sex, and menopausal status), lifestyle factors (smoking, physical activity, alcohol, stress), metabolic syndrome factors (insulin resistance), dyslipidemia, abdominal obesity, high blood pressure) and dietary factors. A heart disease risk factor is defined as a factor in which the patient’s exposure to this factor increases the risk of developing a CVD. In contrast, the removal or improvement of this factor decreases this risk. The risk factor’s importance is defined by the association’s strength with the disease (expressed by the relative risk observed in the exposed subjects compared to the unexposed) and the gradual association (parallel to the risk factor).

There are several types of heart disease which include :

- Coronary Artery Disease (CAD)
- Heart Arrhythmias.
- Heart Failure.
- Heart Valve Disease.
- Pericardial Disease.
- Cardiomyopathy (Heart Muscle Disease)
- Congenital Heart Disease.

To examine the cardiac disease mischance, the particular issues which need to be discussed are those related to the behaviors. Furthermore, patients will undergo extensive examinations, such as blood pressure, glucose, vital signs, chest pain, electrocardiograms, maximum heart rate, and elevated levels of sugar, but the bright side may be that successful treatment is feasible if the disease is easily and early detected and anticipated, but treatment for all of these cardiac patients is depending on clinical studies, the patient history, and the responses to questions by the patient

To deal with this disease, there are several methods of prevention, such us natural methods, like stoping smoking, maintaining a healthy weight, adopting a healthy diet and practicing sports regularly.We also have the scientific methods such as drugs and surgeries. The prediction of this disease before being infected is part of the prevention methods, or the computer tools are the most used means in it, more precisely the Machine Learning algorithms . Determining the probability of having cardiac disease manually is hard to depend on as risk factors. Recently, to solve difficult issues, a range of data mining techniques and machine learning techniques are built [6, 7]. Still, more advanced machine learning will assist us to identify patterns and their useful knowledge. While it has several uses in the medical field, machine learning is mainly utilized to forecast the heart disease.

Machine learning (ML) plays a significant role in disease predicting [9]. It predicts whether the patient has a particular disease type or not based on an efficient learning technique. In making of these project I have used several supervised learning techniques for predicting the early stage of heart disease by providing them risk factor Whether they have any chance of getting heart disease or not. Some of the techniques which I have used in these project are :

- K-nearest neighbor (KNN)
- Random forest (RF)
- Ada Boosting With Random Forest
- Gradient Boosting
- XG BOOST

With using all these techniques the highest accuracy of 97.82% I got in K-nearest neighbor (KNN).

LITERATURE REVIEW

There are various work done by various researcher or scientist on heart diease using the UCI dataset which tend to predict heart disease . Using different data mining methods, various levels of accuracy have been achieved. Typically, heart is unable to push the necessary amount of blood to ot her areas of the body in order to satisfy the normal functioning of the body in this disease, and because of this, heart failure eventually occurs. The prevalence of heart disease is very high in the United States. Symptoms of heart disease include shortness of breath, physical body fatigue, swollen feet, and tiredness with associated signs, such as increased jugular venous pressure and peripheral edoema due to functional or non-functional cardiac irregularities. The early-stage investigation approaches used to detect heart dis- ease have been difficult, and the resulting difficulty is one of the key factors affecting the standard of living. Diagnosis and treatment of heart disease is very difficult, especially in developing countries, owing to the rare availability of diagnostic instruments and the shortage of doctors and other services affecting the proper prediction of heart disease. The precise and correct detection of heart disease is important to reduce the associated risk of serious heart complications and to improve heart safety. Approximately 3 percent of the health care financial budget is impacted by the costs of heart disease management.

There are various machine learning techinques that I have used in these project which are K-nearest neighbor (KNN) , Random forest (RF), Ada Boosting With Random Forest , Gradient Boosting, XG BOOST. After the result shows that the K-nearest neighbor (KNN) has achived the highest accuracy of 97.82 % .To Which RF achived 86 % accuracy , Ada Boosting With Random Forest achieved accuracy of 91 %, Gradient Boosting achieved accuracy of 89 % , XG BOOST achieved accuracy of 91 %. Heart attack must be diagnosed in a timely and effective way due to its high prevalence.Therefore proceeding with maximum accuracy can lead to provide actual status of the heart to patient whether they are having any chances of heart disease or not.

SOFTWARE TOOLS

1. JUPYTER NOTEBOOK:-

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages".[2] It was spun off from IPython in 2014 by Fernando Pérez[3] and Brian Granger.[4] Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, and also a homage to Galileo's notebooks recording the discovery of the moons of Jupiter. Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab. Jupyter is financially sponsored by NumFOCUS

2. VS CODE:

Visual Studio Code is a source-code editor that can be used with a variety of programming languages,
including Java, JavaScript, Go, Node.js, Python and C++. It is based on the Electron framework,[19] which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

3. GITHUB:-

GitHub, Inc. is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration and wikis for every project. Headquartered in California, it has been a subsidiary of Microsoft since 2018.

It is commonly used to host open-source projects. As of November 2021, GitHub reports having over 73 million developers and more than 200 million repositories (including at least 28 million public repositories). It is the largest source code host as of November 2021.

4. HEROKU :-

Heroku is a cloud platform as a service (PaaS) supporting several programming languages. One of the first cloud platforms, Heroku has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it has features for a developer to build, run and scale applications in a similar manner across most languages. Heroku was acquired by Salesforce in 2010 for $212 million.

5. SCIKIT-LEARN :-

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis

MODULES / LIBRARIES USED

1. FLASK:-

Flask is a **micro web framework written in Python**. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.[3]

Applications that use the Flask framework include Pinterest and LinkedIn.

2. NumPy :-

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.NumPy stands for Numerical Python. NumPy targets the CPython reference implementation of Python, which is a non-
optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy.

3. OS :-

The main purpose of the OS module is to interact with your operating system. The primary use I find for it is to create folders, remove folders, move folders, and sometimes change the working directory. You can also access the names of files within a file path by doing listdir(). We do not cover that in this video, but that's an option.The os module is a part of the standard library, or stdlib, within Python 3

4. MATPLOTLIB :-

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. Matplotlib is a low level graph plotting library in python that serves as a visualization utility.Matplotlib was created by John D. Hunter.Matplotlib is open source and we can use it freely.Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility

5. PICKLE :-

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script

Advantages of using Pickle Module:

1. **Recursive objects (objects containing references to themselves):**
2. **Object sharing (references to the same object in different places):**
3. **User-defined classes and their instances**

It is advisable not to unpickle data received from an untrusted source as they may pose security threat. However, the pickle module has no way of knowing or raise alarm while pickling malicious data

4. JOBLIB :-

Joblib is a set of tools to provide lightweight pipelining in Python. In particular:

1. transparent disk-caching of functions and lazy re-evaluation (memoize pattern)
2. easy simple parallel computing

Joblib is optimized to be fast and robust on large data in particular and has specific optimizations for *numpy* arrays

METHODOLOGY

1. DATASET DESCRIPTION :-

The dataset used in these project is Public Health Dataset. This data set dates from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. It contains 76 attributes, including the predicted attribute, but all published experiments refer to using a subset of 14 of them. The "target" field refers to the presence of heart disease in the patient. It is integer valued 0 = no disease and 1 = disease. of heart disease in the patient. It is integer-valued 0 = no disease and 1 = disease. The first four rows and all the dataset features are shown in Table 1 without any preprocessing. Now the attributes which are used in this research purpose are described as follows and for what they are used or resemble:

- **Age—**age of patient in years, sex—(1 = male; 0 = female).
- **Cp—**chest pain type.
- **Trestbps—**resting blood pressure (in mm Hg on admission to the hospital). The normal range is 120/80 (if you have a normal blood pressure reading, it is fine, but if it is a little higher than it should be, you should try to lower it. Make healthy changes to your lifestyle)**.**
- **Chol—**serum cholesterol shows the amount of triglycerides present. Triglycerides are another lipid that can be measured in the blood. It should be less than 170 mg/dL (may differ in different Labs).
- **FBS—**Fasting blood sugar larger than 120 mg/dl (1 true). Less than 100 mg/dL (5.6 mmol/L) is normal, and 100 to 125 mg/dL (5.6 to 6.9 mmol/L) is considered prediabetes.
- **Restecg—**resting electrocardiographic results.
- **Thalach—**maximum heart rate achieved. The maximum heart rate is 220 minus your age.
- **Exang—**exercise-induced angina (1 yes). Angina is a type of chest pain caused by reduced blood flow to the heart. Angina is a symptom of coronary artery disease.
- **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 fluoroscopy.
- **Thal**—no explanation provided, but probably thalassemia (3 normal; 6 fixed defects; 7 reversible defects).
- **Target (T)**—no disease = 0 and disease = 1, (angiographic disease status).

1.A <u>DATASET FEATURES :-</u>

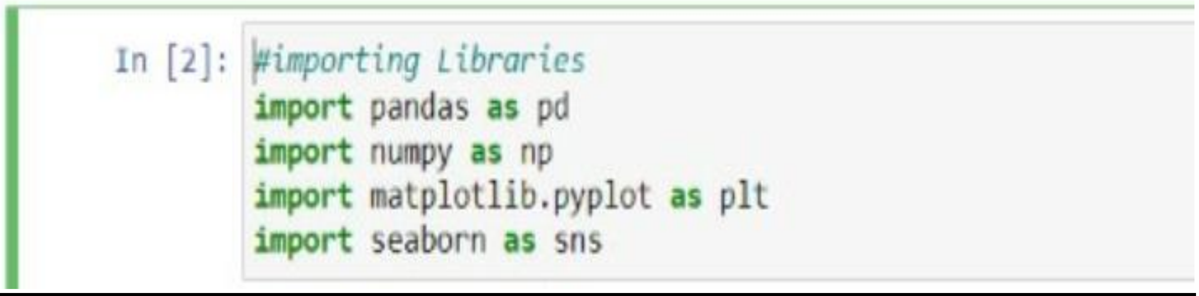
1. **age** - age in year
2. **sex -** (1 = male; 0 = female)
3. **cp** - chest pain type
   - 0: Typical angina: chest pain related decrease blood supply to the heart
   - 1: Atypical angina: chest pain not related to heart
   - 2: Non-anginal pain: typically esophageal spasms (non heart related)
   - 3: Asymptomatic: chest pain not showing signs of disease
4. trestbps - resting blood pressure (in mm Hg on admission to the hospital) anything above 130-140 is typically cause for concern
5. chol - serum cholestoral in mg/dl
   - serum = LDL + HDL + .2 * triglycerides
   - above 200 is cause for concern
6. fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
   - '>126' mg/dL signals diabetes
7. restecg - resting electrocardiographic results
   - 0: Nothing to note
   - 1: ST-T Wave abnormality
     - can range from mild symptoms to severe problems
     - signals non-normal heart beat
   - 2: Possible or definite left ventricular hypertrophy
     - Enlarged heart's main pumping chamber
8. thalach - maximum heart rate achieved
9. exang - exercise induced angina (1 = yes; 0 = no)
10. oldpeak - ST depression induced by exercise relative to rest looks at stress of heart during excercise unhealthy heart will stress more
11. slope - the slope of the peak exercise ST segment
    - 0: Upsloping: better heart rate with excercise (uncommon)

- 1: Flatsloping: minimal change (typical healthy heart)
- 2: Downslopins: signs of unhealthy heart

12. ca - number of major vessels (0-3) colored by flourosopy
    - colored vessel means the doctor can see the blood passing through
    - the more blood movement the better (no clots)

13. thal - thalium stress result
    - 1,3: normal
    - 6: fixed defect: used to be defect but ok now
    - 7: reversable defect: no proper blood movement when excercising
14. target - have disease or not (1=yes, 0=no) (= the predicted attribute)

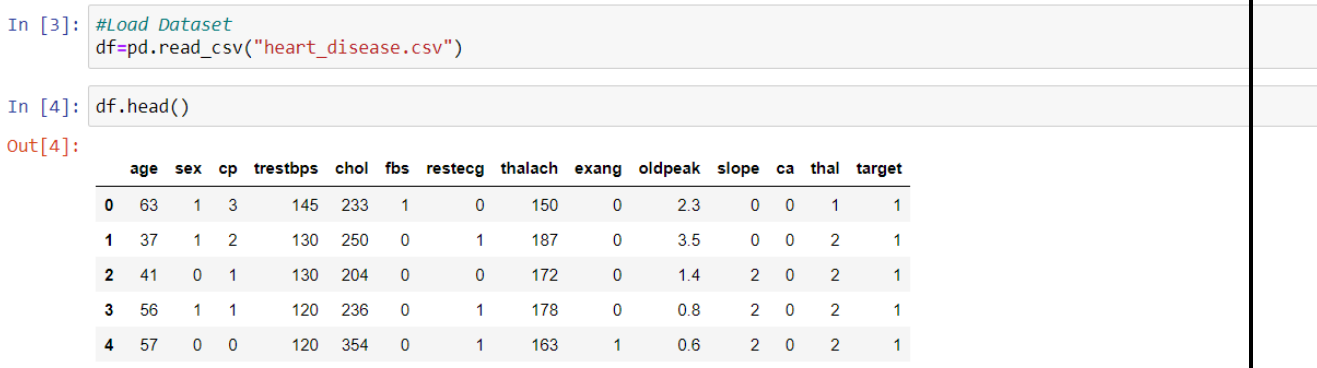
3. IMPORTING NECESSARY MODULES :-



```
In [2]: #importing Libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

4 .LOADING MODULES:-

I am loading data using pandas read_csv function which helps to read the data from the dataset.Here I have to pass location of the dataset and after that I have imported it.



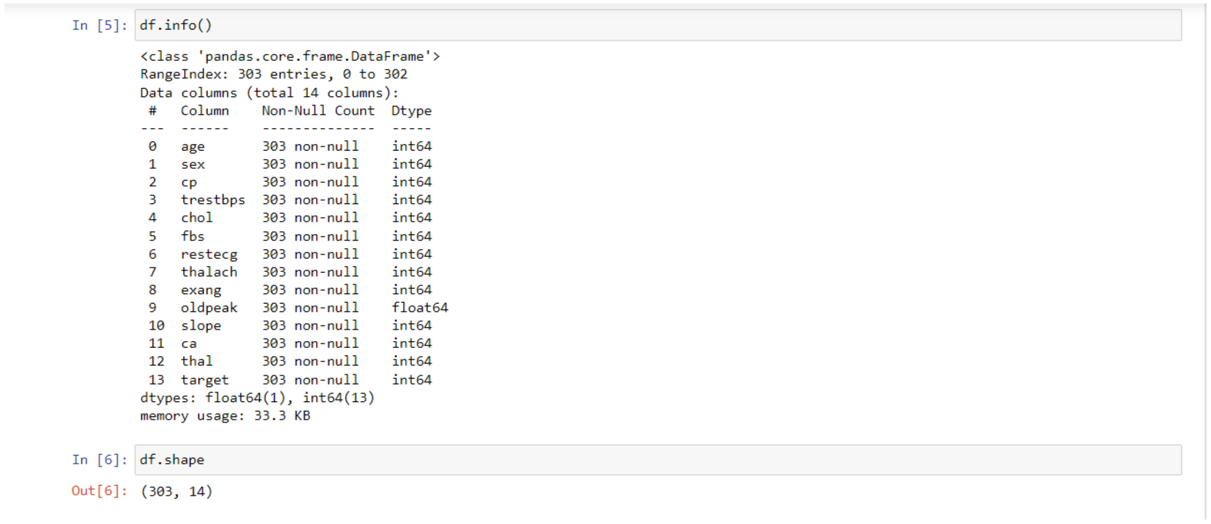
```
In [3]: #Load Dataset
        df=pd.read_csv("heart_disease.csv")

In [4]: df.head()

Out[4]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

5 .Exploratory Data Analysis (EDA):-

->To see the size of dataset I have used shape ( ) function which will print the number of rows in the dataset.

-> As these dataset contain 303 rows.

