DATA SCIENCE
(Predicting The Occurrence Of Heart Diseases)

Summer Internship Report Submitted in Partial Fulfillment of the requirement for undergraduate degree of

Bachelor of Technology
In Computer Science Engineering
By
Krishna Samhitha Peri
221710305023

Department Of Computer Science Engineering
Gitam School Of Technology
Gitam(Deemed to be University)
Hyderabad-502329
June2020

DECLARATION
I submit this industrial training work entitled "PREDICTING THE OCCURENCE OF HEART DISEASES" to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of "Bachelor of Technology" in "Computer Science Engineering". I declare that it was carried out independently by me under the guidance of , GITAM (Deemed To

Be University), Hyderabad, India. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD	Krishna Samhitha Peri
Date: 13-07-2020	221710305023

CERTIFICATE

TABLE OF CONTENTS:-

CHAPTER 1

DATA SCIENCE

1.1.1 What is Data Science :

Data science is an inter-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data. Data science is related to data mining, deep learning and big data.

Data science is a "concept to unify statistics, data analysis, machine learning, domain knowledge and their related methods" in order to "understand and analyze actual phenomena" with data. It uses techniques and theories drawn from many fields within the context of mathematics, statistics, computer science, domain knowledge and information science. Turing award winner Jim Gray imagined data science as a "fourth paradigm" of science (empirical, theoretical, computational and now data-driven) and asserted that "everything about science is changing because of the impact of information technology" and the data deluge.

1.1.2 Need of Data Science :

The principal purpose of Data Science is to find patterns within data. It uses various statistical techniques to analyze and draw insights from the data. From data extraction, wrangling and pre-processing, a Data Scientist must scrutinize the data thoroughly. Then, he has the responsibility of making predictions from the data. The goal of a Data Scientist is to derive conclusions from the data. Through these conclusions, he is able to assist companies in making smarter business decisions.

Data creates magic. Industries need data to help them make careful decisions. Data Science churns raw data into meaningful insights. Therefore, industries need data science. A Data Scientist is a wizard who knows how to create magic using data. A skilled Data Scientist will know how to dig out meaningful information with whatever data he comes across. He helps the company in the right direction. The company requires strong data-driven decisions at which he’s an expert. The Data Scientist is an expert in various underlying fields of Statistics and Computer Science. He uses his analytical aptitude to solve business problems.

Data Scientist is well versed with problem-solving and is assigned to find patterns in data. His goal is to recognize redundant samples and draw insights from it. Data Science requires a variety of tools to extract information from the data. A Data Scientist is responsible for collecting, storing and maintaining the structured and unstructured form of data.

While the role of Data Science focuses on the analysis and management of data, it is dependent on the area that the company is specialized in. This requires the Data Scientist to have domain knowledge of that particular industry.

1.1.3 Uses of Data Science :

Data scientists tackle questions about the future. They start with big data, characterized by the three V's: volume, variety and velocity. Then, they use it as fodder for algorithms and models. The most cutting-edge data scientists, working in machine learning and AI, make models that automatically self-improve, noting and learning from their mistakes.

Data scientists have changed almost every industry. In medicine, their algorithms help predict patient side effects. In sports, their models and metrics have redefined "athletic potential." Data science has even tackled traffic, with route-optimizing models that capture typical rush hours and weekend lulls.

APPLICATIONS OF DATA SCIENCE :

- Identifying and predicting disease
- Personalized healthcare recommendations
- Optimizing shipping routes in real-time
- Getting the most value out of soccer rosters
- Finding the next slew of world-class athletes
- Stamping out tax fraud
- Automating digital ad placement
- Algorithms that help you find love
- Predicting incarceration rates

CHAPTER 2

MACHINE LEARNING

2.1.1 INTRODUCTION:

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

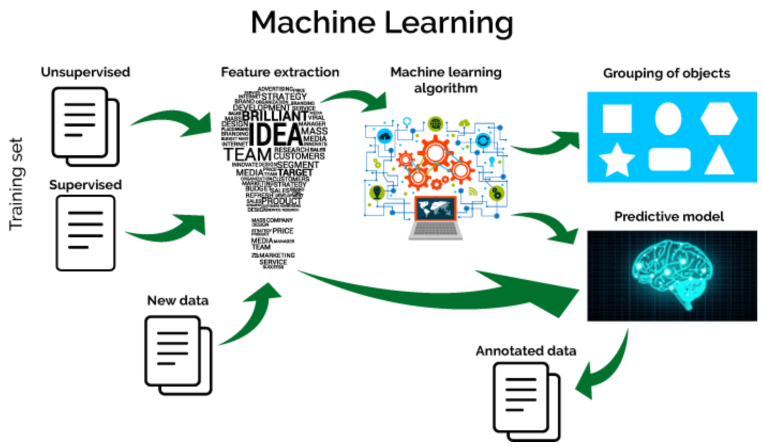
2.1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today’s data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that’s in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works



2.1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data.

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

2.1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

2.1.4 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

2.1.4 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

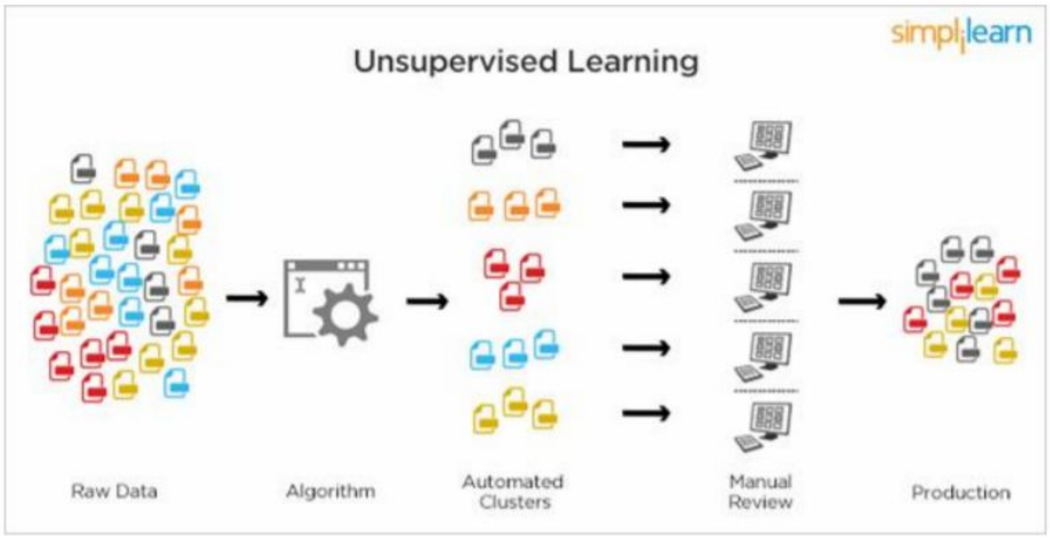


Figure 2 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

2.1.4 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

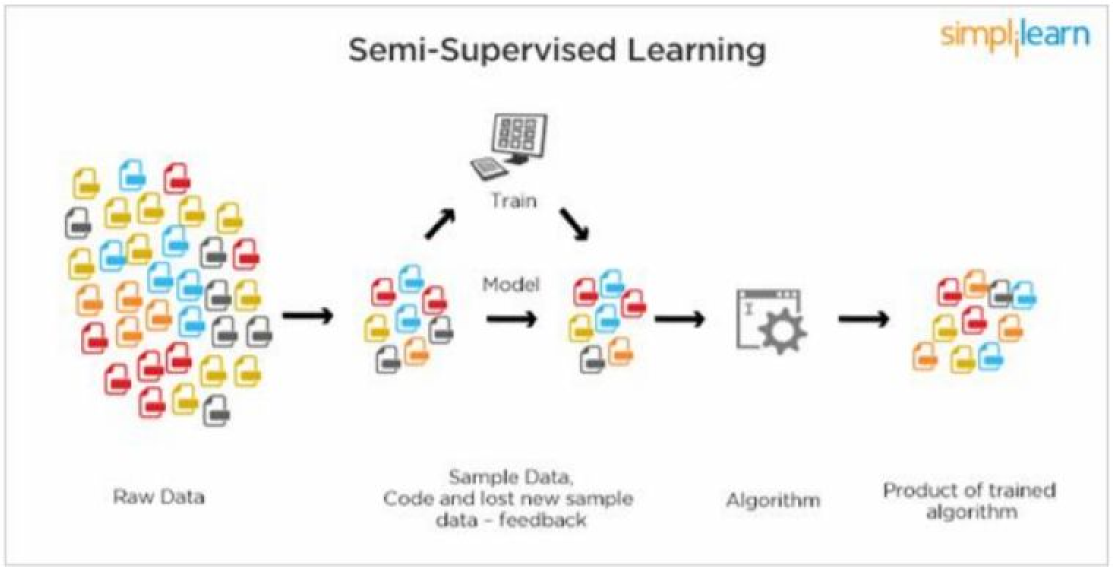


Figure 3 : Semi Supervised Learning

2.1.5 RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER 3

PYTHON

Basic programming language used for machine learning is : PYTHON

3.1 INTRODUCTION TO PYTHON:

● Python is a high-level, interpreted, interactive and object-oriented scripting language.
● Python is a general purpose programming language that is often applied in scripting roles
● Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
● Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.
● Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

3.2 HISTORY OF PYTHON:

● Python was developed by GUIDO VAN ROSSUM in early 1990's .
● Its latest version is 3.7 , it is generally called as python3.

3.3 FEATURES OF PYTHON:

Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.
● Easy-to-read: Python code is more clearly defined and visible to the eyes.
● Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
● A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
● Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
● Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
● Databases: Python provides interfaces to all major commercial databases.

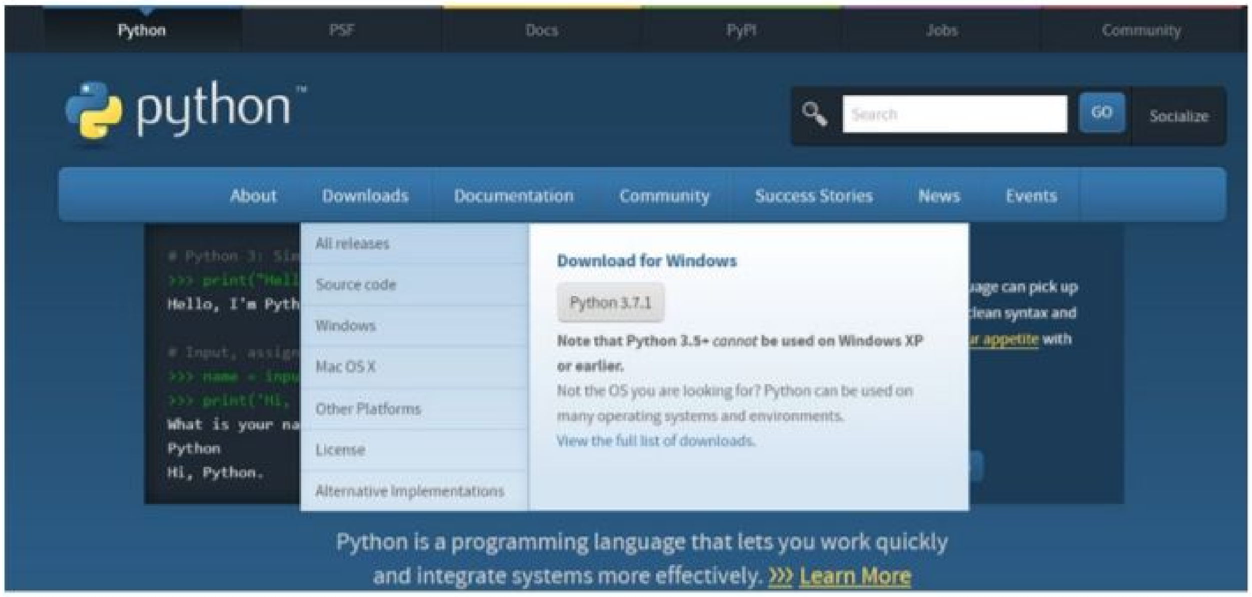
● GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

3.4 HOW TO SETUP PYTHON:

● Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.
● The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

3.4.1 Installation(using python IDLE):

Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
● Download python from www.python.org .
● When the download is completed, double click the file and follow the instructions to install it.
● When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.