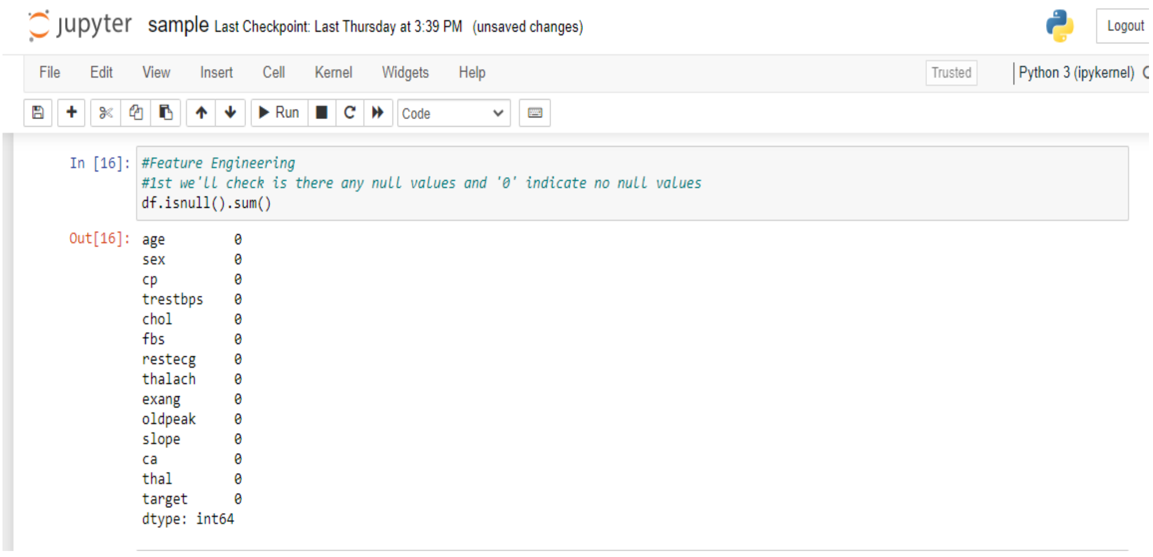


```
In [5]: df.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 303 entries, 0 to 302
        Data columns (total 14 columns):
         #   Column    Non-Null Count  Dtype
        ---  ------    --------------  -----
         0   age       303 non-null    int64
         1   sex       303 non-null    int64
         2   cp        303 non-null    int64
         3   trestbps  303 non-null    int64
         4   chol      303 non-null    int64
         5   fbs       303 non-null    int64
         6   restecg   303 non-null    int64
         7   thalach   303 non-null    int64
         8   exang     303 non-null    int64
         9   oldpeak   303 non-null    float64
         10  slope     303 non-null    int64
         11  ca        303 non-null    int64
         12  thal      303 non-null    int64
         13  target    303 non-null    int64
        dtypes: float64(1), int64(13)
        memory usage: 33.3 KB

In [6]: df.shape

Out[6]: (303, 14)
```

-> After getting size of dataset I have used info ( ) function which will print type of Dataset .As in these dataset there are 13 int types and 1 float data types.

5.1 .FEATURE ENGINEERING:-

- Here I have check whether dataset contain any null value or not.
  Luckily This dataset dosen’t contain any null values.

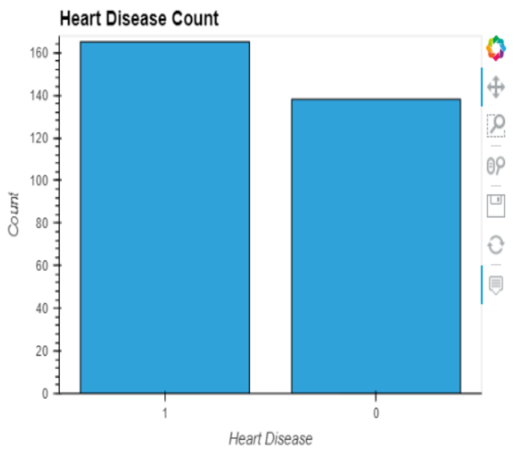


jupyter sample Last Checkpoint: Last Thursday at 3:39 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [16]: #Feature Engineering
         #1st we'll check is there any null values and '0' indicate no null values
         df.isnull().sum()

Out[16]: age         0
         sex         0
         cp          0
         trestbps    0
         chol        0
         fbs         0
         restecg     0
         thalach     0
         exang       0
         oldpeak     0
         slope       0
         ca          0
         thal        0
         target      0
         dtype: int64
```

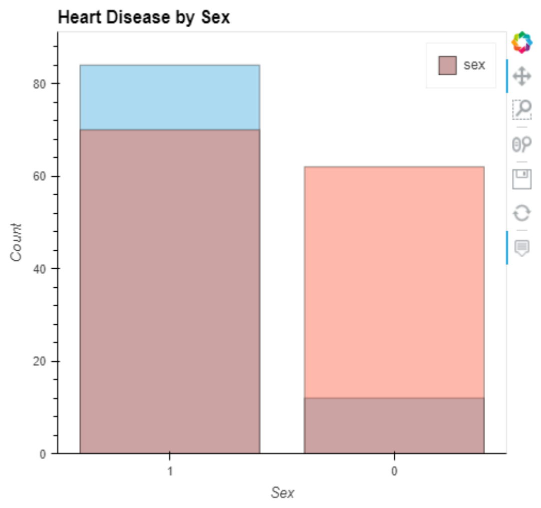
- After checking null values I am checking number of person with heart disease and Without heart disease.

```
In [23]: df.target.value_counts().hvplot.bar(
             title="Heart Disease Count", xlabel='Heart Disease', ylabel='Count',
             width=500, height=350
         )

Out[23]:
```



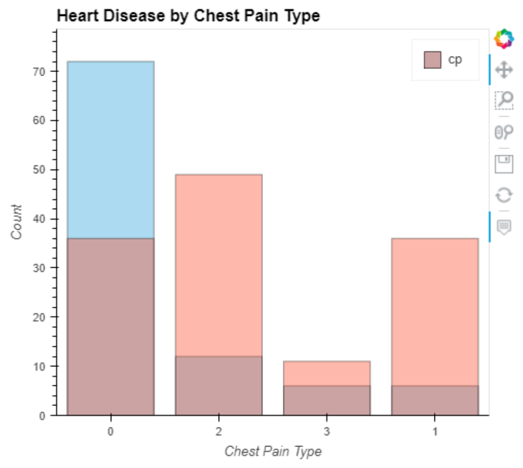
- As seeing from figure we have 165 person with heart disease and 138 person without heart disease.
- Now , I am evaluating heart disease according to sex.

In [50]:
```python
with_disease = df.loc[df['target']==1, 'sex'].value_counts().hvplot.bar(alpha=0.4)
without_disease = df.loc[df['target']==0, 'sex'].value_counts().hvplot.bar(alpha=0.4)

(without_disease * with_disease).opts(
    title="Heart Disease by Sex", xlabel='Sex', ylabel='Count',
    width=500, height=450, legend_cols=2, legend_position='top_right'
)
```
Out[50]:



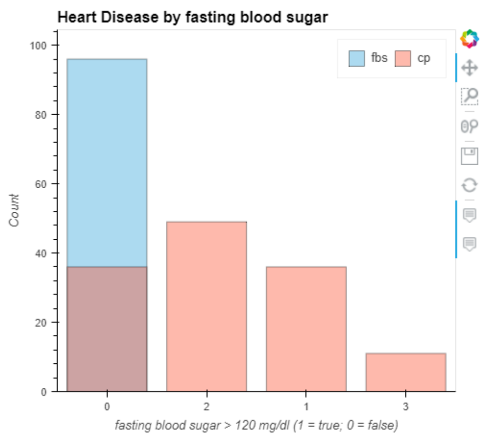
- Evaluating according to Chest pain type:

In [51]:
```python
with_disease = df.loc[df['target']==1, 'cp'].value_counts().hvplot.bar(alpha=0.4)
without_disease = df.loc[df['target']==0, 'cp'].value_counts().hvplot.bar(alpha=0.4)

(without_disease * with_disease).opts(
    title="Heart Disease by Chest Pain Type", xlabel='Chest Pain Type', ylabel='Count',
    width=500, height=450, legend_cols=2, legend_position='top_right'
)
```
Out[51]:



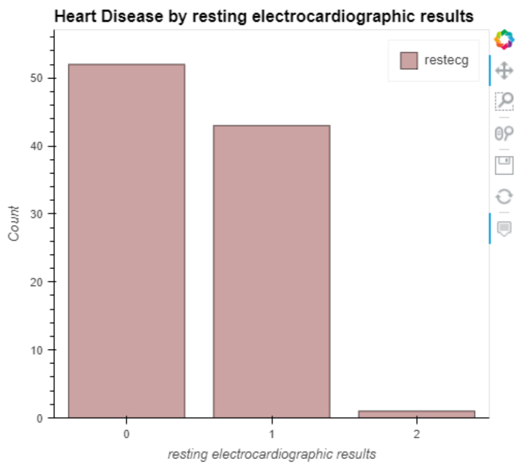
- Evaluating heart disease by fasting blood sugar:

```
In [52]: hwith_disease = df.loc[df['target']==1, 'fbs'].value_counts().hvplot.bar(alpha=0.4)
         without_disease = df.loc[df['target']==0, 'fbs'].value_counts().hvplot.bar(alpha=0.4)

         (without_disease * with_disease).opts(
             title="Heart Disease by fasting blood sugar", xlabel='fasting blood sugar > 120 mg/dl (1 = true; 0 = false)',
             ylabel='Count', width=500, height=450, legend_cols=2, legend_position='top_right'
         )
Out[52]:
```

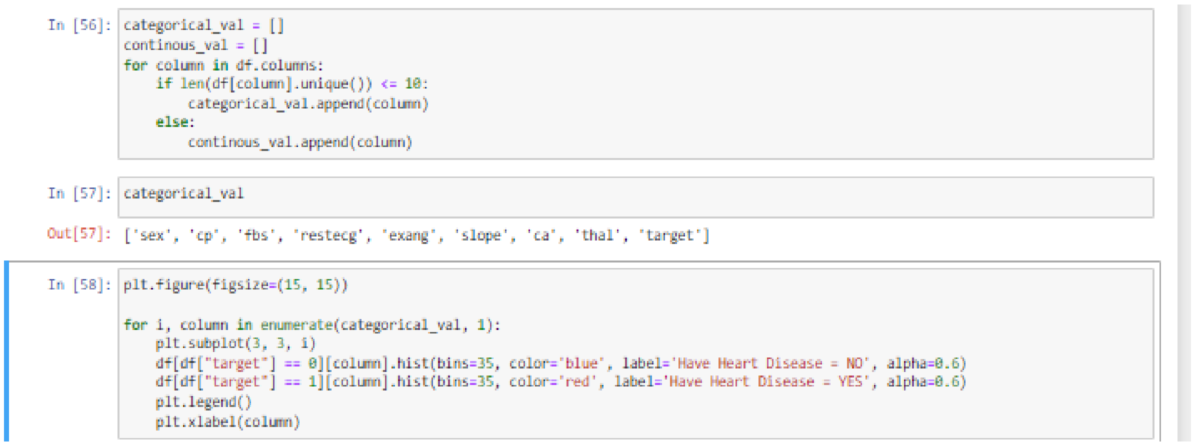


- Evaluating Heart Disease by electrocardiographic results:

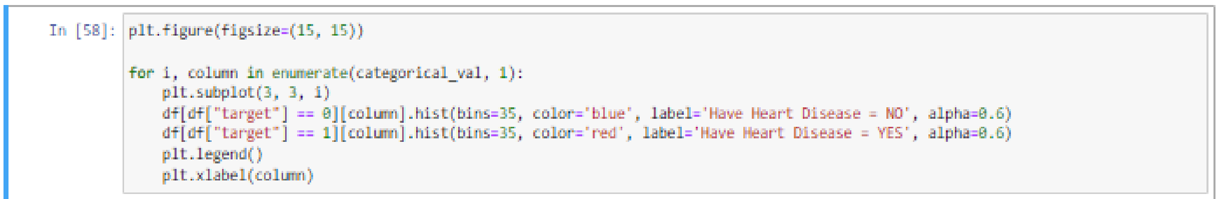
```
In [53]: with_disease = df.loc[df['target']==1, 'restecg'].value_counts().hvplot.bar(alpha=0.4)
         without_disease = df.loc[df['target']==0, 'restecg'].value_counts().hvplot.bar(alpha=0.4)

         (without_disease * without_disease).opts(
             title="Heart Disease by resting electrocardiographic results", xlabel='resting electrocardiographic results',
             ylabel='Count', width=500, height=450, legend_cols=2, legend_position='top_right'
         )
Out[53]:
```



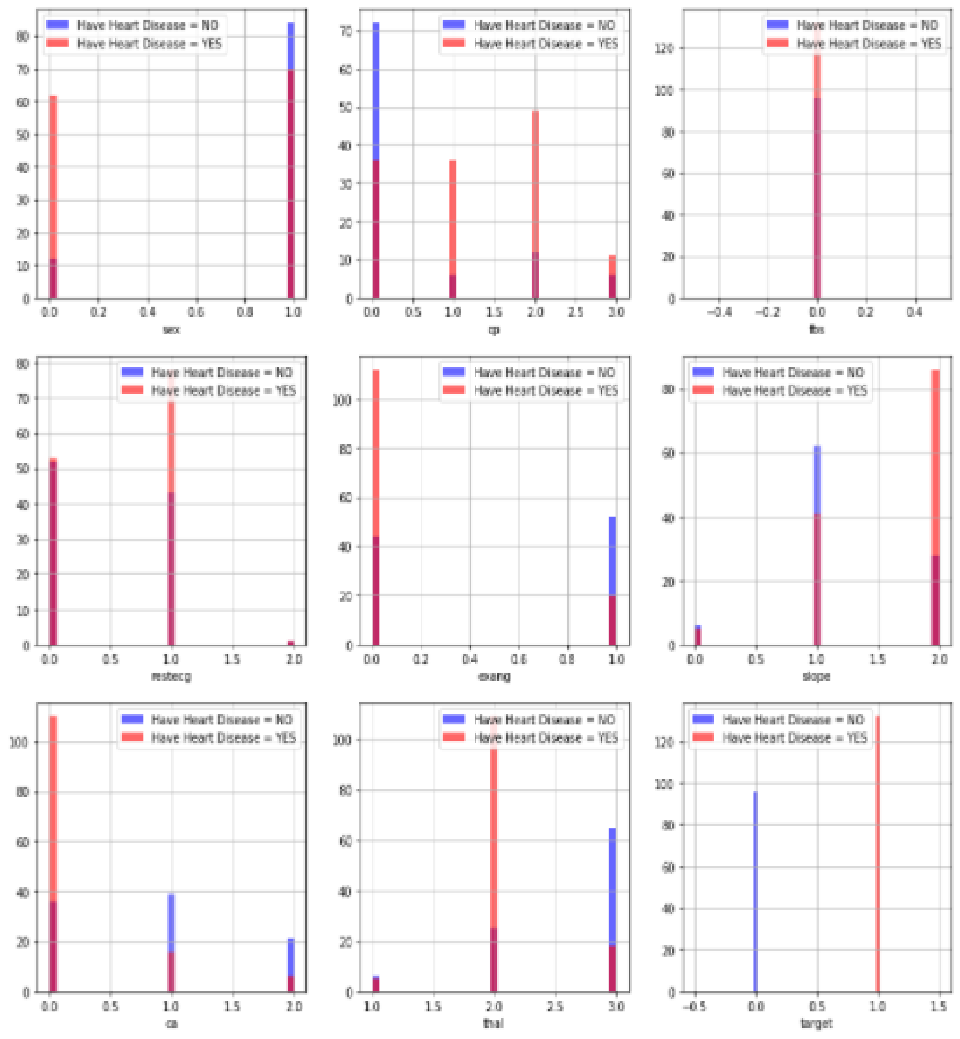
- Exploratory data analysis result:-



```
In [56]: categorical_val = []
         continous_val = []
         for column in df.columns:
             if len(df[column].unique()) <= 10:
                 categorical_val.append(column)
             else:
                 continous_val.append(column)

In [57]: categorical_val

Out[57]: ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']