Note that Python 3.5+ *cannot* be used on Windows XP or earlier.
Not the OS you are looking for? Python can be used on many operating systems and environments.
View the full list of downloads.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Figure 4 : Python download

3.4.2 Installation(using Anaconda):

● Python programs are also executed using Anaconda.
● Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
● Conda is a package manager quickly installs and manages packages.
● In WINDOWS:
● In windows
● Step 1: Open Anaconda.com/downloads in web browser.
● Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
● Step 3: select installation type(all users)
● Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
● Step 5: Open jupyter notebook (it opens in default browser)

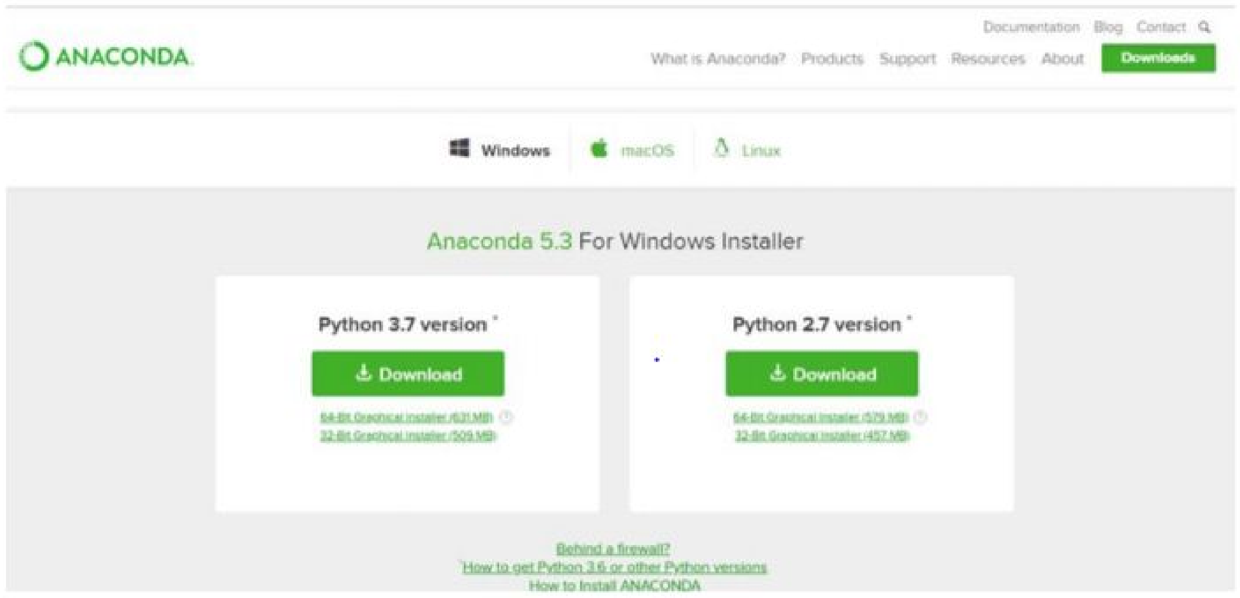

ANACONDA
Documentation Blog Contact
What is Anaconda? Products Support Resources About Downloads
Windows macOS Linux
Anaconda 5.3 For Windows Installer
Python 3.7 version
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (509 MB)
Python 2.7 version
Download
64-Bit Graphical Installer (579 MB)
32-Bit Graphical Installer (457 MB)
Behind a firewall?
How to get Python 3.6 or other Python versions
How to Install ANACONDA

Figure 5 : Anaconda download

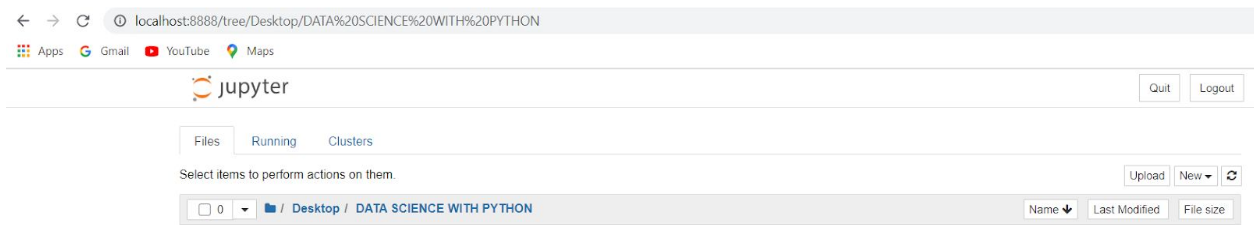

localhost:8888/tree/Desktop/DATA%20SCIENCE%20WITH%20PYTHON
Apps
Gmail
YouTube
Maps
jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
Desktop / DATA SCIENCE WITH PYTHON
Name
Last Modified
File size

Figure 6 : Jupyter notebook

3.5 PYTHON VARIABLE TYPES:

● Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

● Variables are nothing but reserved memory locations to store values.

● Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

● Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

● Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

● Python has five standard data types –

o Numbers
o Strings
o Lists
o Tuples
o Dictionary

3.5.1 Python Numbers:

● Number data types store numeric values. Number objects are created when you assign a value to them.

● Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

3.5.2 Python Strings:

● Strings in Python are identified as a contiguous set of characters represented in the quotation marks.

● Python allows for either pairs of single or double quotes.

3.5.3 Python Lists:

● Lists are the most versatile of Python's compound data types.
● A list contains items separated by commas and enclosed within square brackets
● To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
● The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
● The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

3.5.4 Python Tuples:

● The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
● Tuples can be thought of as read-only lists.
● For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

3.5.5 Python Dictionary:

● Python's dictionaries are kind of hash table type. They work like associative arrays or hashes in pearl.
● Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
● You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
● What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

3.6 PYTHON FUNCTION:

3.6.1 Defining a Function: You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()). Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

3.6.2 Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. 2.7 PYTHON USING OOP's CONCEPTS: 2.7.1 Class:

● Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

● Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables arenot used as frequently as instance variables are.

● Data member: A class variable or instance variable that holds data associated with a class and its objects.

● Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

● Defining a Class:

o We define a class in a very similar way how we define a function.

o Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is

3.6.3 __init__ method in Class:

● The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.

● The init method has a special name that starts and ends with two underscores:__init__().

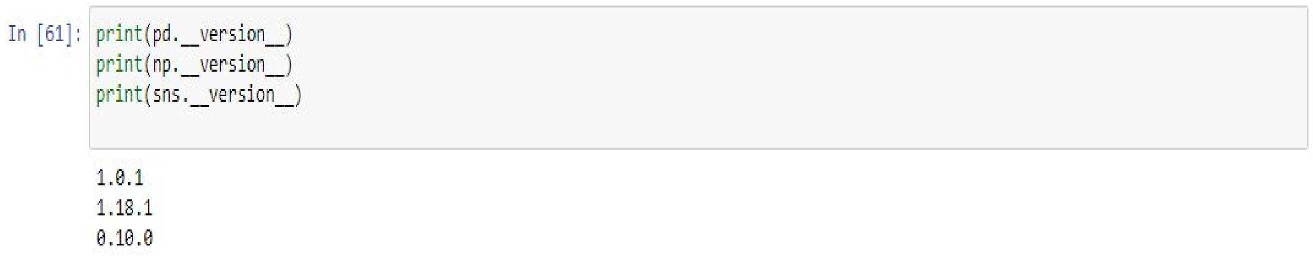
CHAPTER4
CASE STUDY

4.1 Project requirements-

4.1.1 Packages used

- -numpy
- -pandas
- -matplotlib
- -seaborn
- -sklearn

4.1.2 Versions of packages used



```
In [61]: print(pd.__version__)
         print(np.__version__)
         print(sns.__version__)

1.0.1
1.18.1
0.10.0
```

4.1.3 Algorithms used

- -Logistic Regression
- -K Neighbors Classifier
- -RandomForestClassifier
- -GausianNaiveBayesClassifier

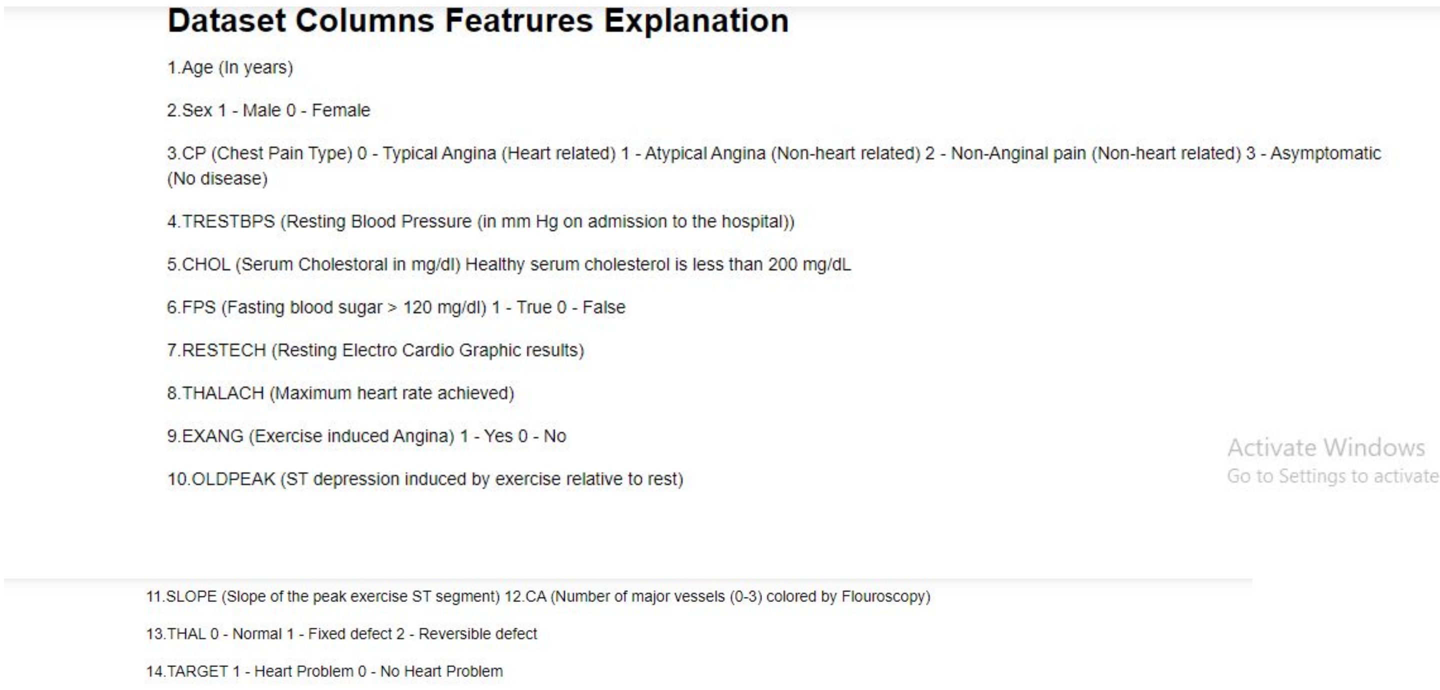
4.2 Problem Statement

Prediction of heart disease occurrence among different people based on a specific dataset

4.3 Dataset Description

Explanation of data set columns features:-

Image 4.3



Dataset Columns Featrures Explanation

1.Age (In years)

2.Sex 1 - Male 0 - Female

3.CP (Chest Pain Type) 0 - Typical Angina (Heart related) 1 - Atypical Angina (Non-heart related) 2 - Non-Anginal pain (Non-heart related) 3 - Asymptomatic (No disease)

4.TRESTBPS (Resting Blood Pressure (in mm Hg on admission to the hospital))

5.CHOL (Serum Cholestoral in mg/dl) Healthy serum cholesterol is less than 200 mg/dL

6.FPS (Fasting blood sugar > 120 mg/dl) 1 - True 0 - False

7.RESTECH (Resting Electro Cardio Graphic results)

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)

11.SLOPE (Slope of the peak exercise ST segment) 12.CA (Number of major vessels (0-3) colored by Flouroscopy)

13.THAL 0 - Normal 1 - Fixed defect 2 - Reversible defect

14.TARGET 1 - Heart Problem 0 - No Heart Problem

4.4 Objective of case study

Based on various factors affecting human heart like serum cholesterol levels, resting blood pressure levels,ThalasSemia and physical factors like age,gender etc we predict the occurrence of heart diseases among people.
To apply machine learning algorithms on dataset to train, test the model and to check for results of various models with its metrics.

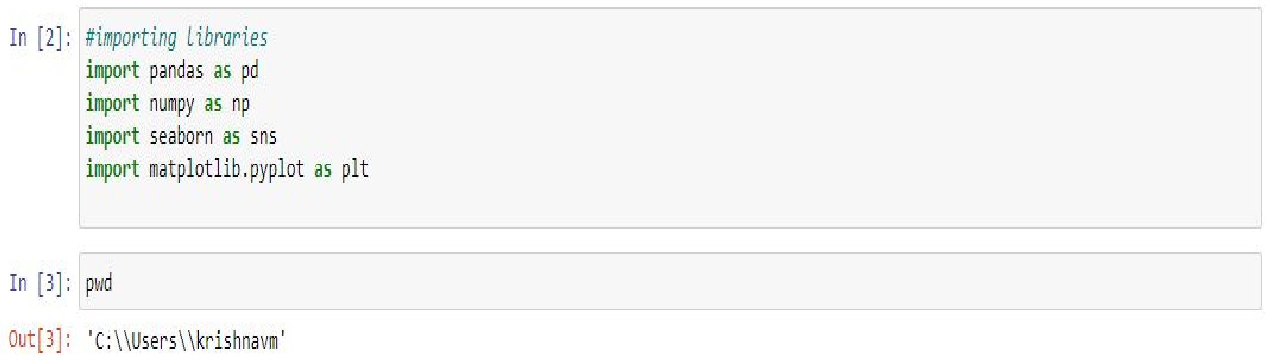
CHAPTER5
MODEL BUILDING

5. Data Preprocessing :-

5.1- importing different libraries required for performing operations on dataset

-checking the path of the file

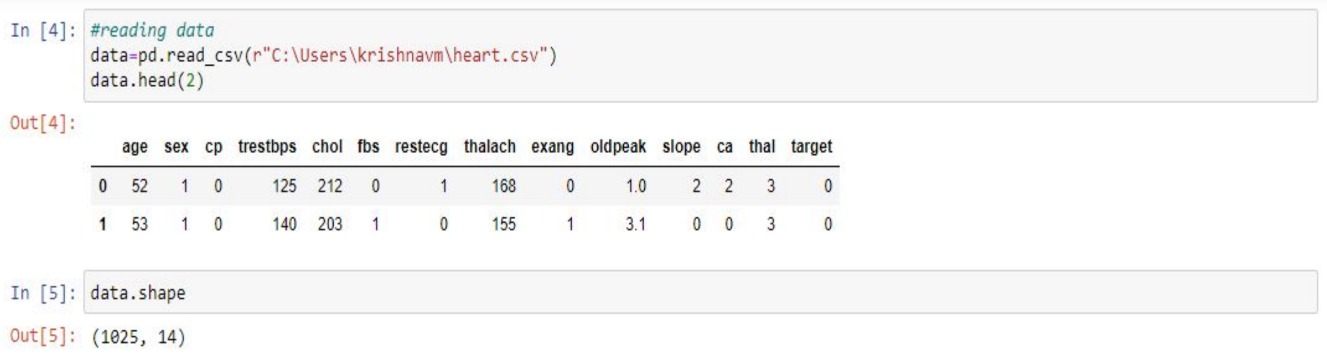
Image 5.1:-



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

In [3]: pwd

Out[3]: 'C:\\Users\\krishnavm'
```

5.2- Reading the data set and checking its shape

Image 5.2:-



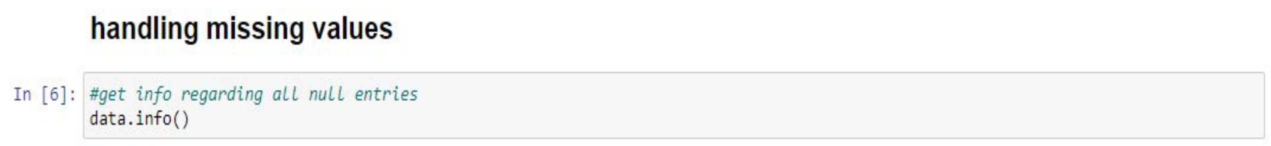
```
In [4]: #reading data
        data=pd.read_csv(r"C:\Users\krishnavm\heart.csv")
        data.head(2)

Out[4]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0

```
In [5]: data.shape

Out[5]: (1025, 14)
```

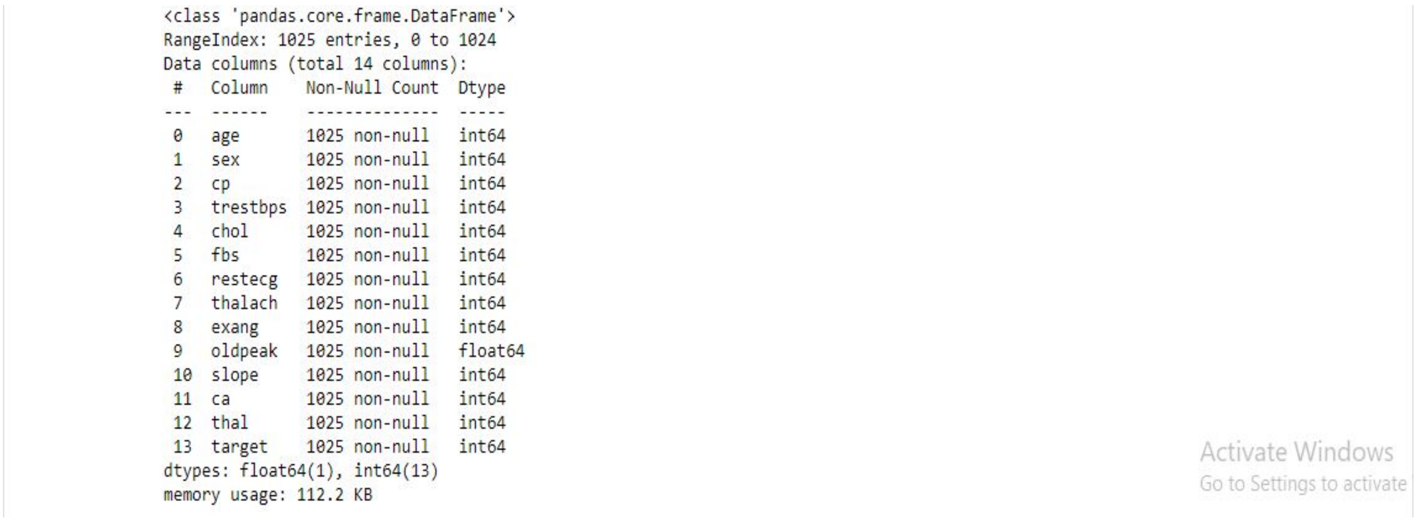
5.3- Handling missing values

Image5.3.1:-



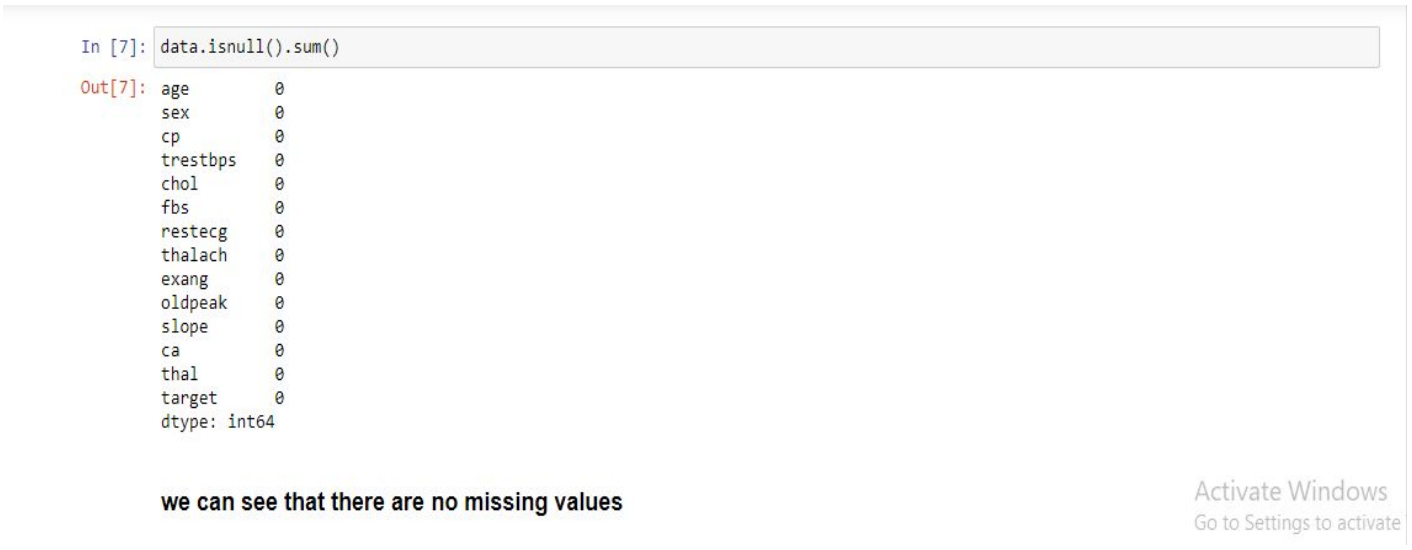
handling missing values

```
In [6]: #get info regarding all null entries
        data.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

Image 5.3.2:-



```
In [7]: data.isnull().sum()