In [58]: plt.figure(figsize=(15, 15))

         for i, column in enumerate(categorical_val, 1):
             plt.subplot(3, 3, i)
             df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
             df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
             plt.legend()
             plt.xlabel(column)
```

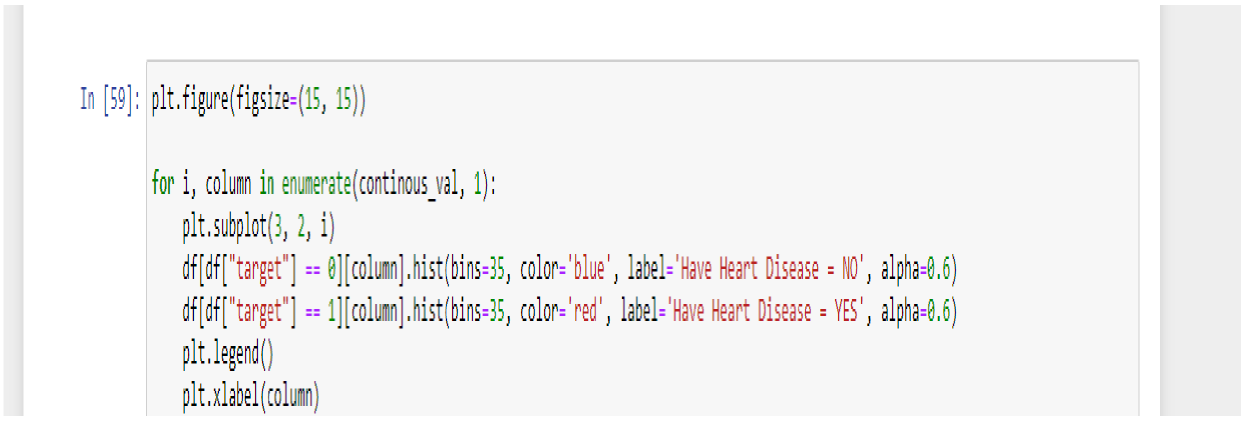


```
In [58]: plt.figure(figsize=(15, 15))

         for i, column in enumerate(categorical_val, 1):
             plt.subplot(3, 3, i)
             df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
             df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
             plt.legend()
             plt.xlabel(column)
```

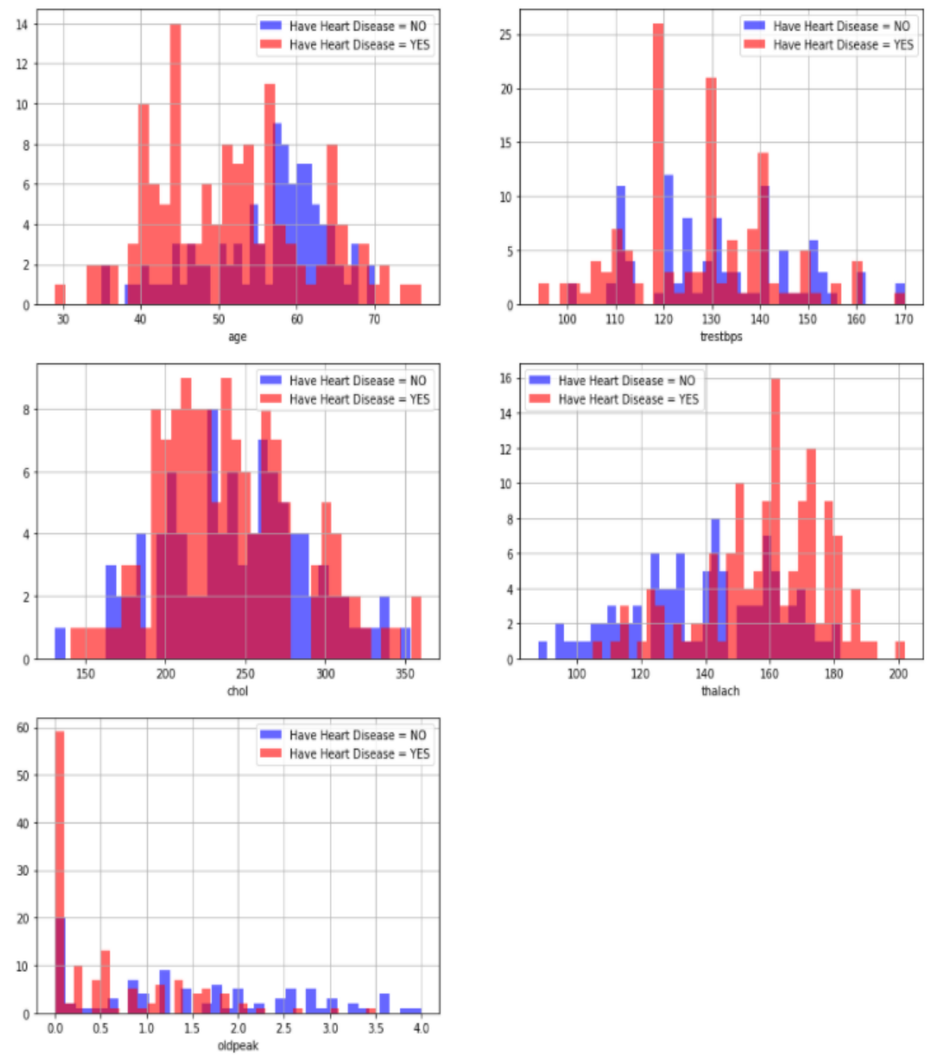


Result after Exploratory Data Analysis:-

- **Chest Pain (CP)**: People with cp equl to 1, 2, 3 are more likely to have heart disease than people with cp equal to 0.
- **Resting Electrocardiographic Results (resecg)** : People with value 1 (signals non-normal heart beat, can range from mild symptoms to severe problems) are more likely to have heart disease.
- **Exercise Induced Angina (exang)** : People with value 0 (No ==> exercice induced angina) have heart disease more than people with value 1 (Yes ==> exercice induced angina).
- **The Slope Of The Peak Exercise ST Segment (slope)** : People with slope value equal to 2 (Downslopins: signs of unhealthy heart) are more likely to have heart disease than people with slope value equal to 0 (Upsloping: better heart rate with excercise) or 1 (Flatsloping: minimal change (typical healthy heart)).
- **number of major vessels (0-3) colored by flourosopy (Ca)** : the more blood movement the better so people with ca equal to 0 are more likely to have heart disease.
- **Thalium Stress Result** (thal) : People with thal value equal to 2 (fixed defect: used to be defect but ok now) are more likely to have heart disease.



```
In [59]: plt.figure(figsize=(15, 15))

         for i, column in enumerate(continous_val, 1):
             plt.subplot(3, 2, i)
             df[df["target"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
             df[df["target"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
             plt.legend()
             plt.xlabel(column)
```



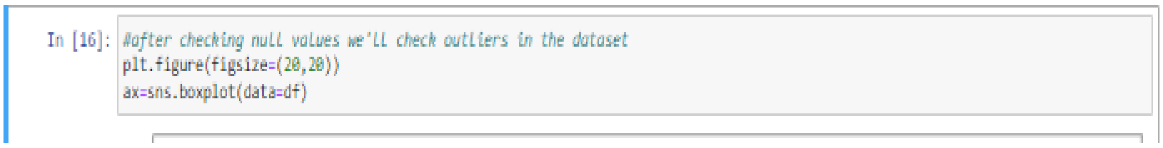
- **Trestbps** : resting blood pressure (in mm Hg on admission to the hospital) anything above 130-140 is typically cause for concern
- **serum cholestoral in mg/dl (chol)** : above 200 is cause for concern.

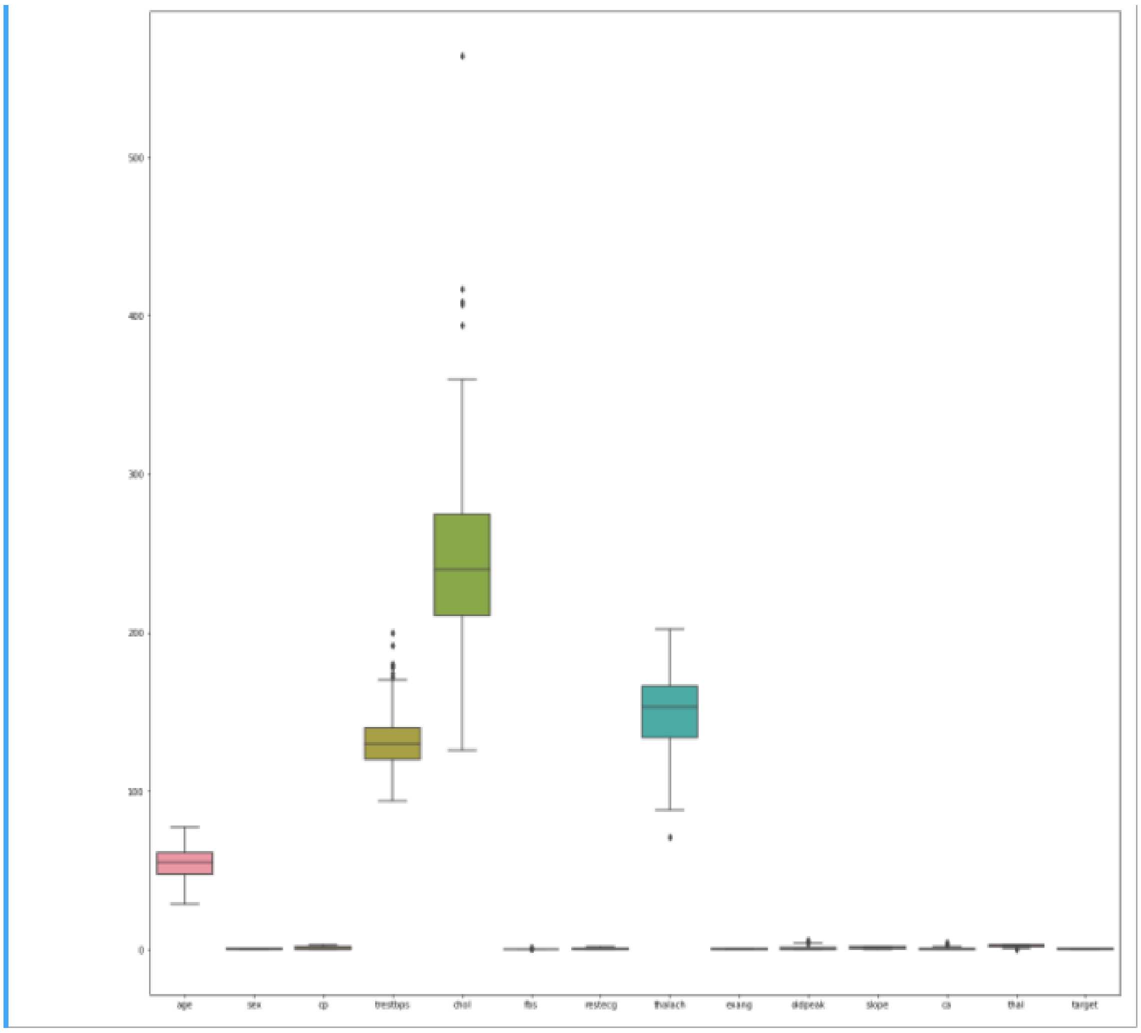
Maximum heart rate achieved (thalach) : People how acheived a maximum more than 140 are more likely to have heart disease.

- **OldPeak** ST depression induced by exercise relative to rest looks at stress of heart during excercise unhealthy heart will stress more

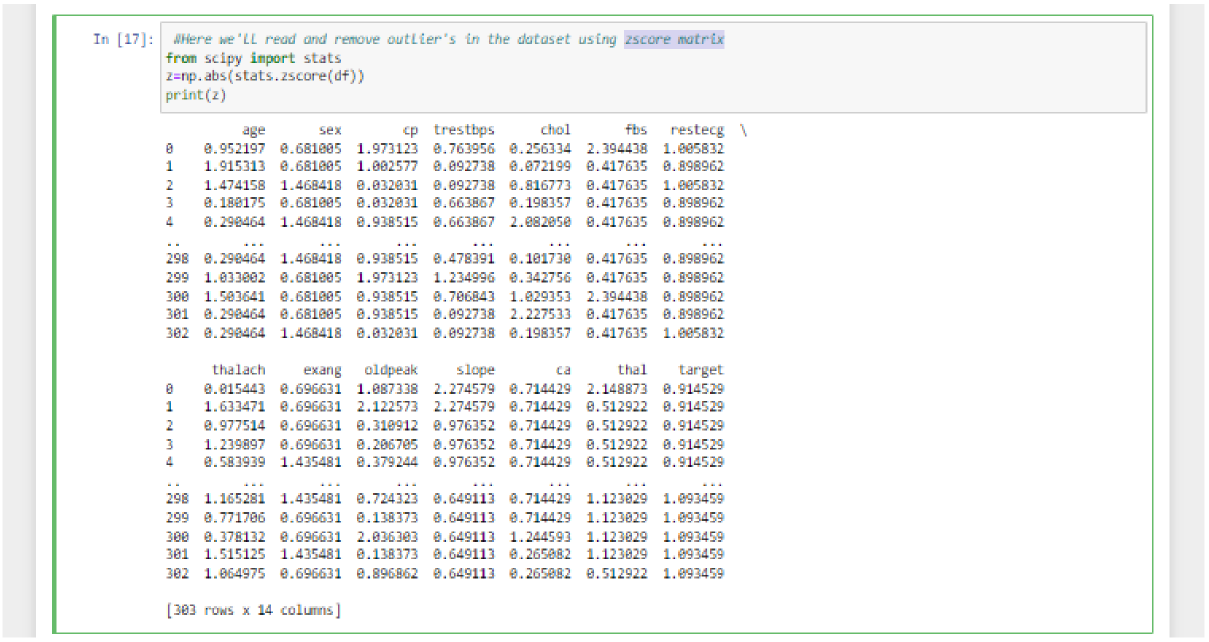
- After EDA I’m checking does this dataset contain any outlier’s

- Outliers are data points that don’t fit the pattern of rest of the numbers. They are the extremely high or extremely low values in the data set.
- A simple way to find an outlier is to examine the numbers in the data set. We will see that most numbers are clustered around a range and some numbers are way too low or too high compared to rest of the numbers. Such numbers are known as outliers.
- A data point that is distinctly separate from the rest of the data. One definition of outlier is any data point more than 1.5 interquartile ranges IQRs .IQRs below the first quartile or above the third quartile. The interquartile range IQRIQR is the difference between the third quartile and the first quartile of the data set.

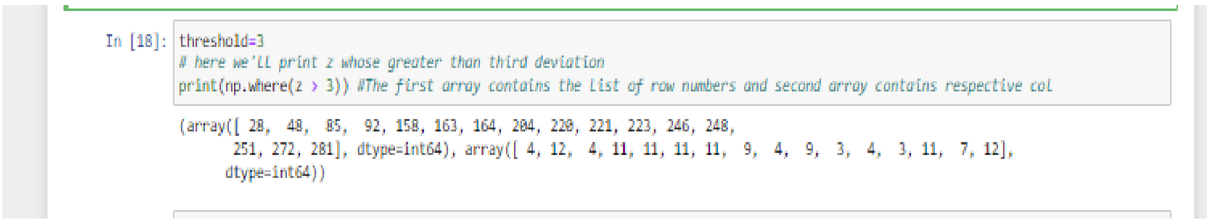

```
In [16]: #after checking null values we'll check outliers in the dataset
         plt.figure(figsize=(20,20))
         ax=sns.boxplot(data=df)
```



- To remove and read outlier’s we can use zscore matrix.
- After zscore matrix we will print z which is greater than it’s 3rd standard deviation.
- The first array contains the list of row numbers and second array contains respective column

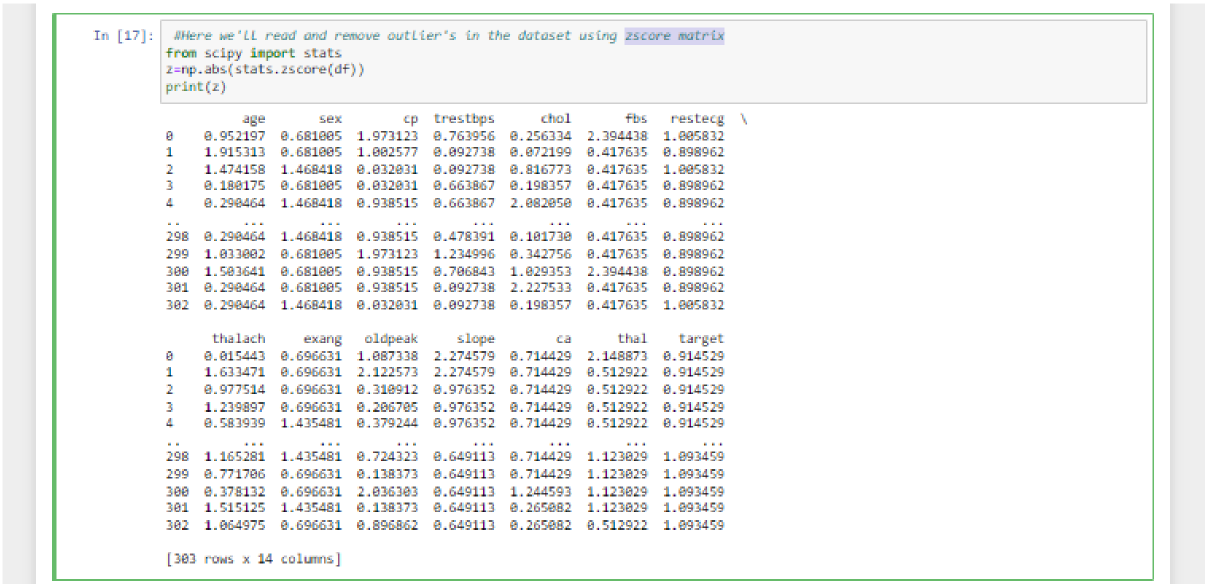