Out[7]: 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
```

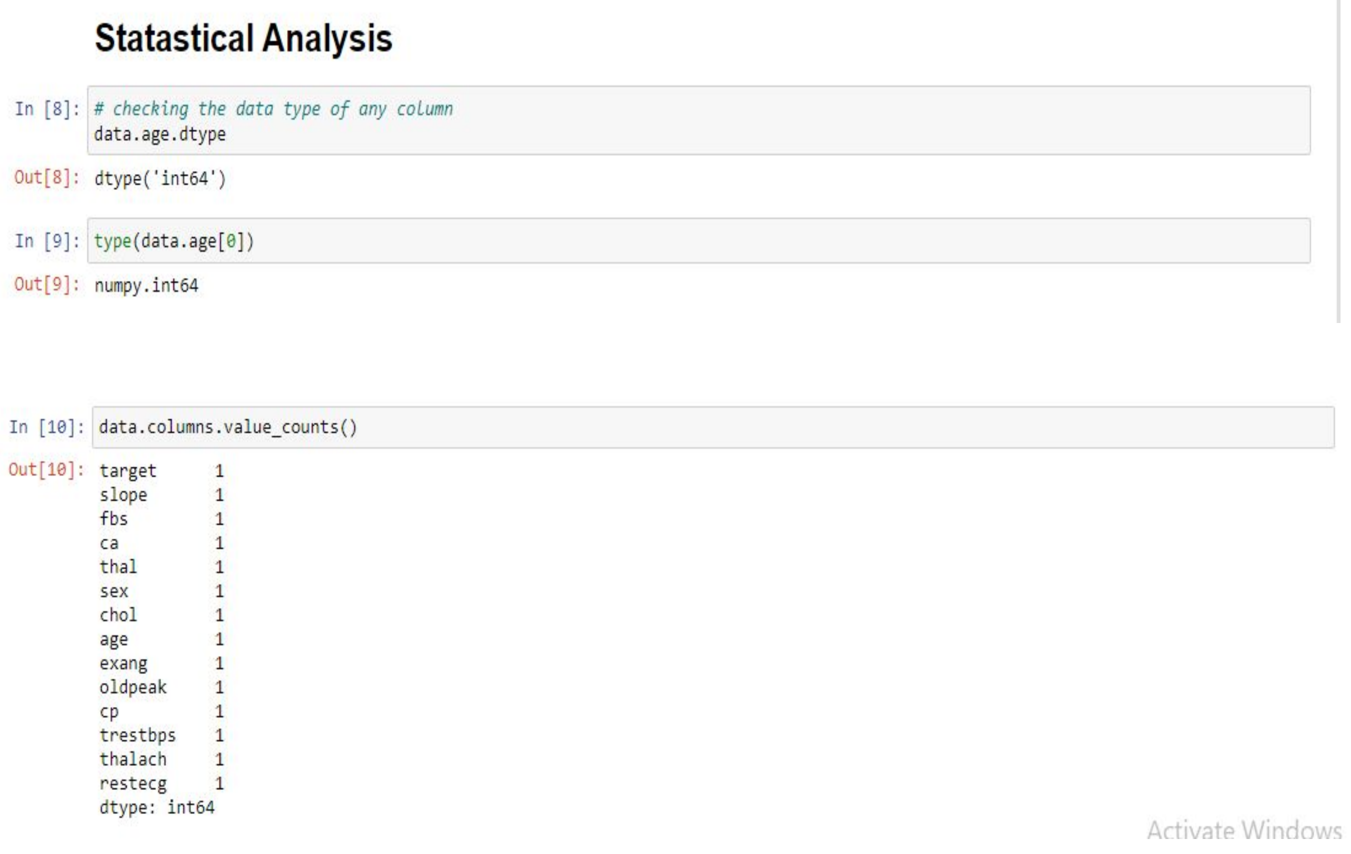
we can see that there are no missing values

Observation- from these above performed operations to handle missing values we can see that there are no missing values present in the data set

5.4- Statistical Analysis:-

Now we have performed some basic statistical operations on the data set like finding the data type of selected columns, using different methods like value_counts etc as shown in the figure below.

Image 5.4.1:-



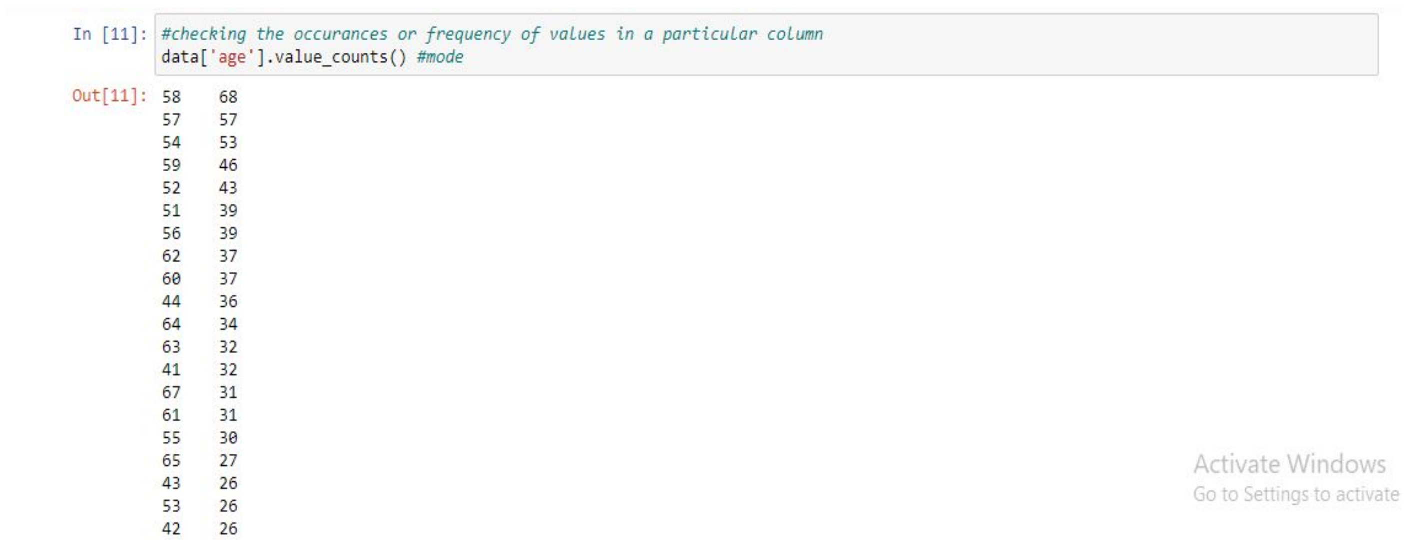
Statastical Analysis

```
In [8]: # checking the data type of any column
        data.age.dtype

Out[8]: dtype('int64')

In [9]: type(data.age[0])

Out[9]: numpy.int64
```

```
In [10]: data.columns.value_counts()

Out[10]: target      1
         slope       1
         fbs         1
         ca          1
         thal        1
         sex         1
         chol        1
         age         1
         exang       1
         oldpeak     1
         cp          1
         trestbps    1
         thalach     1
         restecg     1
         dtype: int64
```

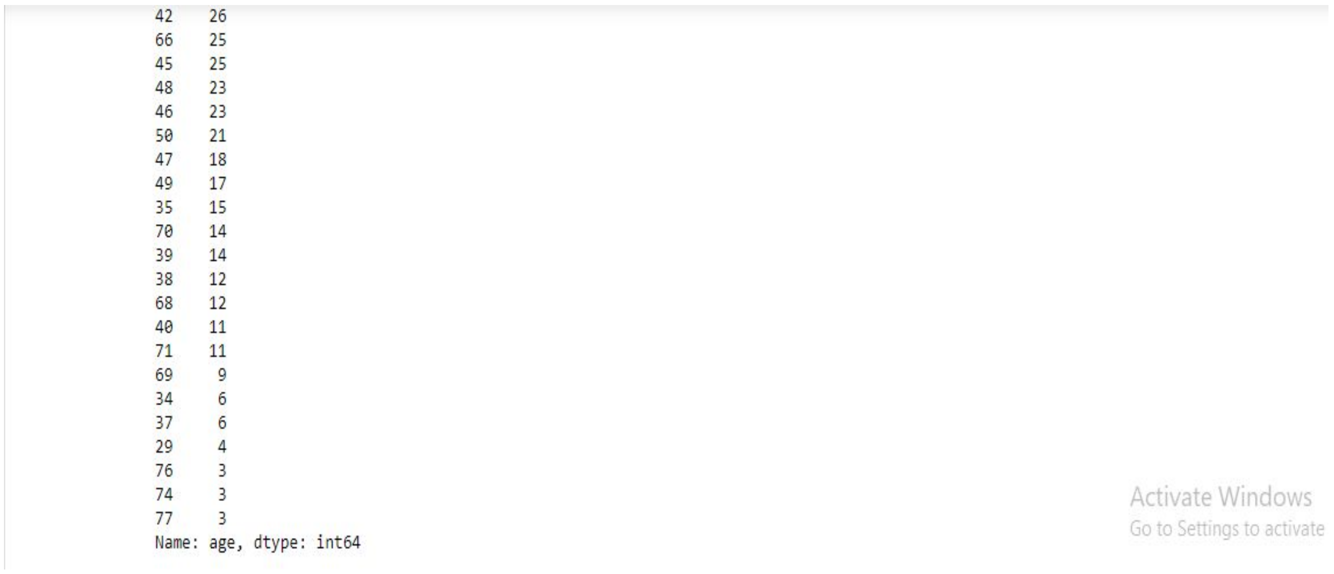
Activate Windows

Image 5.4.2:- here we are checking for frequency of values in selected columns



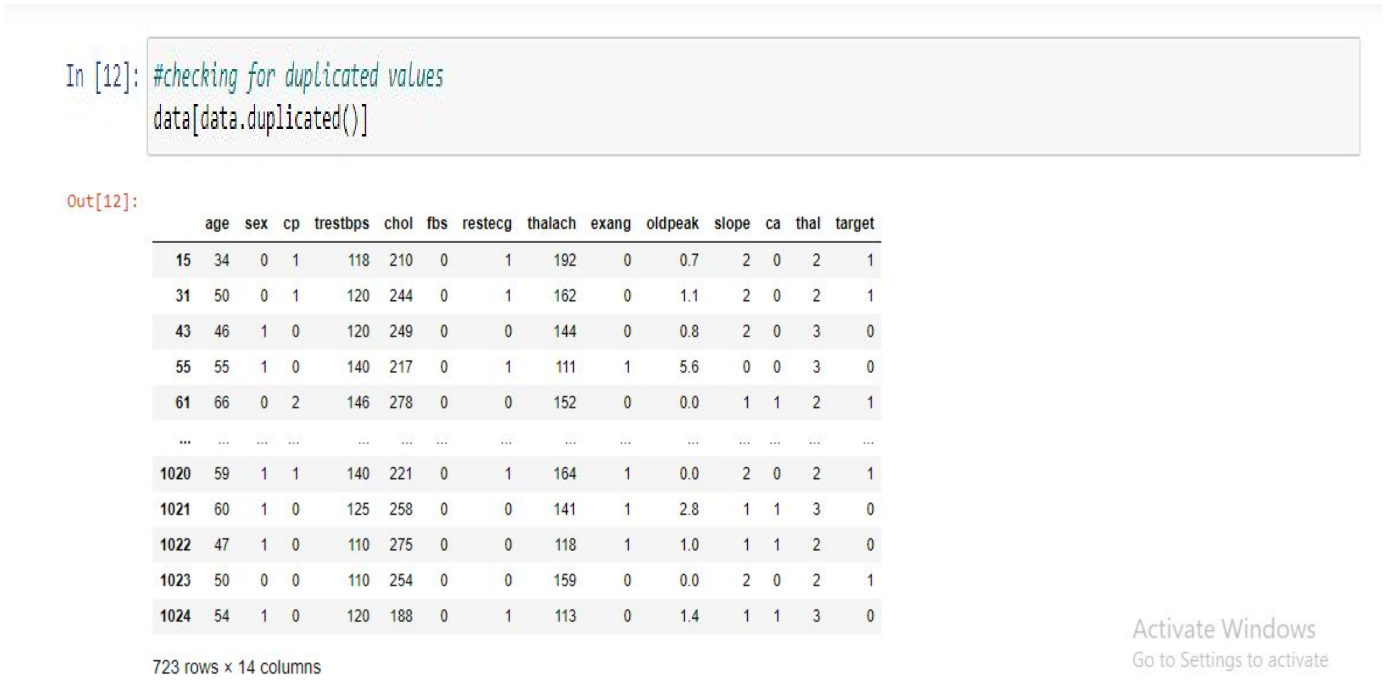
```
In [11]: #checking the occurances or frequency of values in a particular column
         data['age'].value_counts() #mode

Out[11]: 58    68
         57    57
         54    53
         59    46
         52    43
         51    39
         56    39
         62    37
         60    37
         44    36
         64    34
         63    32
         41    32
         67    31
         61    31
         55    30
         65    27
         43    26
         53    26
         42    26
```

Activate Windows
Go to Settings to activate



```
42    26
66    25
45    25
48    23
46    23
50    21
47    18
49    17
35    15
70    14
39    14
38    12
68    12
40    11
71    11
69     9
34     6
37     6
29     4
76     3
74     3
77     3
Name: age, dtype: int64
```

5.4.3- here we are checking for duplicated values in dataset
Image 5.4.3-



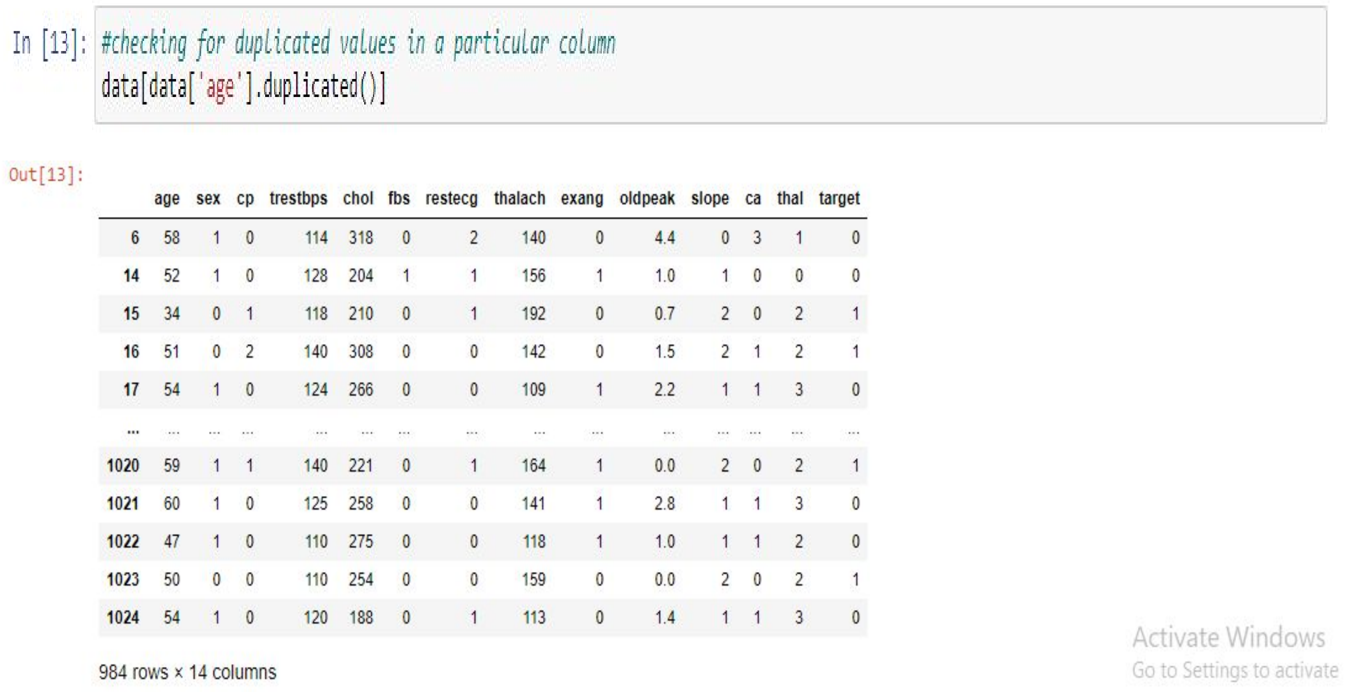
```
In [12]: #checking for duplicated values
         data[data.duplicated()]
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
15	34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
31	50	0	1	120	244	0	1	162	0	1.1	2	0	2	1
43	46	1	0	120	249	0	0	144	0	0.8	2	0	3	0
55	55	1	0	140	217	0	1	111	1	5.6	0	0	3	0
61	66	0	2	146	278	0	0	152	0	0.0	1	1	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

723 rows × 14 columns

Image 5.4.3.1:-



In [13]:
```
#checking for duplicated values in a particular column
data[data['age'].duplicated()]
```

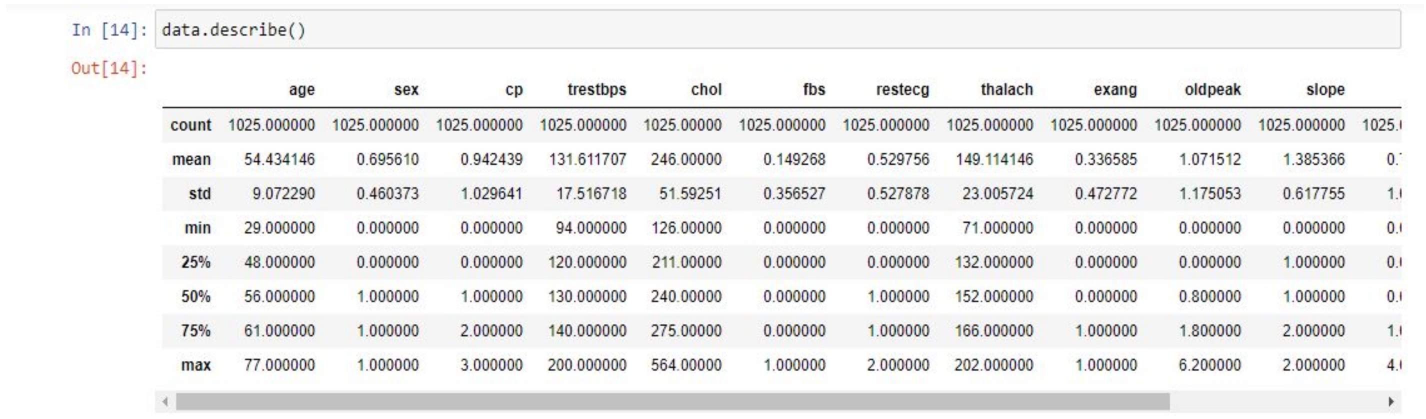
Out[13]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
6	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
14	52	1	0	128	204	1	1	156	1	1.0	1	0	0	0
15	34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
16	51	0	2	140	308	0	0	142	0	1.5	2	1	2	1
17	54	1	0	124	266	0	0	109	1	2.2	1	1	3	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

984 rows × 14 columns

Activate Windows
Go to Settings to activate

5.4.4- different functions for further statistical analysis
Image 5.4.4:-
Using describe function and other functions like mean ,mode etc to analyse the statistical information of data in the below images

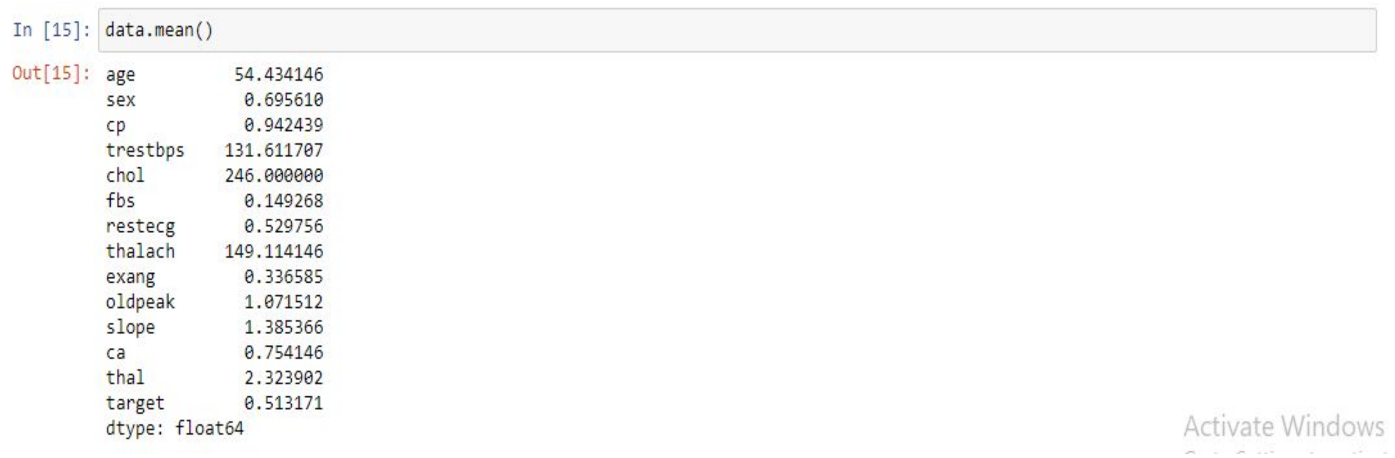


In [14]:
```
data.describe()
```

Out[14]:

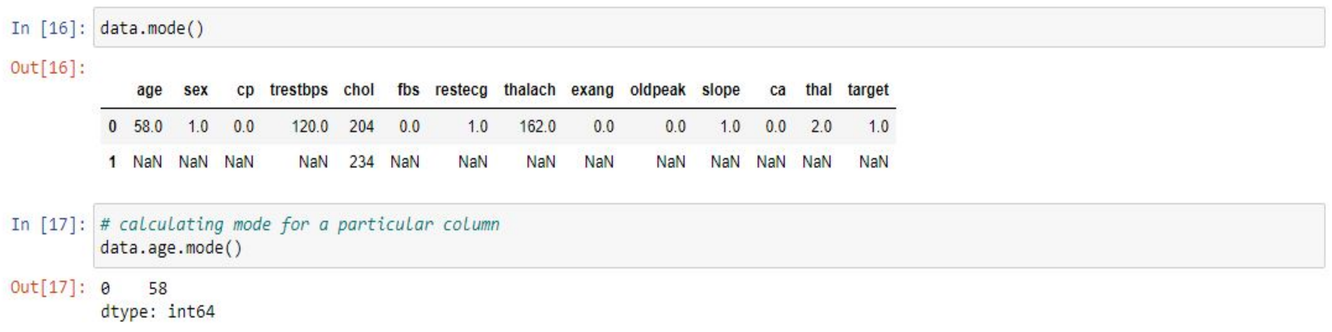
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.(
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512	1.385366	0.⁻
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053	0.617755	1.(
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.(
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000	1.000000	0.(
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.(
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000	2.000000	1.(
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.(

Image5.4.4.1- finding mean of different columns



```
In [15]: data.mean()

Out[15]: age          54.434146
         sex           0.695610
         cp            0.942439
         trestbps    131.611707
         chol        246.000000
         fbs           0.149268
         restecg       0.529756
         thalach     149.114146
         exang         0.336585
         oldpeak       1.071512
         slope         1.385366
         ca            0.754146
         thal          2.323902
         target        0.513171
         dtype: float64
```

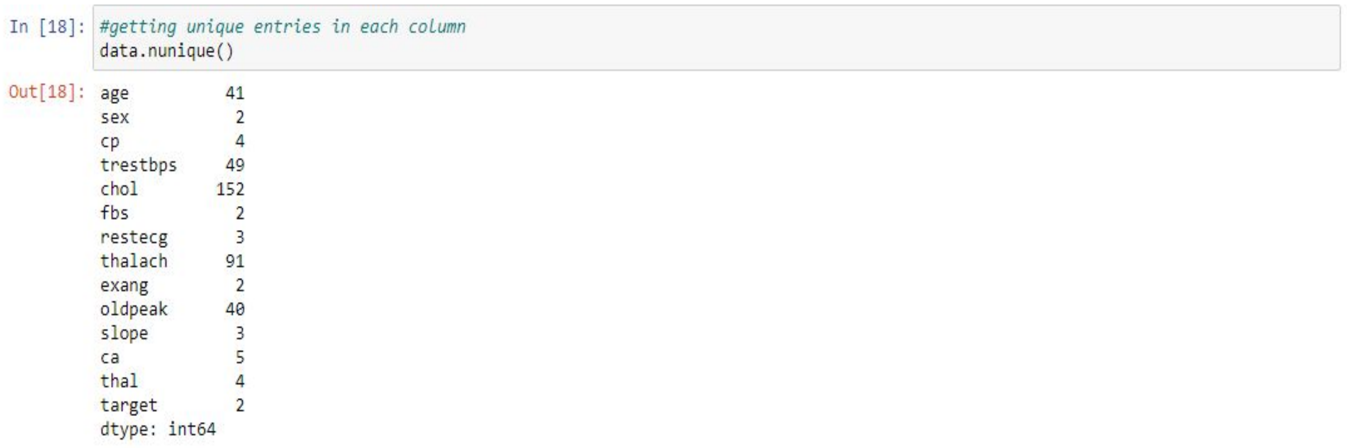
Image 5.4.4.2- finding mode



```
In [16]: data.mode()

Out[16]:
     age  sex   cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope   ca  thal  target
  0  58.0  1.0  0.0   120.0   204   0.0    1.0     162.0    0.0     0.0     1.0   0.0  2.0    1.0
  1  NaN  NaN  NaN     NaN    234   NaN    NaN      NaN     NaN     NaN     NaN   NaN  NaN    NaN

In [17]: # calculating mode for a particular column
         data.age.mode()

Out[17]: 0    58
         dtype: int64
```