```
In [17]: #Here we'll read and remove outlier's in the dataset using zscore matrix
         from scipy import stats
         z=np.abs(stats.zscore(df))
         print(z)

              age       sex        cp  trestbps      chol       fbs   restecg  \
0    0.952197  0.681005  1.973123  0.763956  0.256334  2.394438  1.005832
1    1.915313  0.681005  1.002577  0.092738  0.072199  0.417635  0.898962
2    1.474158  1.468418  0.032031  0.092738  0.816773  0.417635  1.005832
3    0.180175  0.681005  0.032031  0.663867  0.198357  0.417635  0.898962
4    0.290464  1.468418  0.938515  0.663867  2.082050  0.417635  0.898962
..        ...       ...       ...       ...       ...       ...       ...
298  0.290464  1.468418  0.938515  0.478391  0.101730  0.417635  0.898962
299  1.033002  0.681005  1.973123  1.234996  0.342756  0.417635  0.898962
300  1.503641  0.681005  0.938515  0.706843  1.029353  2.394438  0.898962
301  0.290464  0.681005  0.938515  0.092738  2.227533  0.417635  0.898962
302  0.290464  1.468418  0.032031  0.092738  0.198357  0.417635  1.005832

      thalach     exang   oldpeak     slope        ca      thal    target
0    0.015443  0.696631  1.087338  2.274579  0.714429  2.148873  0.914529
1    1.633471  0.696631  2.122573  2.274579  0.714429  0.512922  0.914529
2    0.977514  0.696631  0.310912  0.976352  0.714429  0.512922  0.914529
3    1.239897  0.696631  0.206705  0.976352  0.714429  0.512922  0.914529
4    0.583939  1.435481  0.379244  0.976352  0.714429  0.512922  0.914529
..        ...       ...       ...       ...       ...       ...       ...
298  1.165281  1.435481  0.724323  0.649113  0.714429  1.123029  1.093459
299  0.771706  0.696631  0.138373  0.649113  0.714429  1.123029  1.093459
300  0.378132  0.696631  2.036303  0.649113  1.244593  1.123029  1.093459
301  1.515125  1.435481  0.138373  0.649113  0.265082  1.123029  1.093459
302  1.064975  0.696631  0.896862  0.649113  0.265082  0.512922  1.093459

[303 rows x 14 columns]
```



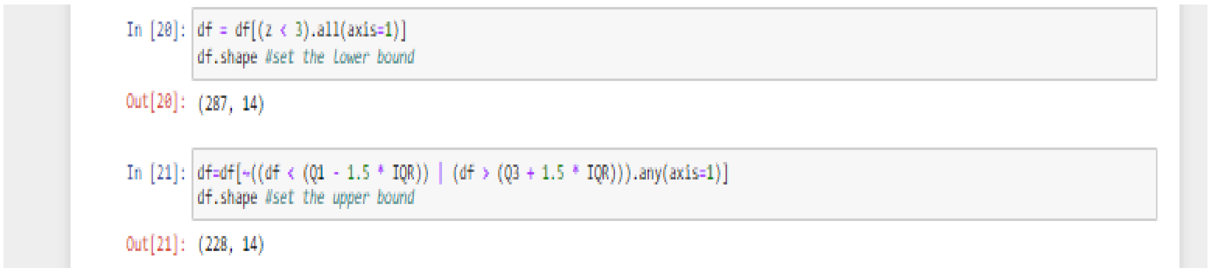
```
In [18]: threshold=3
         # here we'll print z whose greater than third deviation
         print(np.where(z > 3)) #The first array contains the list of row numbers and second array contains respective col

         (array([ 28,  48,  85,  92, 158, 163, 164, 204, 220, 221, 223, 246, 248,
                251, 272, 281], dtype=int64), array([ 4, 12,  4, 11, 11, 11, 11,  9,  4,  9,  3,  4,  3, 11,  7, 12],
               dtype=int64))
```

- After that I am checking Interquartile range



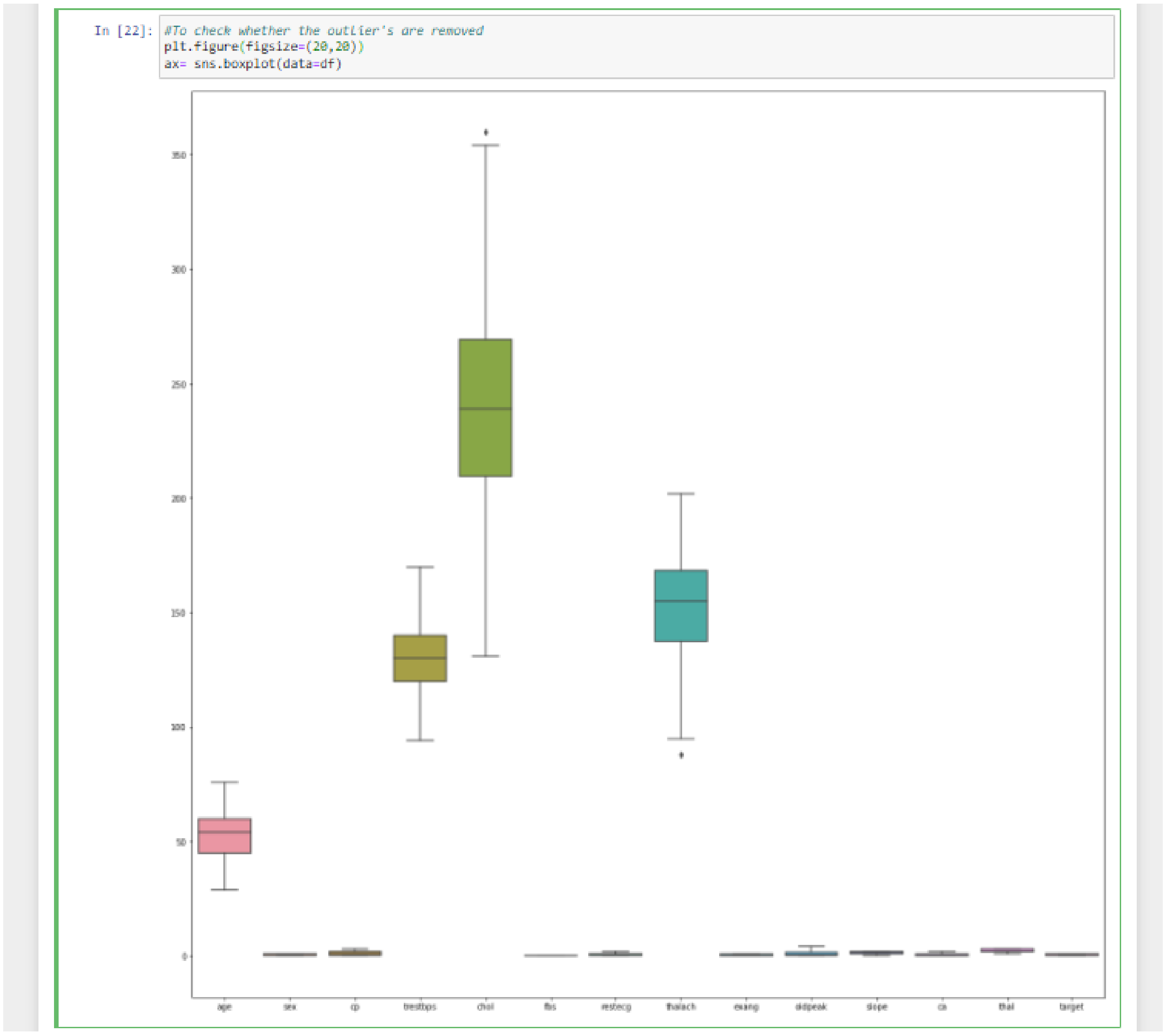
```
In [17]: #Here we'll read and remove outlier's in the dataset using zscore matrix
         from scipy import stats
         z=np.abs(stats.zscore(df))
         print(z)

              age       sex        cp  trestbps      chol       fbs   restecg  \
0    0.952197  0.681005  1.973123  0.763956  0.256334  2.394438  1.005832
1    1.915313  0.681005  1.002577  0.092738  0.072199  0.417635  0.898962
2    1.474158  1.468418  0.032031  0.092738  0.816773  0.417635  1.005832
3    0.180175  0.681005  0.032031  0.663867  0.198357  0.417635  0.898962
4    0.290464  1.468418  0.938515  0.663867  2.082050  0.417635  0.898962
..        ...       ...       ...       ...       ...       ...       ...
298  0.290464  1.468418  0.938515  0.478391  0.101730  0.417635  0.898962
299  1.033002  0.681005  1.973123  1.234996  0.342756  0.417635  0.898962
300  1.503641  0.681005  0.938515  0.706843  1.029353  2.394438  0.898962
301  0.290464  0.681005  0.938515  0.092738  2.227533  0.417635  0.898962
302  0.290464  1.468418  0.032031  0.092738  0.198357  0.417635  1.005832

      thalach     exang   oldpeak     slope        ca      thal    target
0    0.015443  0.696631  1.087338  2.274579  0.714429  2.148873  0.914529
1    1.633471  0.696631  2.122573  2.274579  0.714429  0.512922  0.914529
2    0.977514  0.696631  0.310912  0.976352  0.714429  0.512922  0.914529
3    1.239897  0.696631  0.206705  0.976352  0.714429  0.512922  0.914529
4    0.583939  1.435481  0.379244  0.976352  0.714429  0.512922  0.914529
..        ...       ...       ...       ...       ...       ...       ...
298  1.165281  1.435481  0.724323  0.649113  0.714429  1.123029  1.093459
299  0.771706  0.696631  0.138373  0.649113  0.714429  1.123029  1.093459
300  0.378132  0.696631  2.036303  0.649113  1.244593  1.123029  1.093459
301  1.515125  1.435481  0.138373  0.649113  0.265082  1.123029  1.093459
302  1.064975  0.696631  0.896862  0.649113  0.265082  0.512922  1.093459

[303 rows x 14 columns]
```

- After that I’m checking and set the lower and upper bound.



```
In [20]: df = df[(z < 3).all(axis=1)]
         df.shape #set the lower bound

Out[20]: (287, 14)

In [21]: df=df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)]
         df.shape #set the upper bound

Out[21]: (228, 14)
```

- As it has reduced the number of rows now we can check if the outlier’s still exist.

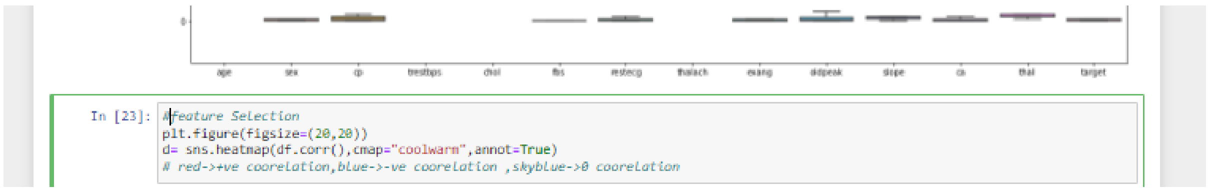


```
In [22]: #To check whether the outlier's are removed
         plt.figure(figsize=(20,20))
         ax= sns.boxplot(data=df)
```

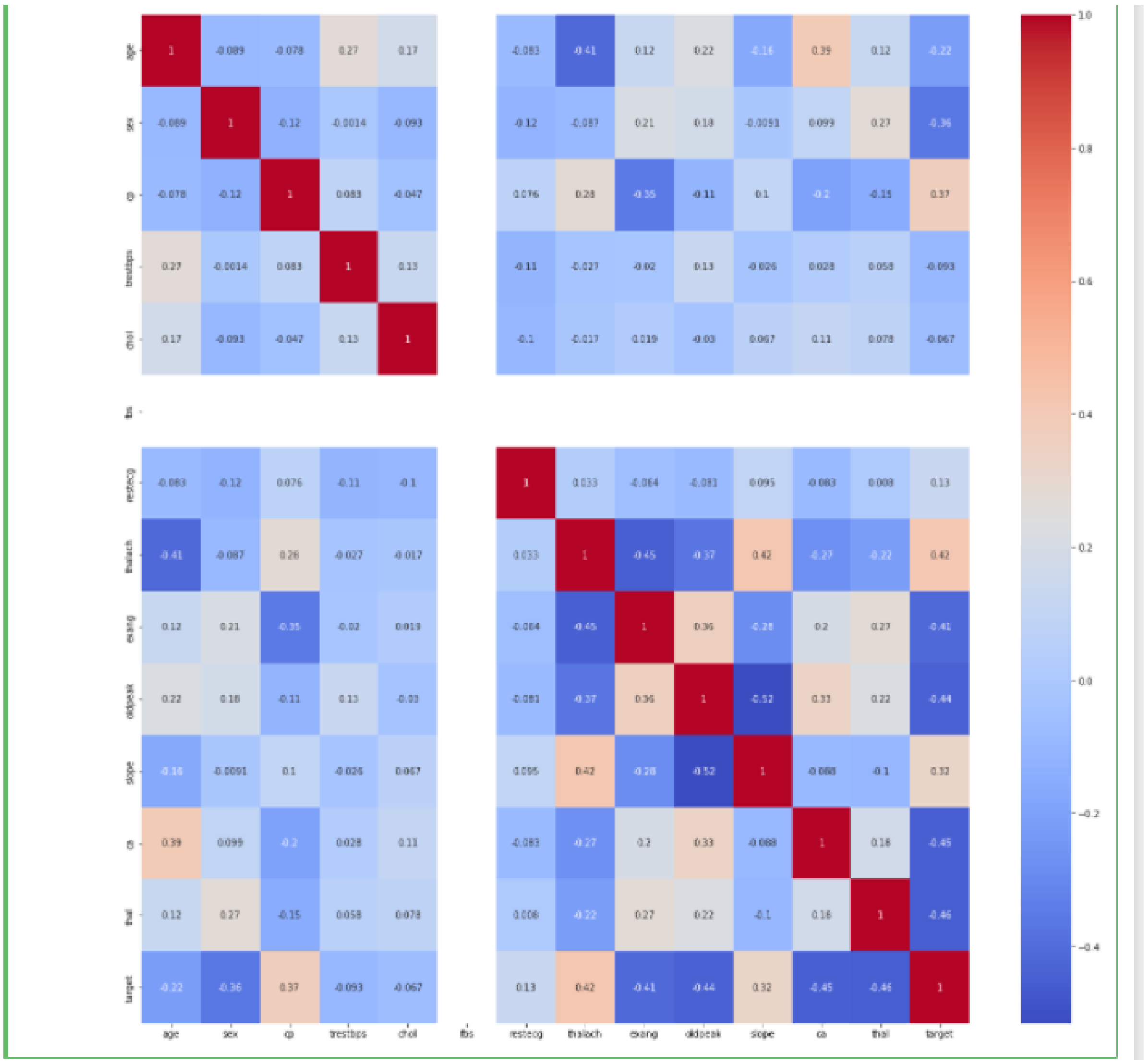
- As the outlier’s has been reduced to minimum.

5 . **FEATURE SELECTION:-**

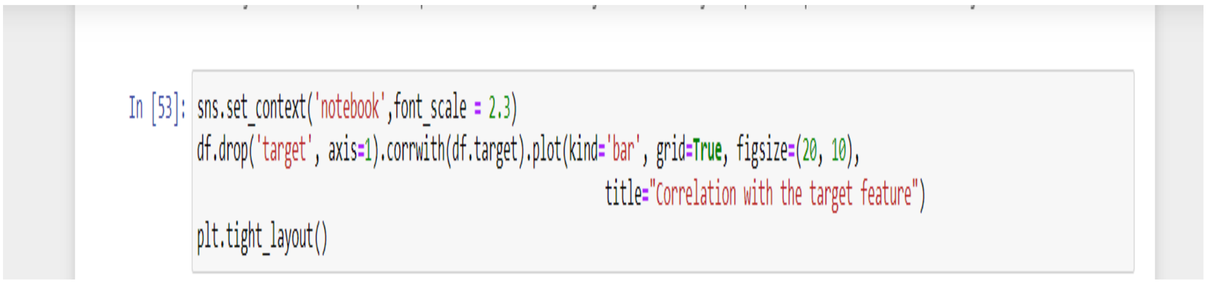
- Here I am checking correlation between various features to analayze which Feature is positively corelated and which is negative corelated.



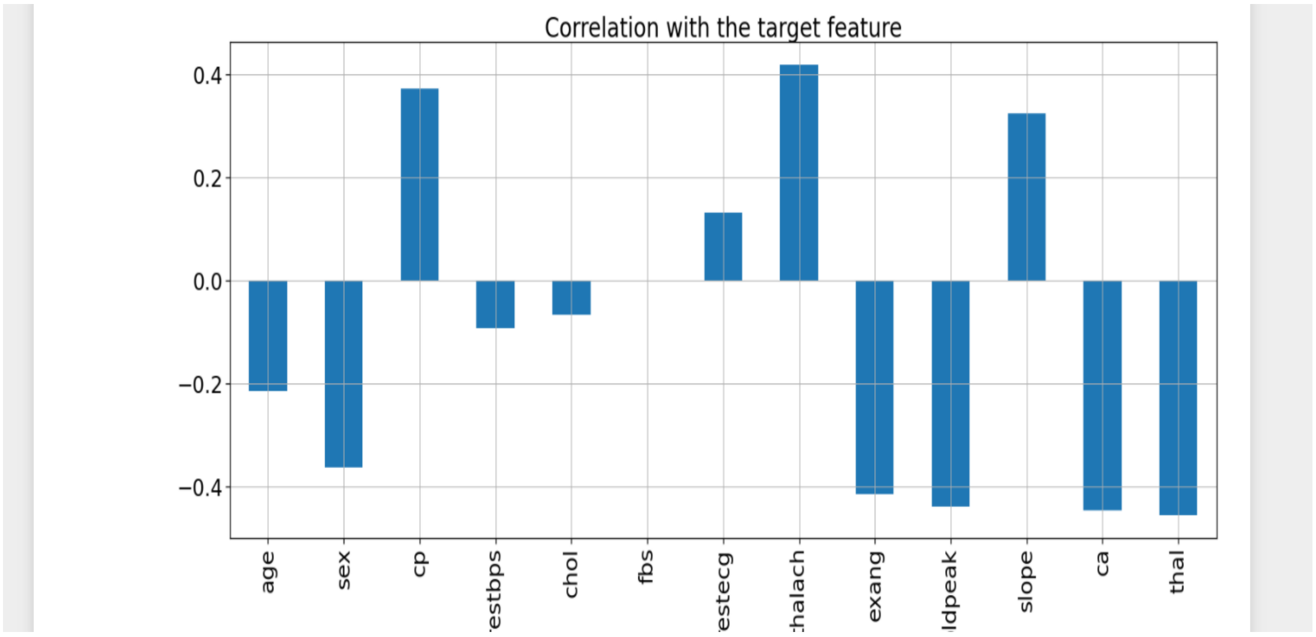
```
In [23]: #Feature Selection
         plt.figure(figsize=(20,20))
         d= sns.heatmap(df.corr(),cmap="coolwarm",annot=True)
         # red->+ve coorelation,blue->-ve coorelation ,skyblue->0 coorelation
```



- Now ,I am checking correlation of the target variable between the features.