5.4.5-

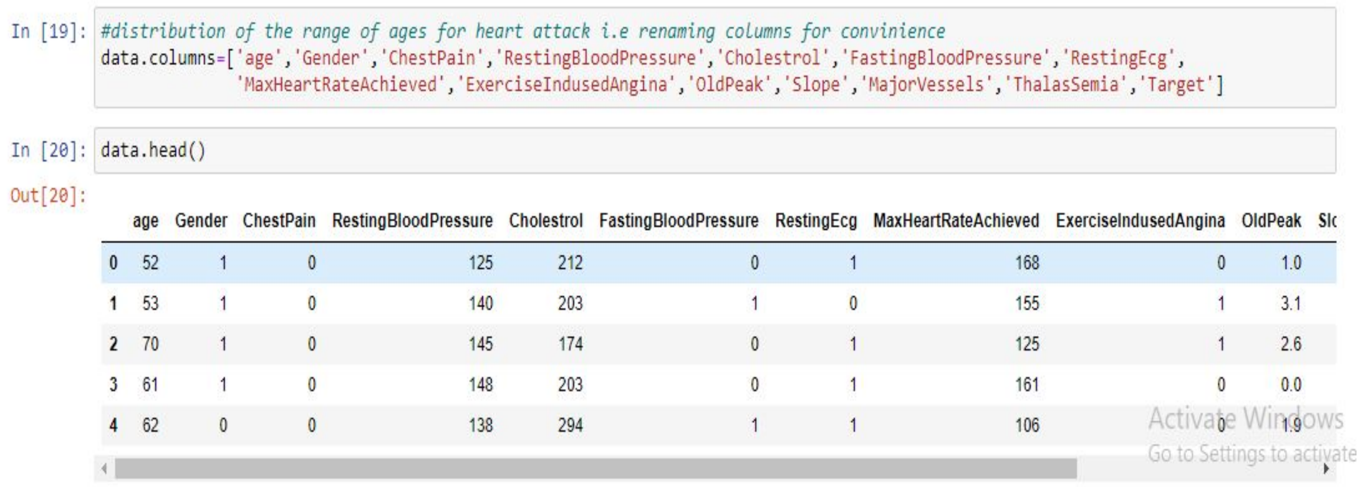
Image 5.4.5- analysing unique entries in column as shown in below image



```
In [18]: #getting unique entries in each column
         data.nunique()

Out[18]: age          41
         sex           2
         cp            4
         trestbps     49
         chol        152
         fbs           2
         restecg       3
         thalach      91
         exang         2
         oldpeak      40
         slope         3
         ca            5
         thal          4
         target        2
         dtype: int64
```

5.5-Renaming

Image 5.5- as a part of preprocessing and feature engineering we are renaming the columns to work conveniently with the dataset further



```
In [19]: #distribution of the range of ages for heart attack i.e renaming columns for convinience
         data.columns=['age','Gender','ChestPain','RestingBloodPressure','Cholestrol','FastingBloodPressure','RestingEcg',
                       'MaxHeartRateAchieved','ExerciseIndusedAngina','OldPeak','Slope','MajorVessels','ThalasSemia','Target']
```

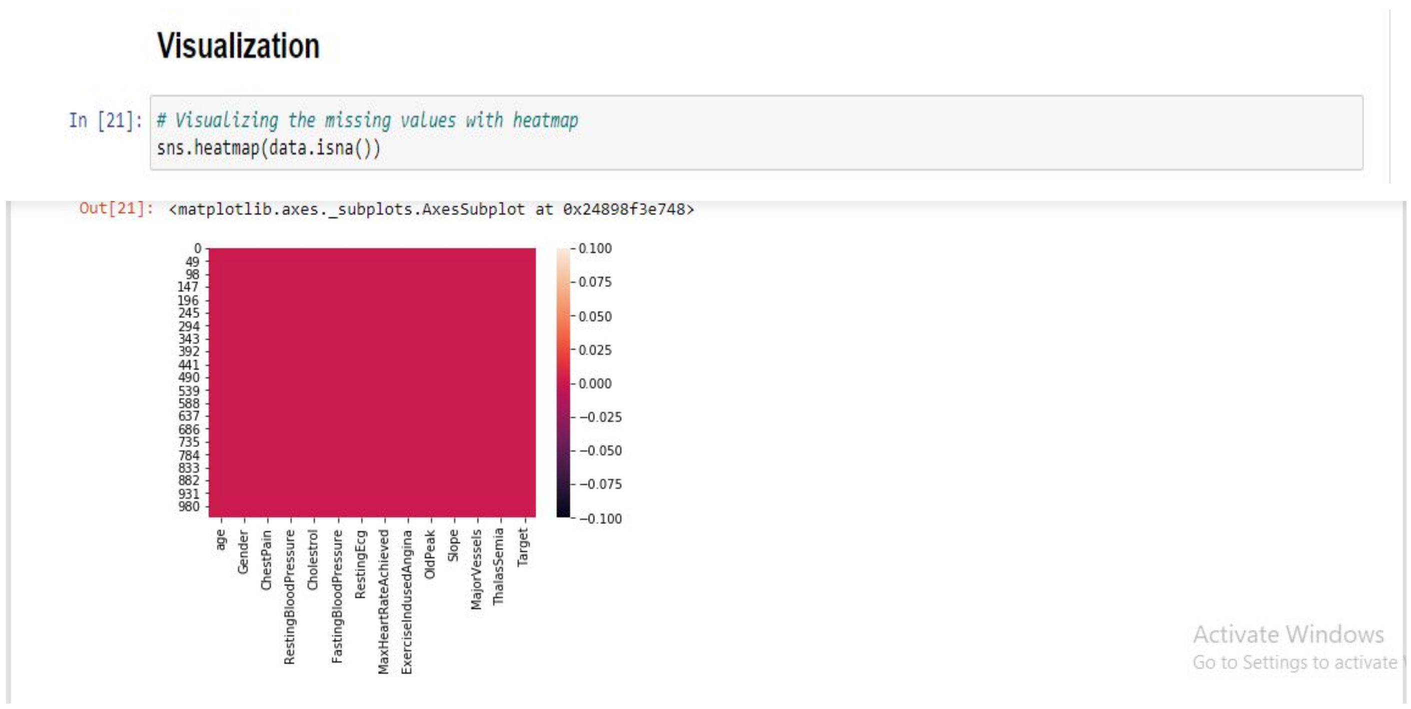
```
In [20]: data.head()
```

Out[20]:

	age	Gender	ChestPain	RestingBloodPressure	Cholestrol	FastingBloodPressure	RestingEcg	MaxHeartRateAchieved	ExerciseIndusedAngina	OldPeak	Slo
0	52	1	0	125	212	0	1	168	0	1.0	
1	53	1	0	140	203	1	0	155	1	3.1	
2	70	1	0	145	174	0	1	125	1	2.6	
3	61	1	0	148	203	0	1	161	0	0.0	
4	62	0	0	138	294	1	1	106	0	1.9	

5.6- Visualization of data
Here we will be using different graphs and techniques to visually represent analysis of our data set.

5.6.1- visualisation using heatmap

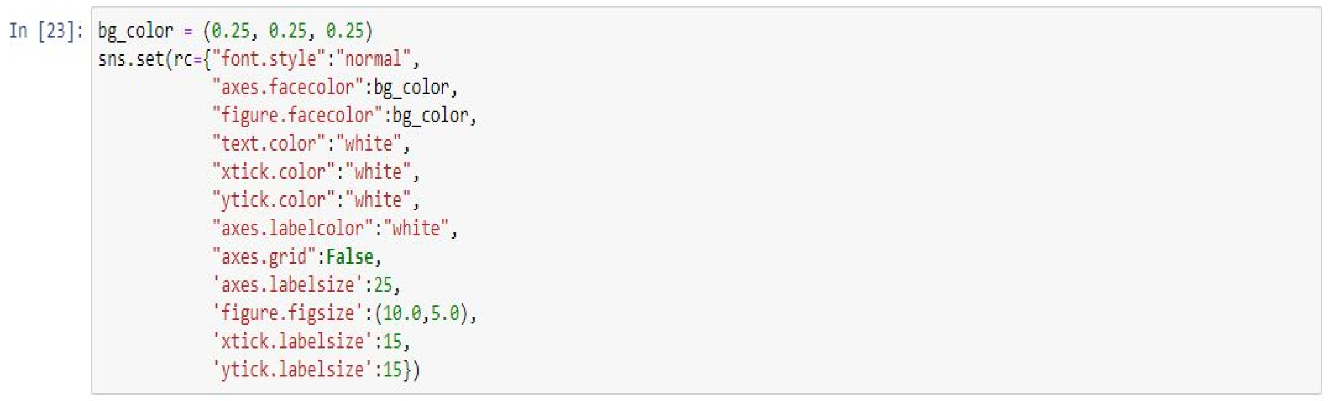


Visualization

```
In [21]: # Visualizing the missing values with heatmap
         sns.heatmap(data.isna())
```

```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x24898f3e748>
```

5.6.2-setting background color and other features for graphs
Image 5.6.2-

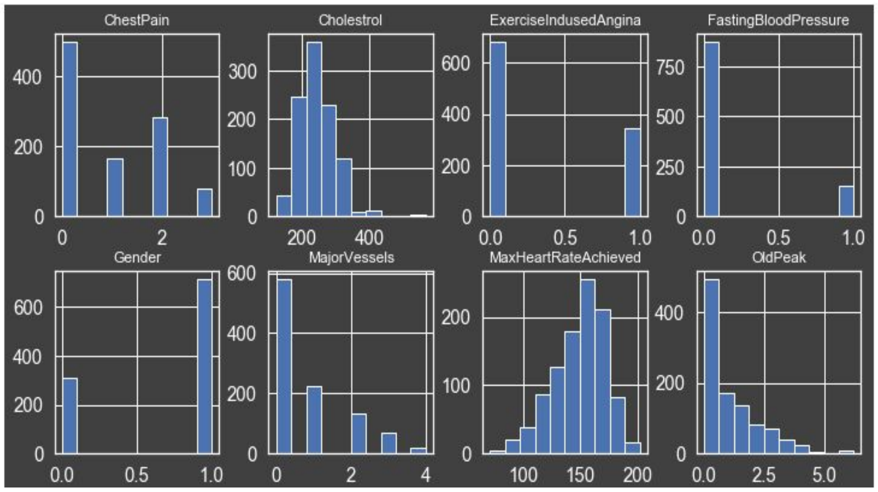


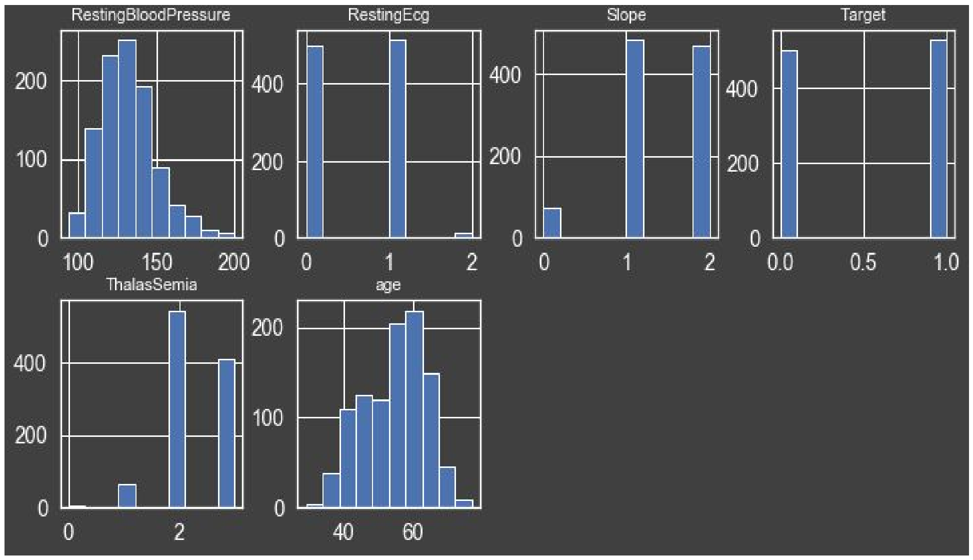
```
In [23]: bg_color = (0.25, 0.25, 0.25)
         sns.set(rc={"font.style":"normal",
                     "axes.facecolor":bg_color,
                     "figure.facecolor":bg_color,
                     "text.color":"white",
                     "xtick.color":"white",
                     "ytick.color":"white",
                     "axes.labelcolor":"white",
                     "axes.grid":False,
                     'axes.labelsize':25,
                     'figure.figsize':(10.0,5.0),
                     'xtick.labelsize':15,
                     'ytick.labelsize':15})
```