```
In [53]: sns.set_context('notebook',font_scale = 2.3)
         df.drop('target', axis=1).corrwith(df.target).plot(kind='bar', grid=True, figsize=(20, 10),
                                                        title="Correlation with the target feature")
         plt.tight_layout()
```



- From The above graph we can clearly analyze:

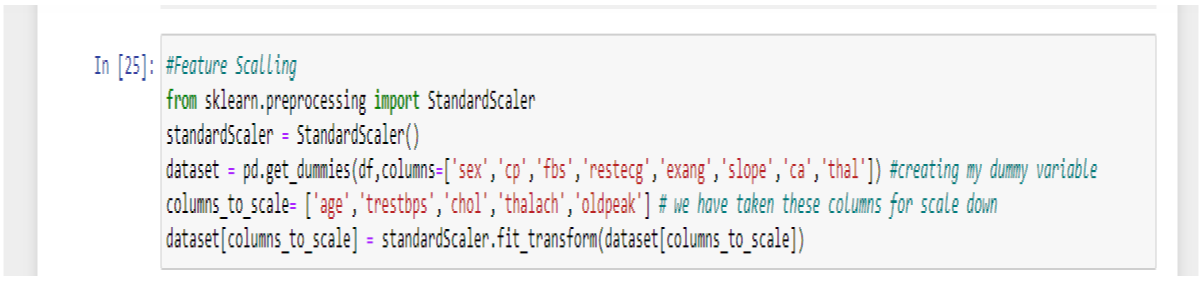
1. Four feature i.e **cp, restecg, thalach, slope** are positively correlated with the target feature.

2. Other features are negatively correlated with the target feature.

6 . <u>FEATURE SCALLING:-</u>

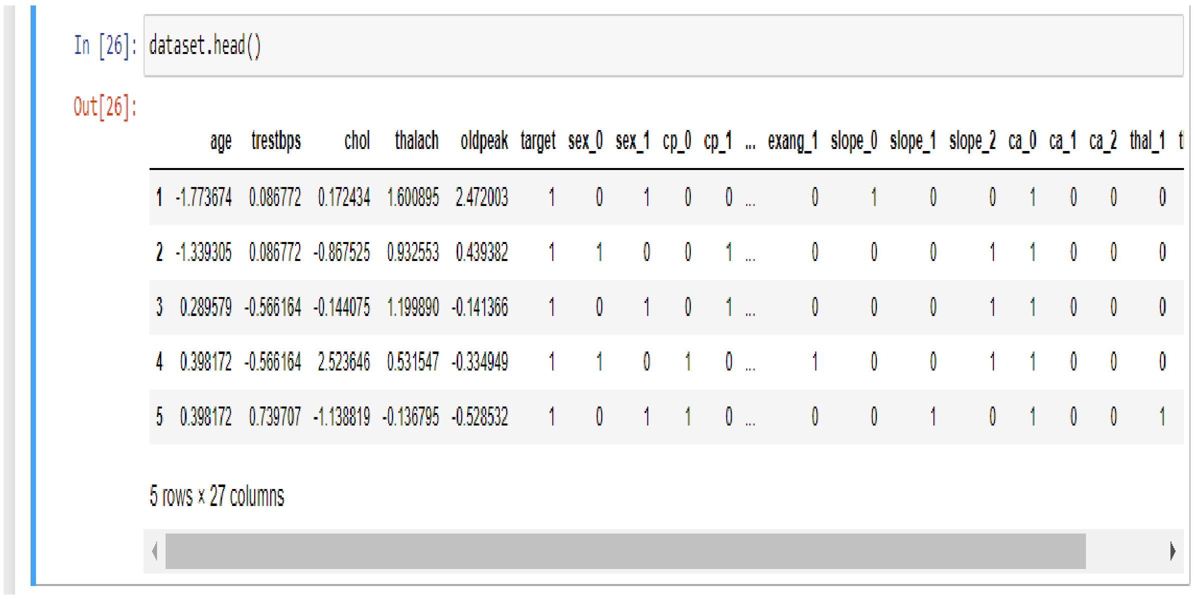
- After exploring the dataset, I observed that I need to convert some categorical variables into dummy variables and scale all the values before training the Machine Learning models. First, I'll use the get_dummies method to create dummy columns for categorical variables.

- In this step, we normalize our data by applying Standardization by using **StandardScalar( )** and **fit_transform( )** functions of scikit-learn library



```
In [25]: #Feature Scalling
         from sklearn.preprocessing import StandardScaler
         standardScaler = StandardScaler()
         dataset = pd.get_dummies(df,columns=['sex','cp','fbs','restecg','exang','slope','ca','thal']) #creating my dummy variable
         columns_to_scale= ['age','trestbps','chol','thalach','oldpeak'] # we have taken these columns for scale down
         dataset[columns_to_scale] = standardScaler.fit_transform(dataset[columns_to_scale])
```

- Here I have choosen data from head i.e from top

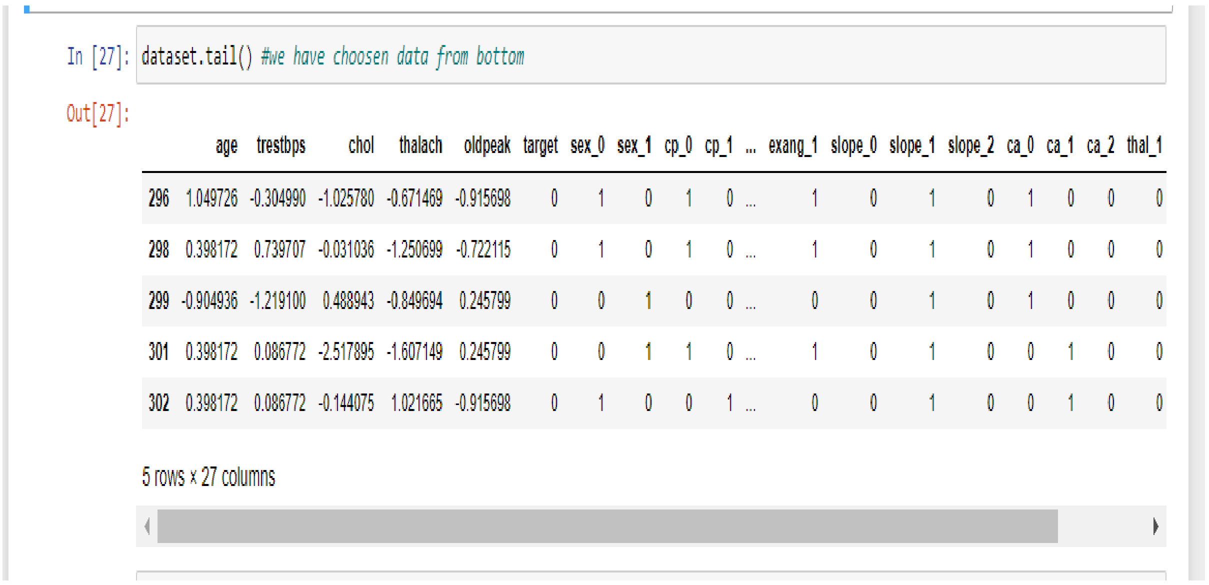


```
In [26]: dataset.head()

Out[26]:
```

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	exang_1	slope_0	slope_1	slope_2	ca_0	ca_1	ca_2	thal_1	t
1	-1.773674	0.086772	0.172434	1.600895	2.472003	1	0	1	0	0	...	0	1	0	0	1	0	0	0	
2	-1.339305	0.086772	-0.867525	0.932553	0.439382	1	1	0	0	1	...	0	0	0	1	1	0	0	0	
3	0.289579	-0.566164	-0.144075	1.199890	-0.141366	1	0	1	0	1	...	0	0	0	1	1	0	0	0	
4	0.398172	-0.566164	2.523646	0.531547	-0.334949	1	1	0	1	0	...	1	0	0	1	1	0	0	0	
5	0.398172	0.739707	-1.138819	-0.136795	-0.528532	1	0	1	1	0	...	0	0	1	0	1	0	0	1	

5 rows × 27 columns

- Again selecting data from bottom i.e tail.



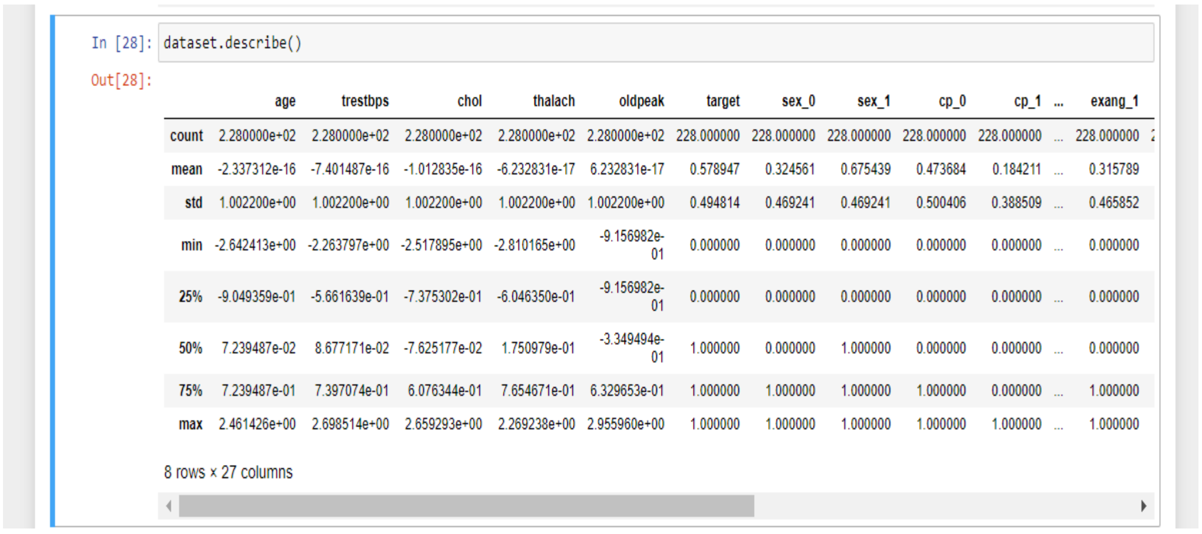
```
In [27]: dataset.tail() #we have choosen data from bottom
```

Out[27]:

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	exang_1	slope_0	slope_1	slope_2	ca_0	ca_1	ca_2	thal_1
296	1.049726	-0.304990	-1.025780	-0.671469	-0.915698	0	1	0	1	0	...	1	0	1	0	1	0	0	0
298	0.398172	0.739707	-0.031036	-1.250699	-0.722115	0	1	0	1	0	...	1	0	1	0	1	0	0	0
299	-0.904936	-1.219100	0.488943	-0.849694	0.245799	0	0	1	0	0	...	0	0	1	0	1	0	0	0
301	0.398172	0.086772	-2.517895	-1.607149	0.245799	0	0	1	1	0	...	1	0	1	0	0	1	0	0
302	0.398172	0.086772	-0.144075	1.021665	-0.915698	0	1	0	0	1	...	0	0	1	0	0	1	0	0

5 rows × 27 columns

- Again using



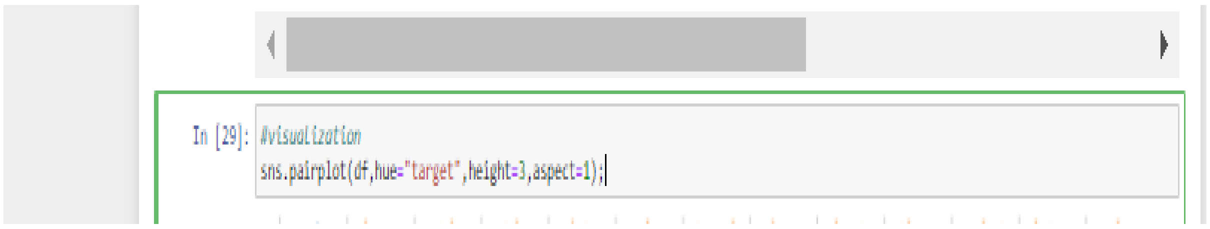
```
In [28]: dataset.describe()
```

Out[28]:

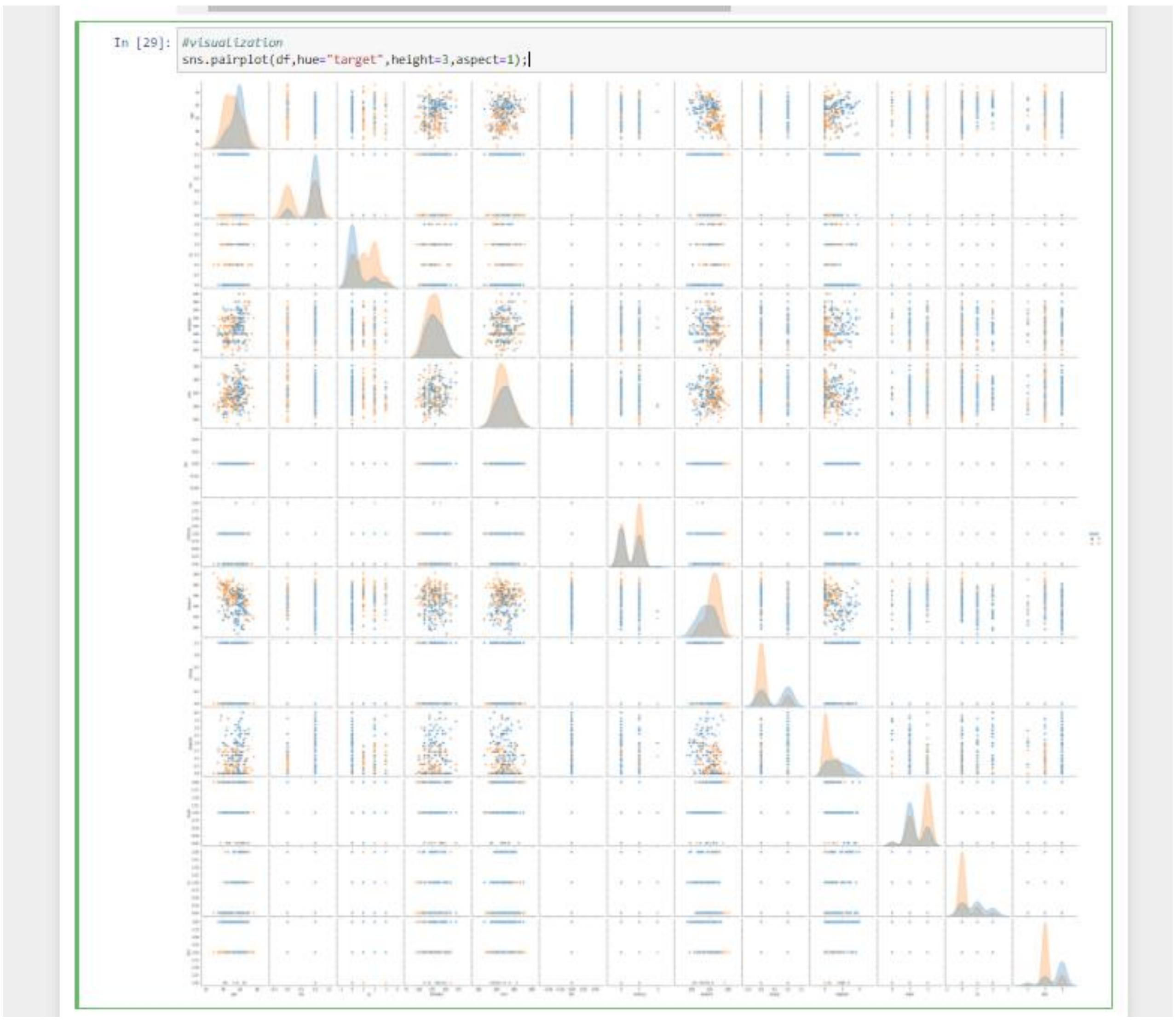
	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	exang_1
count	2.280000e+02	2.280000e+02	2.280000e+02	2.280000e+02	2.280000e+02	228.000000	228.000000	228.000000	228.000000	228.000000	...	228.000000
mean	-2.337312e-16	-7.401487e-16	-1.012835e-16	-6.232831e-17	6.232831e-17	0.578947	0.324561	0.675439	0.473684	0.184211	...	0.315789
std	1.002200e+00	1.002200e+00	1.002200e+00	1.002200e+00	1.002200e+00	0.494814	0.469241	0.469241	0.500406	0.388509	...	0.465852
min	-2.642413e+00	-2.263797e+00	-2.517895e+00	-2.810165e+00	-9.156982e-01	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
25%	-9.049359e-01	-5.661639e-01	-7.375302e-01	-6.046350e-01	-9.156982e-01	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
50%	7.239487e-02	8.677171e-02	-7.625177e-02	1.750979e-01	-3.349494e-01	1.000000	0.000000	1.000000	0.000000	0.000000	...	0.000000
75%	7.239487e-01	7.397074e-01	6.076344e-01	7.654671e-01	6.329653e-01	1.000000	1.000000	1.000000	1.000000	0.000000	...	1.000000
max	2.461426e+00	2.698514e+00	2.659293e+00	2.269238e+00	2.955960e+00	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000

8 rows × 27 columns

- Using The **pairplot( )** function of seaborn helps in creating an axes grid through which each numeric variable present in data is shared across y-axes in the form of rows and across x-axes in form of a column. Scatter plots are created to show pairwise relationships and in the diagonal, the distribution plot is created to show the distribution of the data in the column.