Image 5.6.3- plotting a histogram for different columns present in the data set

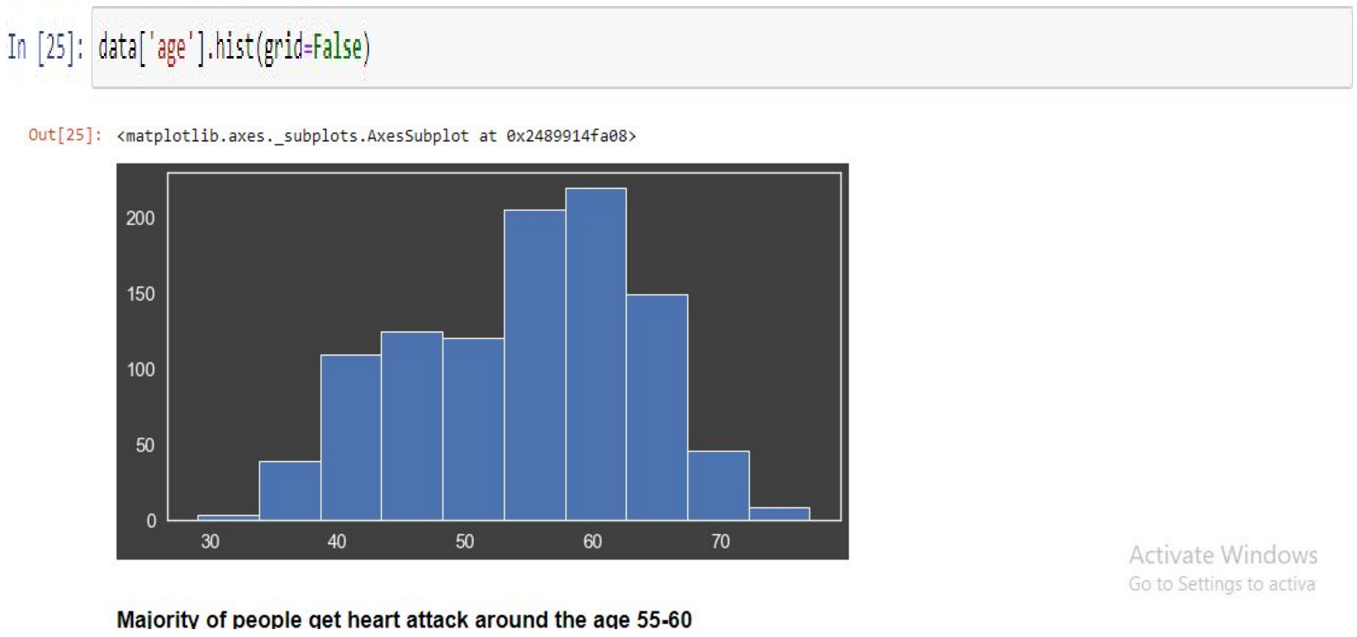
```
In [24]: data.hist(figsize=(12,12))
```

```
Out[24]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000024898F14C48>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024899176688>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x00000248991ACC88>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x00000248991E5748>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x000002489921E848>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024899255908>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x000002489928CA48>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x00000248992C5B88>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x00000248992D2788>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024899308948>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x000002489936EF08>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x00000248993A9F08>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x00000248993E5048>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x000002489941E188>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x0000024899457288>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x000002489948F448>]],
               dtype=object)
```





5.6.4-Plotting histogram for the age column
Image 5.6.4-

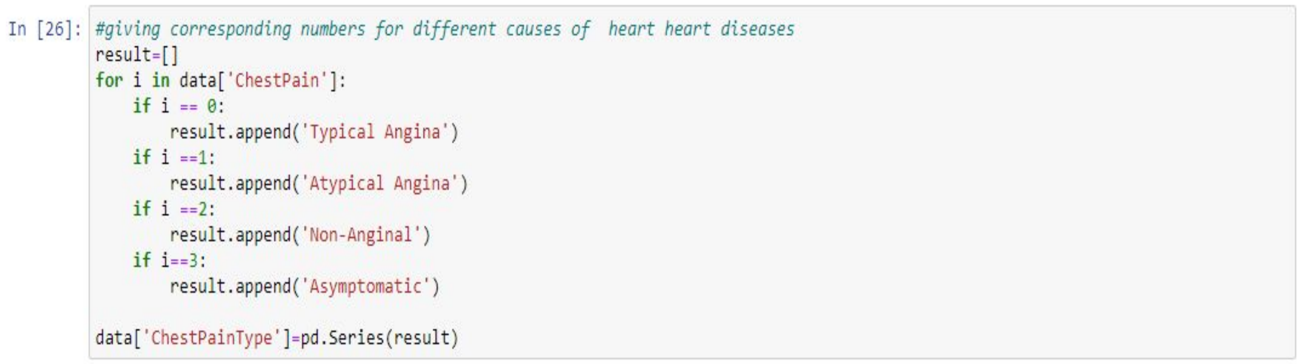


```
In [25]: data['age'].hist(grid=False)

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x2489914fa08>
```

Majority of people get heart attack around the age 55-60

Observation-From this above histogram we can observe that majority of people who belong to the age group of 55-60 are often prone to heart diseases

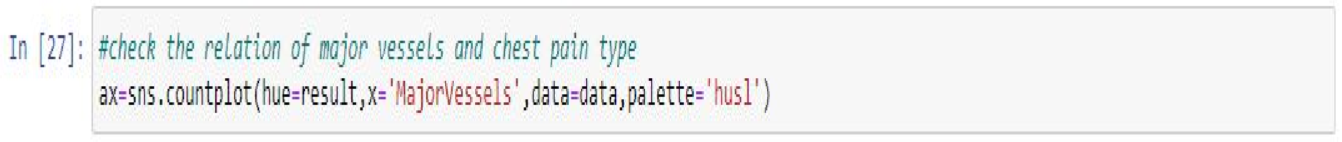
5.6.5-here we are are building a code inorder to number the different kind of chest pains as shown in the below image
Image 5.6.5-



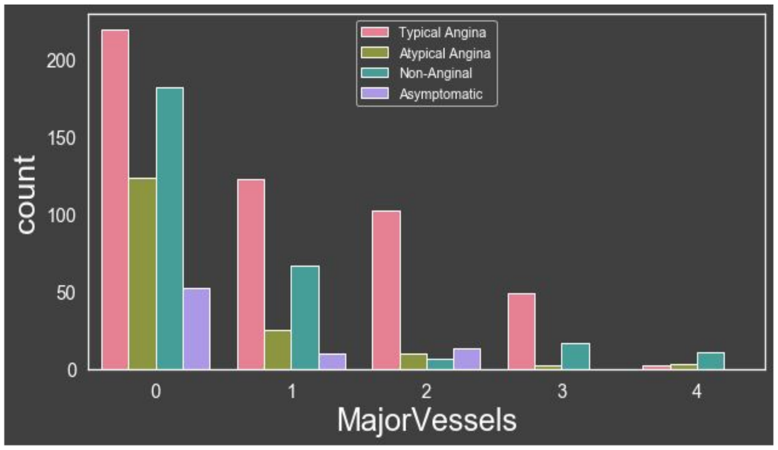
```
In [26]: #giving corresponding numbers for different causes of  heart heart diseases
         result=[]
         for i in data['ChestPain']:
             if i == 0:
                 result.append('Typical Angina')
             if i ==1:
                 result.append('Atypical Angina')
             if i ==2:
                 result.append('Non-Anginal')
             if i==3:
                 result.append('Asymptomatic')

         data['ChestPainType']=pd.Series(result)
```

5.6.6-we are checking the relation between major vessels and chest pain as follows

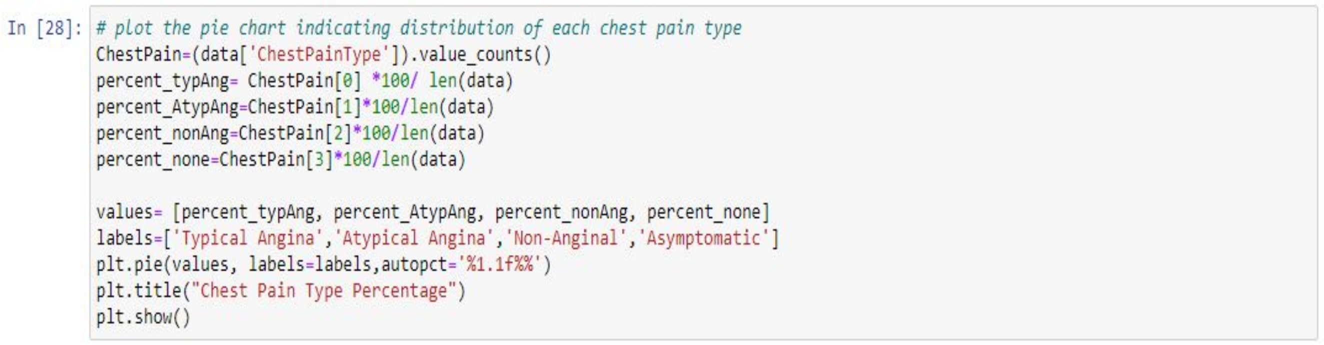
Image 5.6.6-



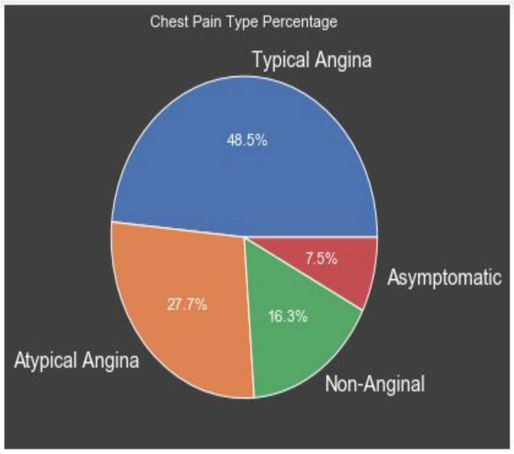
```
In [27]: #check the relation of major vessels and chest pain type
         ax=sns.countplot(hue=result,x='MajorVessels',data=data,palette='husl')
```



5.6.7-Plotting a pie chart to indicate different kinds of chest pain
image5.6.7-



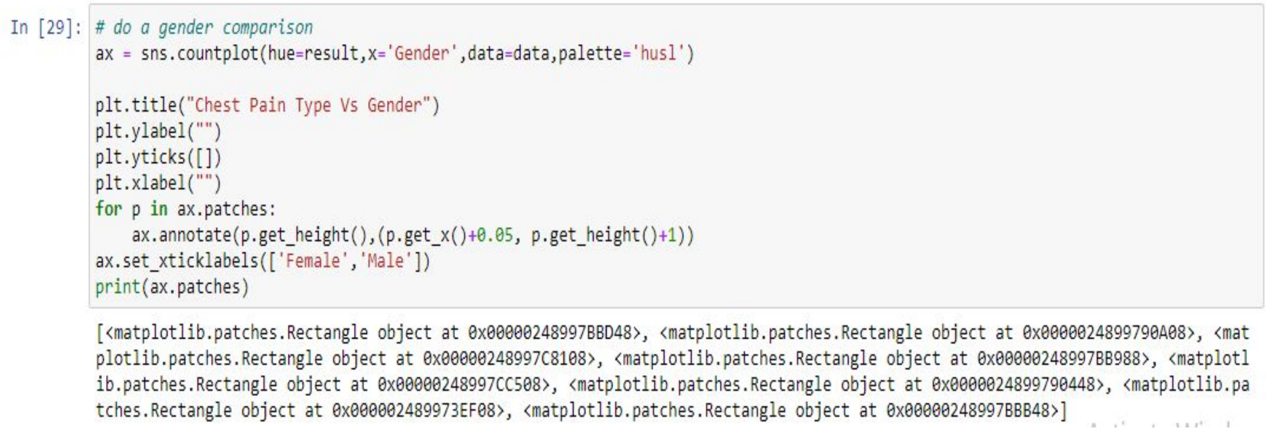
```
In [28]: # plot the pie chart indicating distribution of each chest pain type
         ChestPain=(data['ChestPainType']).value_counts()
         percent_typAng= ChestPain[0] *100/ len(data)
         percent_AtypAng=ChestPain[1]*100/len(data)
         percent_nonAng=ChestPain[2]*100/len(data)
         percent_none=ChestPain[3]*100/len(data)

         values= [percent_typAng, percent_AtypAng, percent_nonAng, percent_none]
         labels=['Typical Angina','Atypical Angina','Non-Anginal','Asymptomatic']
         plt.pie(values, labels=labels,autopct='%1.1f%%')
         plt.title("Chest Pain Type Percentage")
         plt.show()
```



observation- majority of the type of chest pain is observed to be Typical Angina. so from previous obsevation along with this one we can draw a conclusion that most of the people from age 55-60 experience a typical angina chest pain.