```
In [29]: #visualization
         sns.pairplot(df,hue="target",height=3,aspect=1);
```

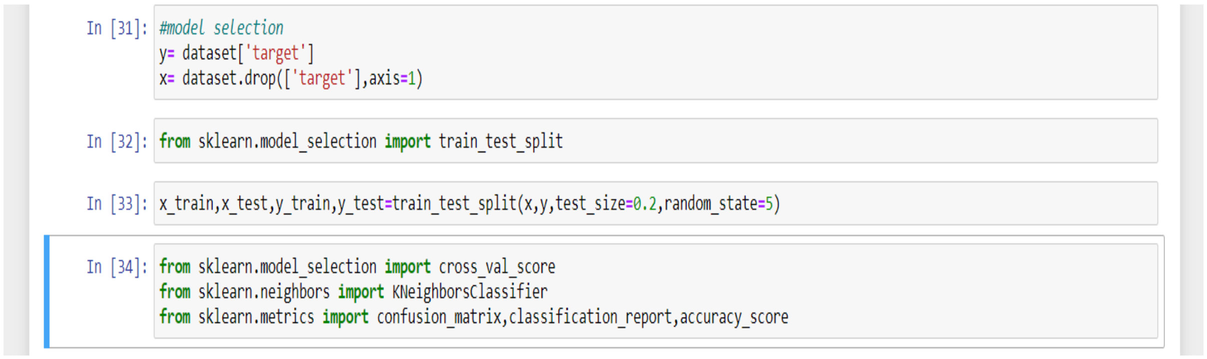


```
In [29]: #visualization
         sns.pairplot(df,hue="target",height=3,aspect=1);
```

6 . MODEL SELECTION & BUILDING:-

A. SPLITTING OUR DATASET :-

Here we have to split data into training and test sets,it’s time to build a machine learning model.
We will train it on the training set and test it on the test set.

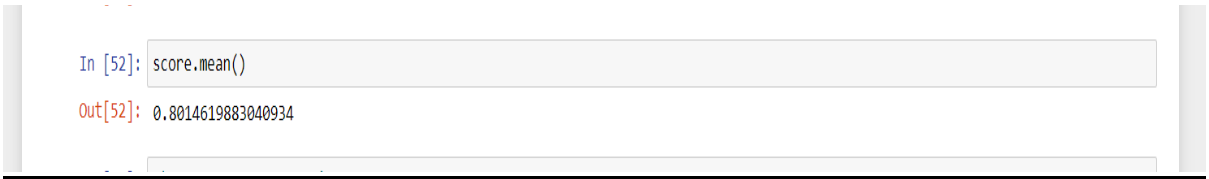


```
In [31]: #model selection
         y= dataset['target']
         x= dataset.drop(['target'],axis=1)

In [32]: from sklearn.model_selection import train_test_split

In [33]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=5)

In [34]: from sklearn.model_selection import cross_val_score
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
```

B. PICKING MODEL :-

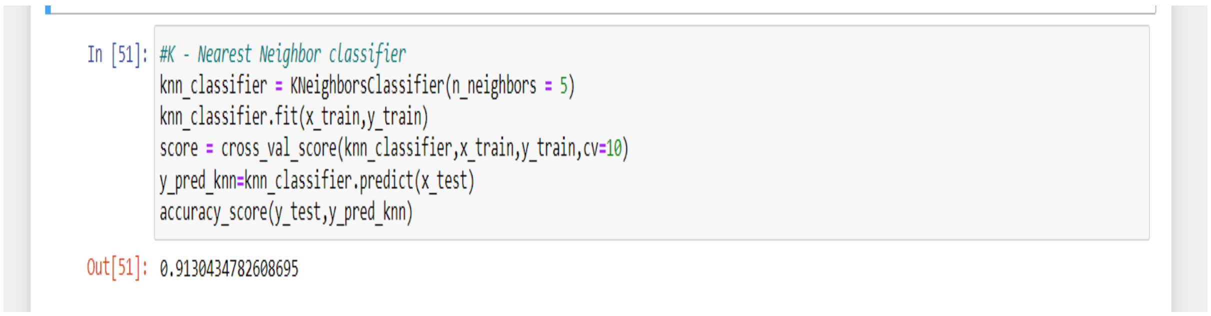
I have used different machine learning models which are:

1. K - Nearest Neighbor classifier
2. Random Forest Classifier
3. XG BOOST
4. Ada Boosting With random forest
5. Gradient Boosting

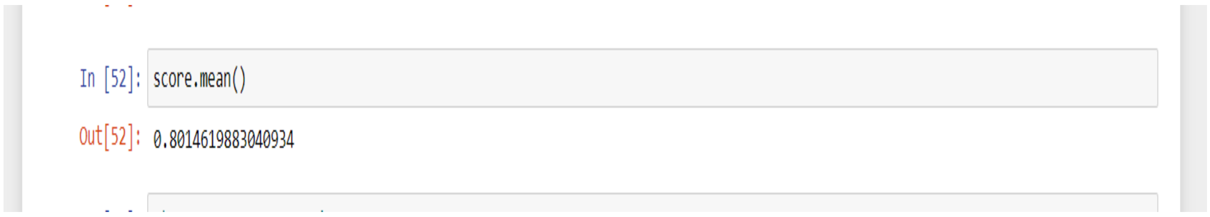


```
In [52]: score.mean()

Out[52]: 0.8014619883040934
```

C. K - Nearest Neighbor Machine Learning Algorithm :-



```
In [51]: #K - Nearest Neighbor classifier
         knn_classifier = KNeighborsClassifier(n_neighbors = 5)
         knn_classifier.fit(x_train,y_train)
         score = cross_val_score(knn_classifier,x_train,y_train,cv=10)
         y_pred_knn=knn_classifier.predict(x_test)
         accuracy_score(y_test,y_pred_knn)

Out[51]: 0.9130434782608695
```

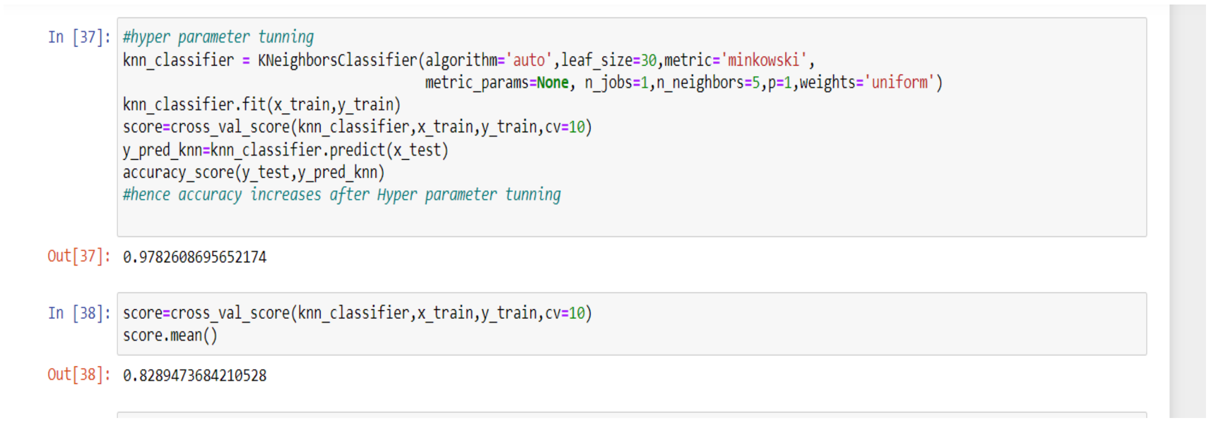


```
In [52]: score.mean()

Out[52]: 0.8014619883040934
```

Therefore after using K-NN we got accuracy of 91.3%

Sr no	Model Name	Accuracy %
0	**K - Nearest Neighbor**	91.3%

Now , we have to perform hyperparameter Tunning:

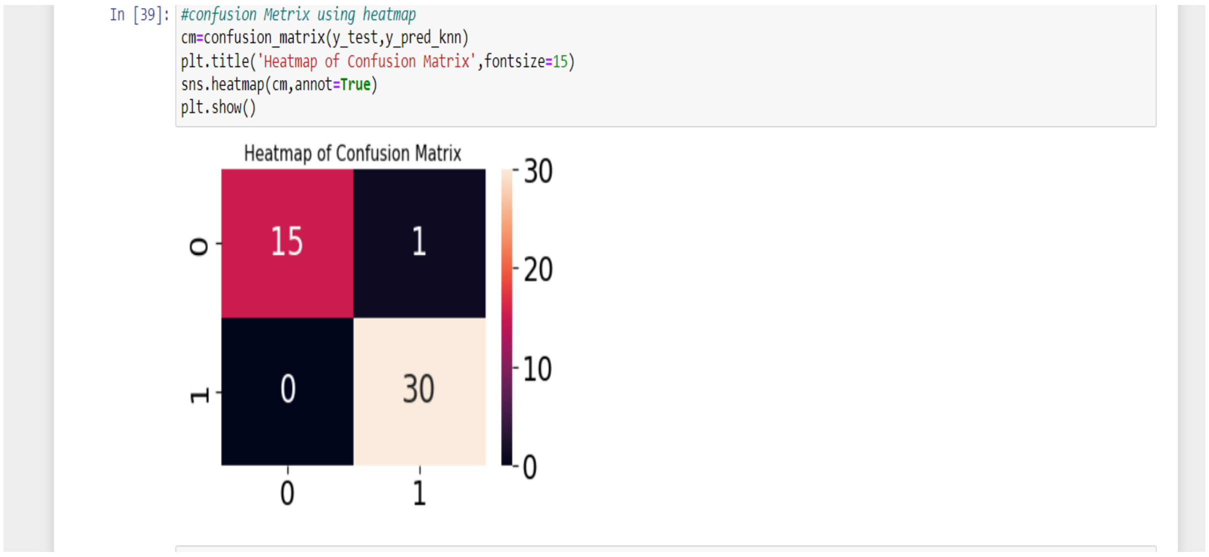


```
In [37]: #hyper parameter tunning
         knn_classifier = KNeighborsClassifier(algorithm='auto',leaf_size=30,metric='minkowski',
                                               metric_params=None, n_jobs=1,n_neighbors=5,p=1,weights='uniform')
         knn_classifier.fit(x_train,y_train)
         score=cross_val_score(knn_classifier,x_train,y_train,cv=10)
         y_pred_knn=knn_classifier.predict(x_test)
         accuracy_score(y_test,y_pred_knn)
         #hence accuracy increases after Hyper parameter tunning

Out[37]: 0.9782608695652174

In [38]: score=cross_val_score(knn_classifier,x_train,y_train,cv=10)
         score.mean()

Out[38]: 0.8289473684210528
```

As after tunning we got maximum accuracy of 97.82

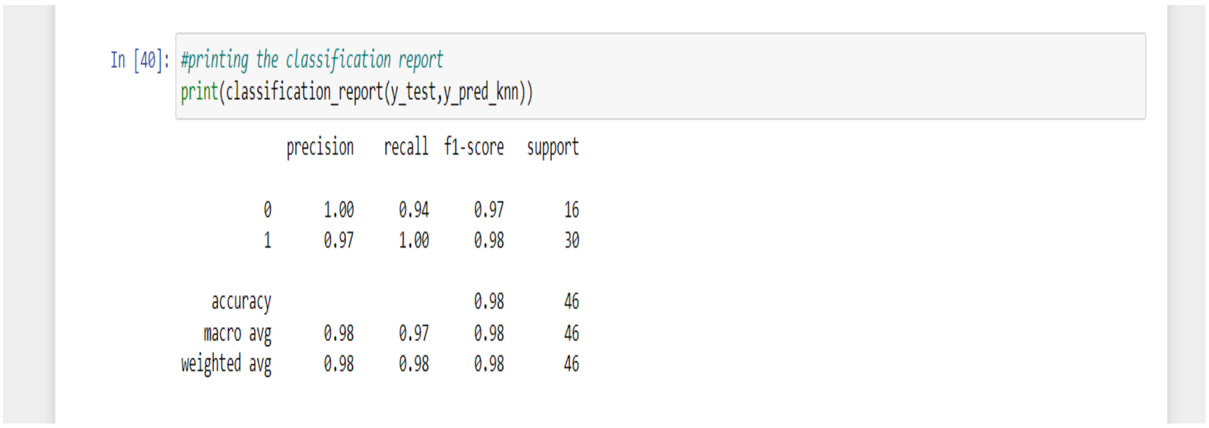
Sr no	Model Name	Accuracy %
0	**Tunned K - Nearest Neighbor**	97.82%

Performing Confusion Matrix using heatmap :



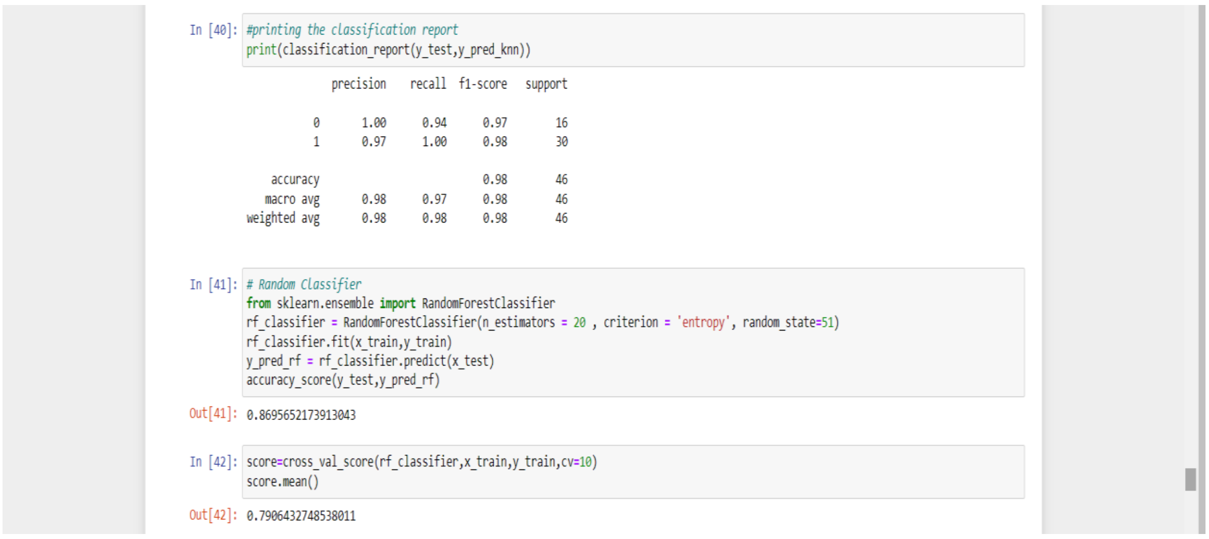
```
In [39]: #confusion Metrix using heatmap
         cm=confusion_matrix(y_test,y_pred_knn)
         plt.title('Heatmap of Confusion Matrix',fontsize=15)
         sns.heatmap(cm,annot=True)
         plt.show()
```

Printing the classification report:-



```
In [40]: #printing the classification report
         print(classification_report(y_test,y_pred_knn))

              precision    recall  f1-score   support

           0       1.00      0.94      0.97        16
           1       0.97      1.00      0.98        30

    accuracy                           0.98        46
   macro avg       0.98      0.97      0.98        46
weighted avg       0.98      0.98      0.98        46
```

D. <u>Random Forest Machine Learning Algorithm :-</u>

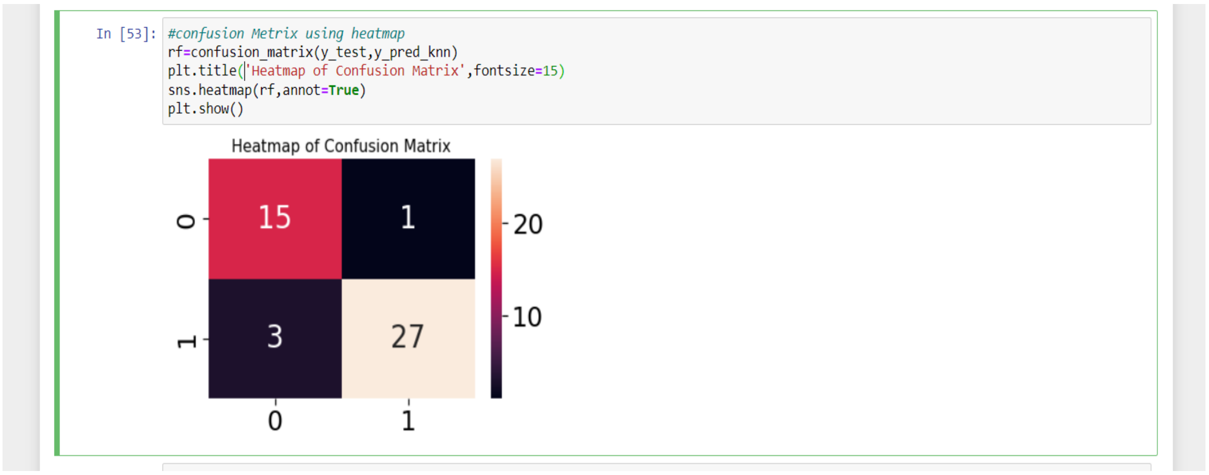


```
In [40]: #printing the classification report
         print(classification_report(y_test,y_pred_knn))

                       precision    recall  f1-score   support

                    0       1.00      0.94      0.97        16
                    1       0.97      1.00      0.98        30

             accuracy                           0.98        46
            macro avg       0.98      0.97      0.98        46
         weighted avg       0.98      0.98      0.98        46


In [41]: # Random Classifier
         from sklearn.ensemble import RandomForestClassifier
         rf_classifier = RandomForestClassifier(n_estimators = 20 , criterion = 'entropy', random_state=51)
         rf_classifier.fit(x_train,y_train)
         y_pred_rf = rf_classifier.predict(x_test)
         accuracy_score(y_test,y_pred_rf)

Out[41]: 0.8695652173913043

In [42]: score=cross_val_score(rf_classifier,x_train,y_train,cv=10)
         score.mean()

Out[42]: 0.7906432748538011
```

After using Random Forest the accuracy achieved was 86.95 % Which is less than K-NN

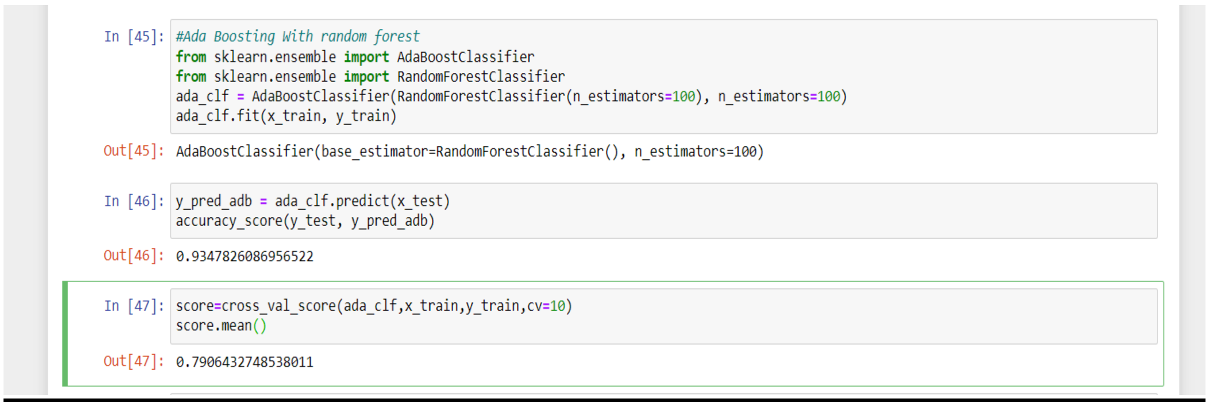
Sr no	Model Name	Accuracy %
0	K - Nearest Neighbor	97.82%
1	Random Forest	86.95 %

Confusion Matrix using Heatmap is :



```
In [53]: #confusion Metrix using heatmap
         rf=confusion_matrix(y_test,y_pred_knn)
         plt.title('Heatmap of Confusion Matrix',fontsize=15)
         sns.heatmap(rf,annot=True)
         plt.show()
```

E. Ada Boosting With Random Forest Machine Learning Algorithm :-

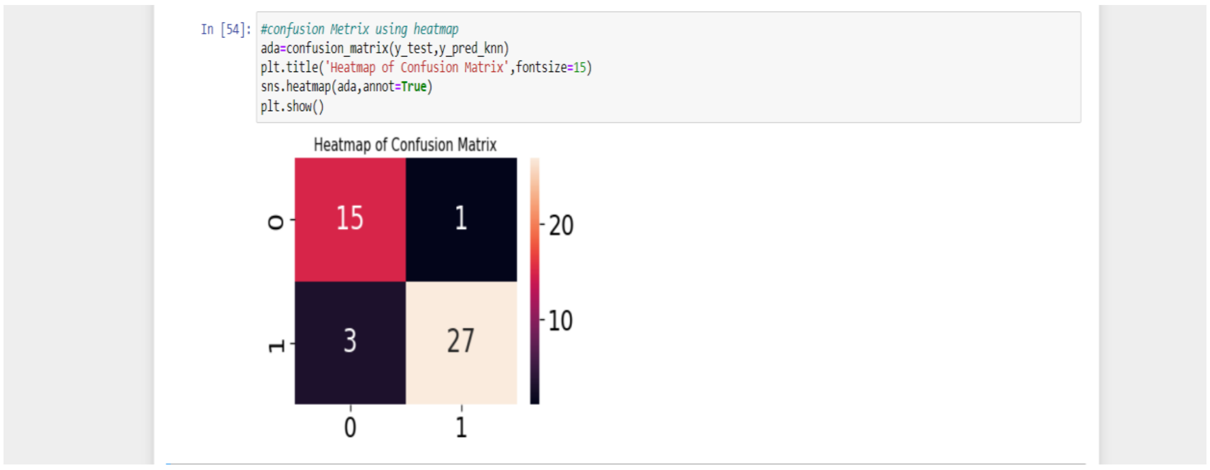


```
In [45]: #Ada Boosting With random forest
         from sklearn.ensemble import AdaBoostClassifier
         from sklearn.ensemble import RandomForestClassifier
         ada_clf = AdaBoostClassifier(RandomForestClassifier(n_estimators=100), n_estimators=100)
         ada_clf.fit(x_train, y_train)

Out[45]: AdaBoostClassifier(base_estimator=RandomForestClassifier(), n_estimators=100)

In [46]: y_pred_adb = ada_clf.predict(x_test)
         accuracy_score(y_test, y_pred_adb)

Out[46]: 0.9347826086956522

In [47]: score=cross_val_score(ada_clf,x_train,y_train,cv=10)
         score.mean()

Out[47]: 0.7906432748538011
```

After Using Ada Boost the accuracy Achieved is 93.47% Which is less than K-NN.

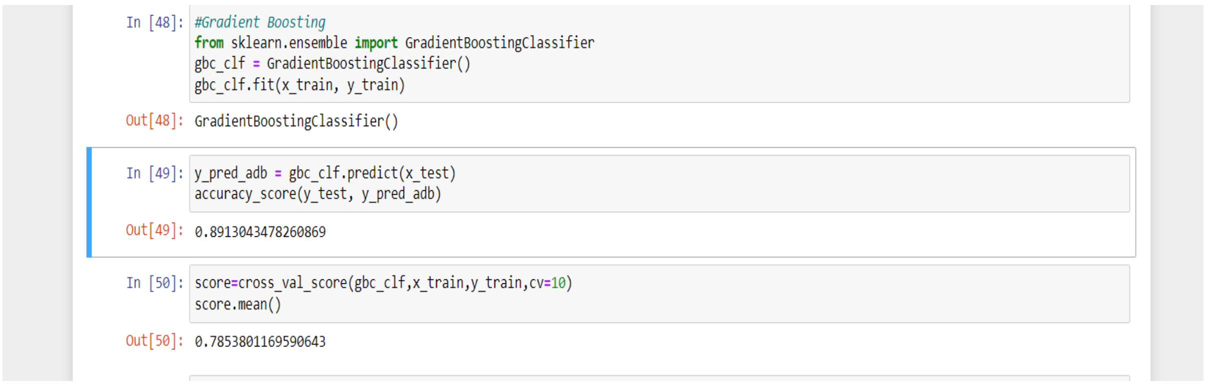
Sr no	Model Name	Accuracy %
0	K - Nearest Neighbor	97.82%
1	Random Forest	86.95 %
2	Ada Boost With Random Forest	93.47 %

Confusion Matrix using heatmap is :-



```
In [54]: #confusion Metrix using heatmap
         ada=confusion_matrix(y_test,y_pred_knn)
         plt.title('Heatmap of Confusion Matrix',fontsize=15)
         sns.heatmap(ada,annot=True)
         plt.show()
```

F. Gradient Boosting Machine Learning Algorithm :-

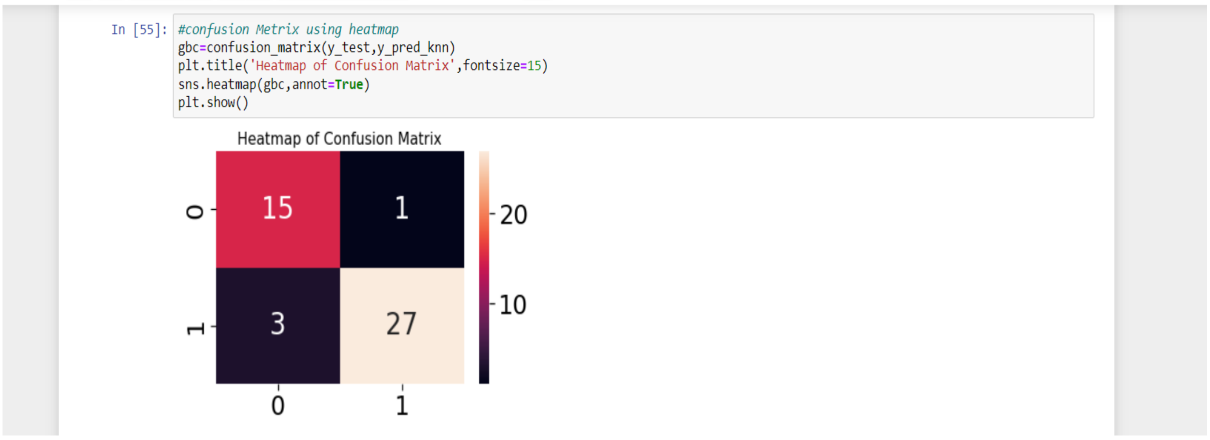


```
In [48]: #Gradient Boosting
         from sklearn.ensemble import GradientBoostingClassifier
         gbc_clf = GradientBoostingClassifier()
         gbc_clf.fit(x_train, y_train)

Out[48]: GradientBoostingClassifier()

In [49]: y_pred_adb = gbc_clf.predict(x_test)
         accuracy_score(y_test, y_pred_adb)

Out[49]: 0.8913043478260869

In [50]: score=cross_val_score(gbc_clf,x_train,y_train,cv=10)
         score.mean()

Out[50]: 0.7853801169590643
```

Using Gradient boosting the accuracy achieved is 89.1% Which is less than K-NN.

Sr no	Model Name	Accuracy %
0	K - Nearest Neighbor	97.82%
1	Random Forest	86.95 %
2	Ada Boost With Random Forest	93.47 %
3	Gradient Boosting	89.91%

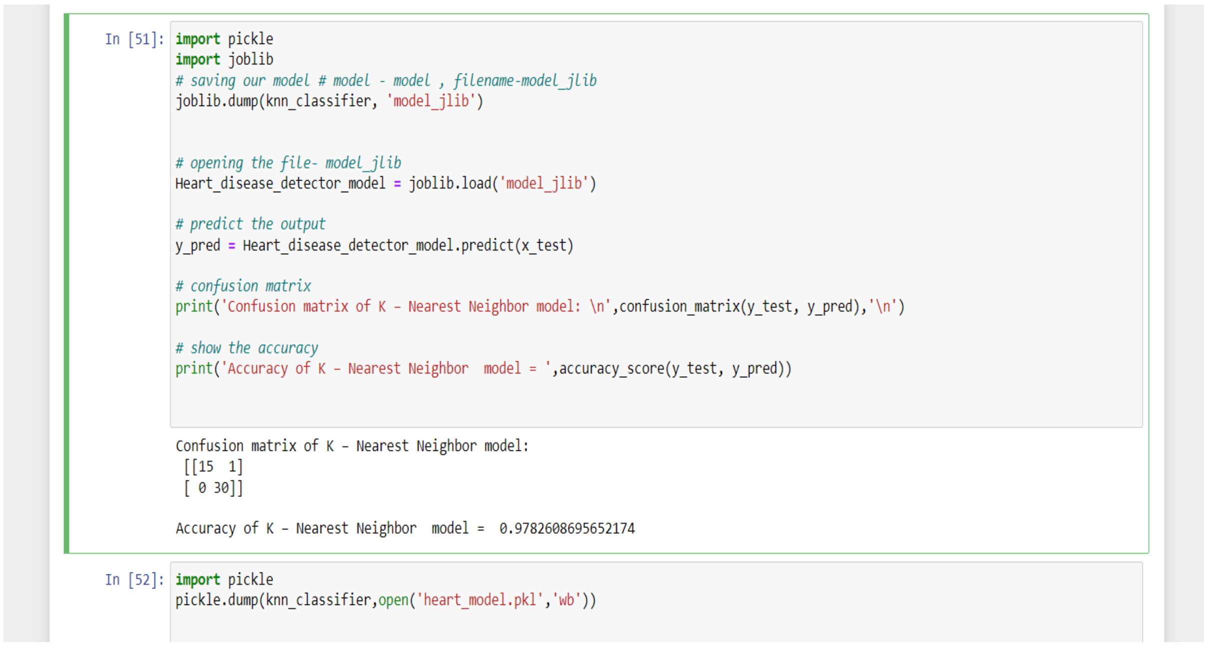
Confusion Matrix using heatmap is :



```
In [55]: #confusion Metrix using heatmap
         gbc=confusion_matrix(y_test,y_pred_knn)
         plt.title('Heatmap of Confusion Matrix',fontsize=15)
         sns.heatmap(gbc,annot=True)
         plt.show()
```

After applying all the algorithm the highest accuracy achieved was of **K - Nearest Neighbor** i.e **97.82%.**

Sr no	Model Name	Accuracy %
0	**K - Nearest Neighbor**	**97.82%**
1	**Random Forest**	**86.95 %**
2	**Ada Boost With Random Forest**	**93.47 %**
3	**Gradient Boosting**	**89.91%**

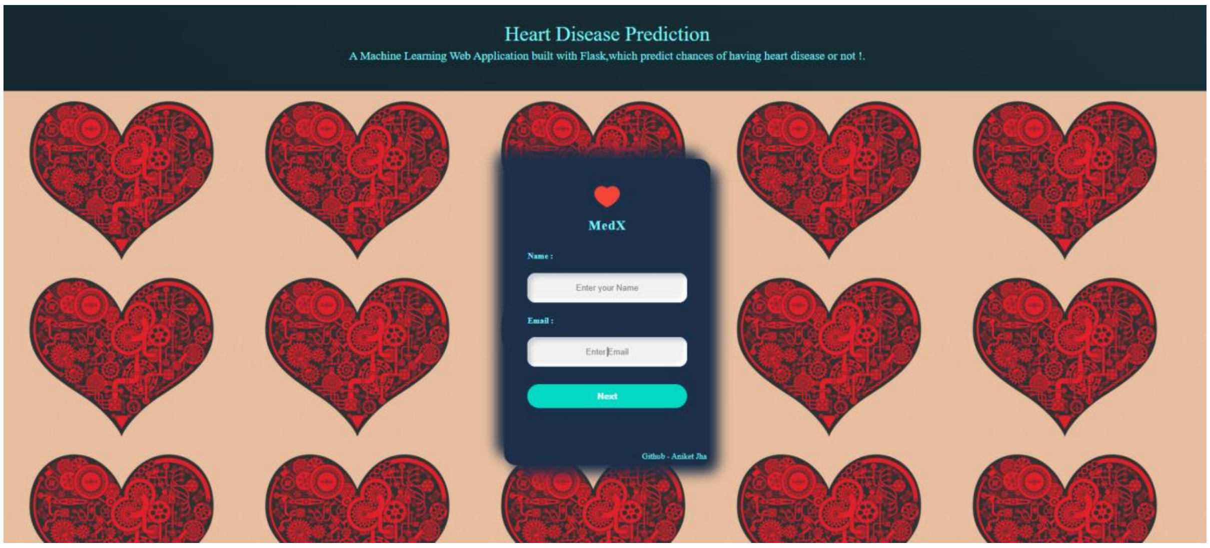


```
In [51]: import pickle
         import joblib
         # saving our model # model - model , filename-model_jlib
         joblib.dump(knn_classifier, 'model_jlib')


         # opening the file- model_jlib
         Heart_disease_detector_model = joblib.load('model_jlib')

         # predict the output
         y_pred = Heart_disease_detector_model.predict(x_test)

         # confusion matrix
         print('Confusion matrix of K – Nearest Neighbor model: \n',confusion_matrix(y_test, y_pred),'\n')

         # show the accuracy
         print('Accuracy of K – Nearest Neighbor  model = ',accuracy_score(y_test, y_pred))
```

```
Confusion matrix of K – Nearest Neighbor model:
 [[15  1]
 [ 0 30]]

Accuracy of K – Nearest Neighbor  model =  0.9782608695652174
```

```
In [52]: import pickle
         pickle.dump(knn_classifier,open('heart_model.pkl','wb'))
```

Result:-


Heart Disease Prediction

A Machine Learning Web Application built with Flask,which predict chances of having heart disease or not !.

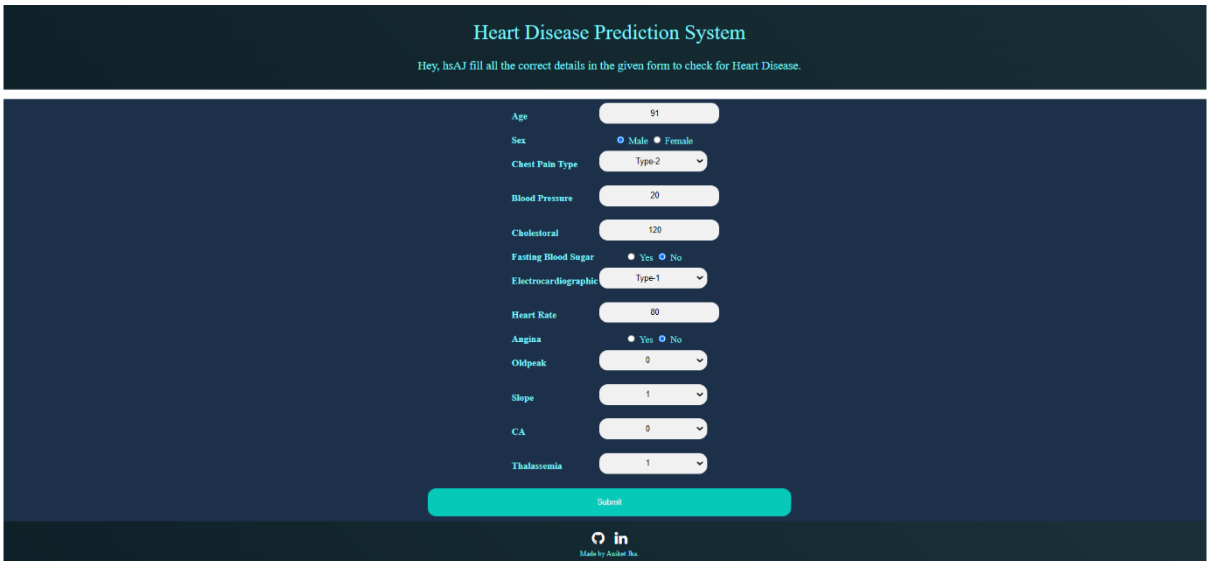
MedX

Name :

Enter your Name

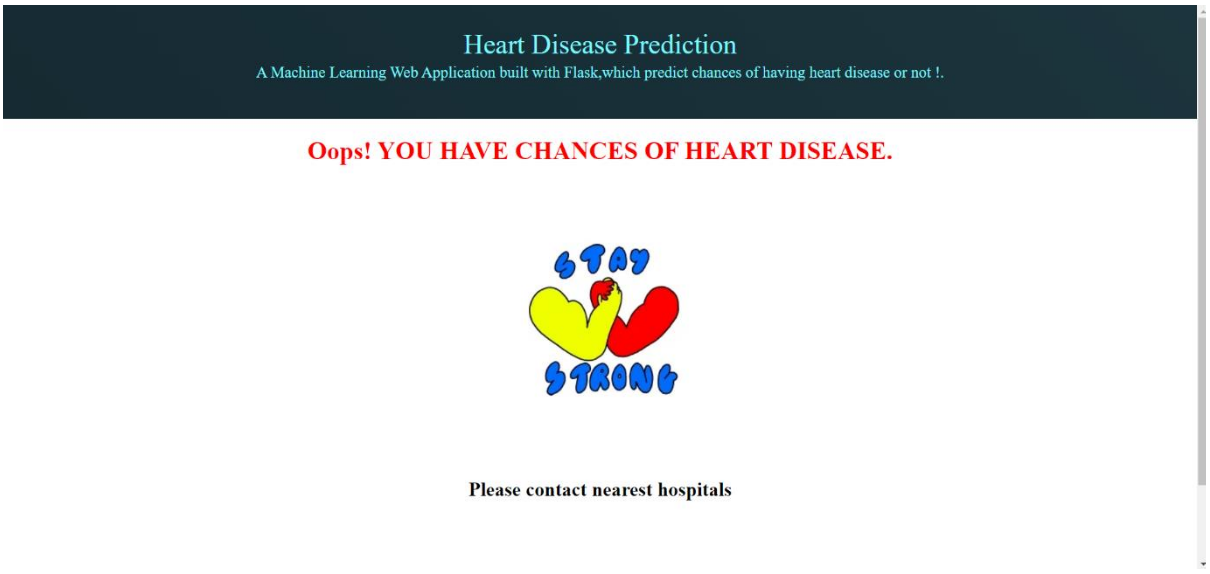
Email :

Enter Email


Heart Disease Prediction System

Hey, hsAJ fill all the correct details in the given form to check for Heart Disease.

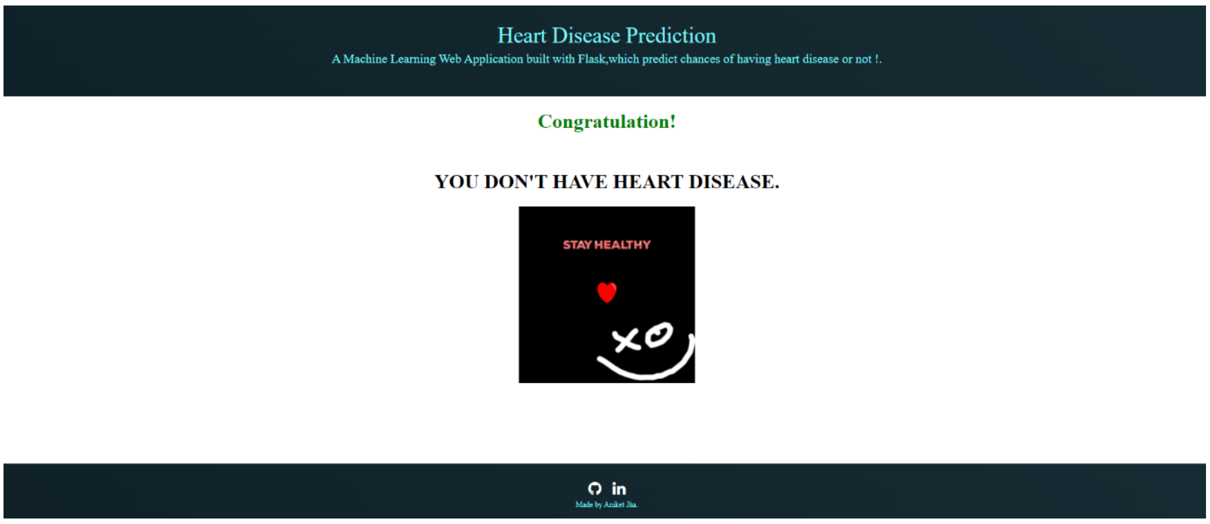
Age	91
Sex	[✓ Male] Female
Chest Pain Type	Type-2
Blood Pressure	20
Cholestoral	120
Fasting Blood Sugar	Yes [✓ No]
Electrocardiographic	Type-1
Heart Rate	80
Angina	Yes [✓ No]
Oldpeak	0
Slope	1
CA	0
Thalassemia	1


Heart Disease Prediction

A Machine Learning Web Application built with Flask,which predict chances of having heart disease or not !.

Oops! YOU HAVE CHANCES OF HEART DISEASE.

Please contact nearest hospitals


Heart Disease Prediction

A Machine Learning Web Application built with Flask,which predict chances of having heart disease or not !.

Congratulation!

YOU DON'T HAVE HEART DISEASE.

Made by Aniket Jha.

FUTURE SCOPE:-

Today’s, world most of the data is computerized, the data is distributed and it is not utilizing properly. By Analyzing the available data we can also use for unknown patterns. The primary motive of this research is the prediction of heart diseases with high rate of accuracy.

The future scope of this system aims at giving more sophisticated prediction models, risk calculation tools and feature extraction tools for other clinical risks. This method can also be used to select the proper treatment methods for a patient in future, instead of just predicting the chances of developing a heart disease among the patient.In future my plan is to add more disease prediction like diabetes, symptoms,Pneumonia detection and many more.With that to provide a platform where user can self diagnosis and can check their health status ,In addition to that I’m also thinking to add doctor detail’s where user can get all the info related to that particular doctor in addition this I’ am also willing to add appointment and give user a functionality where they can get the best hospital near them with just giving PIN code which will extract provide that.To pack all these and to eradicate communication difference I am also willing to add a Voice assistant which user can just do all these with just voice.

CONCLUSION

Heart acts a major role in corporeal organism. The diseases of heart wants more perfection and exactness for diagnose and analyses.In real time heart diseases may not be detect in early stage. Due to heart disease, there is an increased in the number of deaths, day by day. The implementation of a method to efficiently and reliably predict heart diseases has become compulsory. The main motivation of this study is to find a powerful ML algorithm for detection of heart disease.

This project provides the deep insight into machine learning techniques for classification of heart diseases. The role of classifier is crucial in healthcare industry so that the results can be used for predicting the treatment which can be provided to patients. The existing techniques are studied and compared for finding the efficient and accurate systems. Machine learning techniques significantly improves accuracy of cardiovascular risk prediction through which patients can be identified during an early stage of disease and can be benefitted by preventive treatment

In these project I have used **K - Nearest Neighbour, Random Forest, Gradient Boost Ada Boost With Random Forest and the highest accuracy achived is as follows:**

Sr no	Model Name	Accuracy %
0	**K - Nearest Neighboor**	**97.82%**
1	**Random Forest**	**86.95 %**
2	**Ada Boost With Random Forest**	**93.47 %**
3	**Gradient Boosting**	**89.91%**

The outcome of this analysis shows that the **K - Nearest Neighbor** algorithm is the most powerful algorithm for heart disease prediction, with an accuracy score of 100%
It can be concluded that there is a huge scope for machine learning algorithms in predicting cardiovascular diseases or heart related diseases.

REFERENCES

1. https://archive.ics.uci.edu/ml/datasets/heart+disease

2. https://en.wikipedia.org/wiki/Cardiovascular_disease.

3. https://stackoverflow .com

4. https://www.cdc.gov/heartdisease/index.htm#:~:text=Heart%20disease%20is%20the%20leading,can%20lead%20to%20heart%20attack.

5. https://www.medicalnewstoday.com/articles/237191

6. https://flask.palletsprojects.com/en/2.1.x/

Github Link

https://github.com/Aniket11007/Heart_Disease_prediction-WebApp