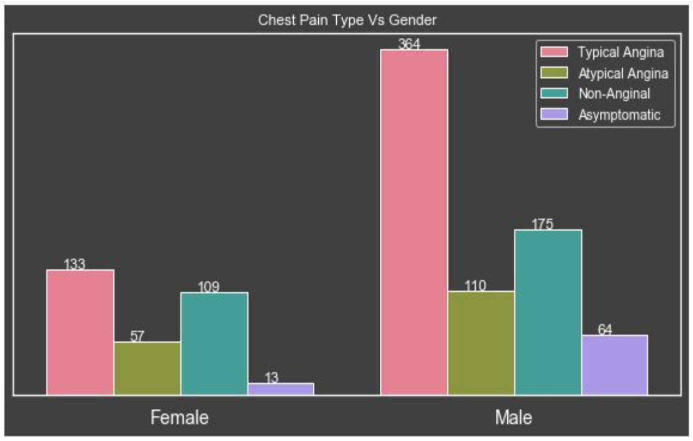
Observation:- we can draw a conclusion that majority of people who are in the age 55-60 are prone to heart diseases from the type of chest pain called typical angina

5.6.8- now we would plot a graph for analysing heart diseases based on gender
Image 5.6.8-



```
In [29]: # do a gender comparison
         ax = sns.countplot(hue=result,x='Gender',data=data,palette='husl')

         plt.title("Chest Pain Type Vs Gender")
         plt.ylabel("")
         plt.yticks([])
         plt.xlabel("")
         for p in ax.patches:
             ax.annotate(p.get_height(),(p.get_x()+0.05, p.get_height()+1))
         ax.set_xticklabels(['Female','Male'])
         print(ax.patches)
```

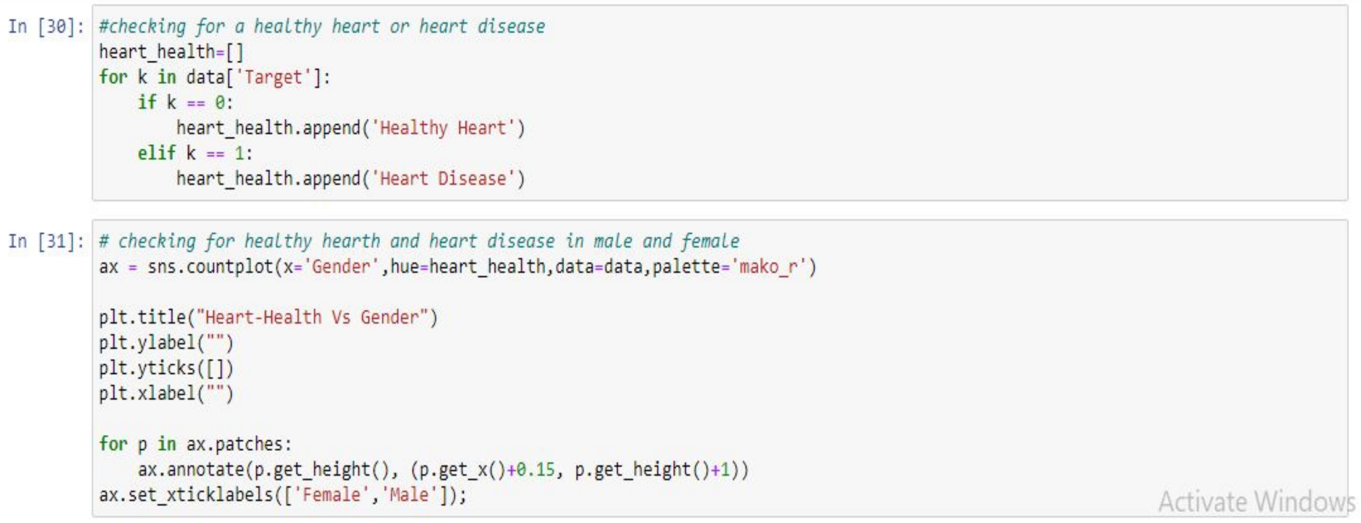
```
[<matplotlib.patches.Rectangle object at 0x00000248997BBD48>, <matplotlib.patches.Rectangle object at 0x0000024899790A08>, <mat
plotlib.patches.Rectangle object at 0x00000248997C8108>, <matplotlib.patches.Rectangle object at 0x00000248997BB988>, <matplotl
ib.patches.Rectangle object at 0x00000248997CC508>, <matplotlib.patches.Rectangle object at 0x0000024899790448>, <matplotlib.pa
tches.Rectangle object at 0x000002489973EF08>, <matplotlib.patches.Rectangle object at 0x00000248997BBB48>]
```



Observation-it can be observed that majority of Typical Angina is experienced by males.

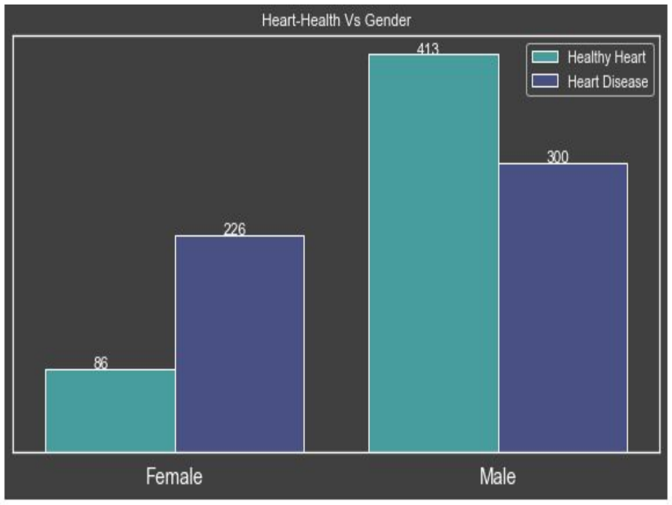
Observation- from the above image it can be seen that chest pain is majorly occurs in males

5.6.9- here we are building a code to categorize healthy heart as 0 and unhealthy heart as 1. And we are visualising this heart health against gender in a countplot
Image 5.6.9-



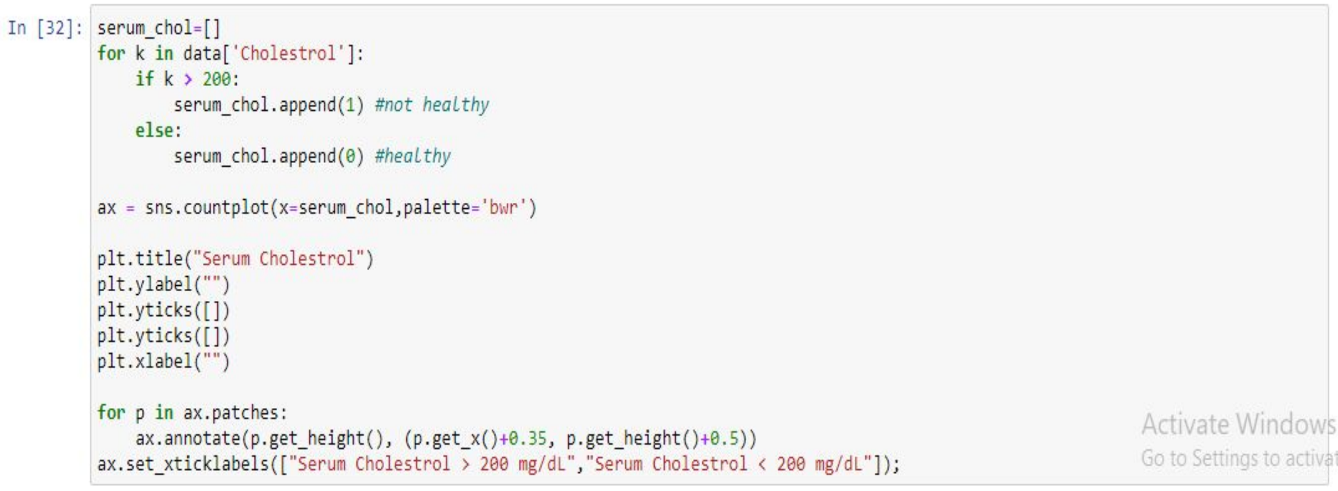
```
In [30]: #checking for a healthy heart or heart disease
         heart_health=[]
         for k in data['Target']:
             if k == 0:
                 heart_health.append('Healthy Heart')
             elif k == 1:
                 heart_health.append('Heart Disease')
```

```
In [31]: # checking for healthy hearth and heart disease in male and female
         ax = sns.countplot(x='Gender',hue=heart_health,data=data,palette='mako_r')

         plt.title("Heart-Health Vs Gender")
         plt.ylabel("")
         plt.yticks([])
         plt.xlabel("")

         for p in ax.patches:
             ax.annotate(p.get_height(), (p.get_x()+0.15, p.get_height()+1))
         ax.set_xticklabels(['Female','Male']);
```

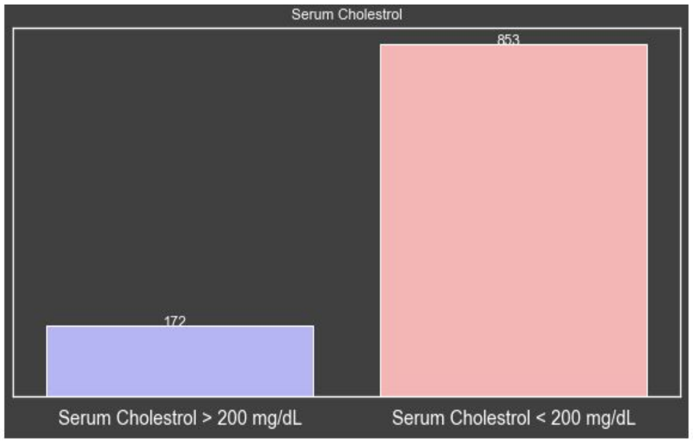


Observation - another conclusion can be drawn that heart diseases occur mostly in males

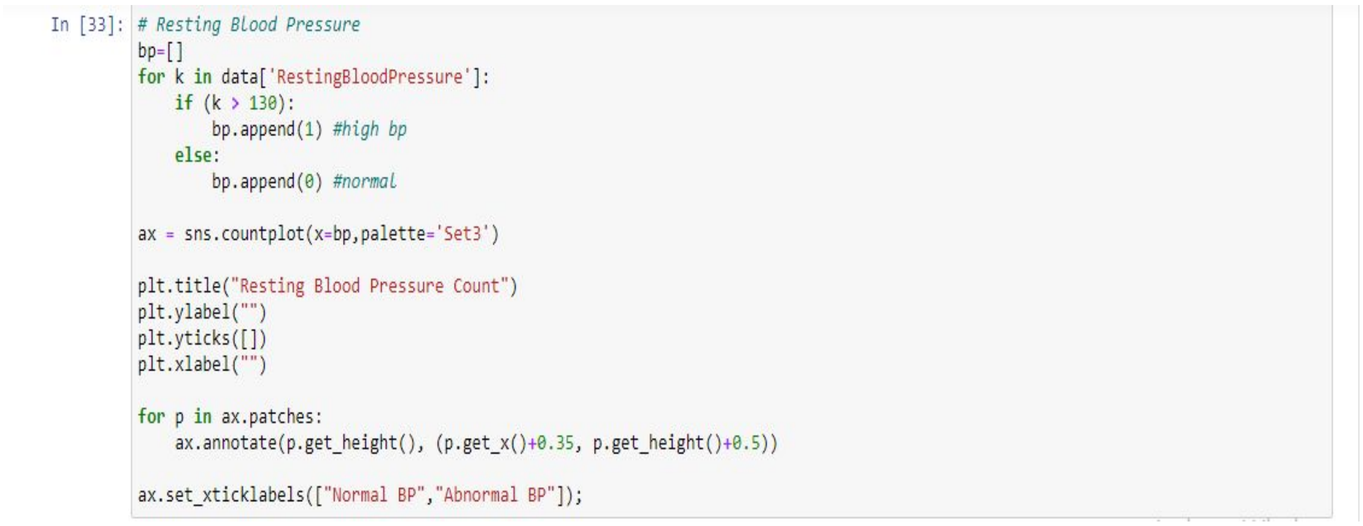
5.6.10- here we are checking the serum cholesterol levels to check if it is a healthy heart or not
image 5.6.10-



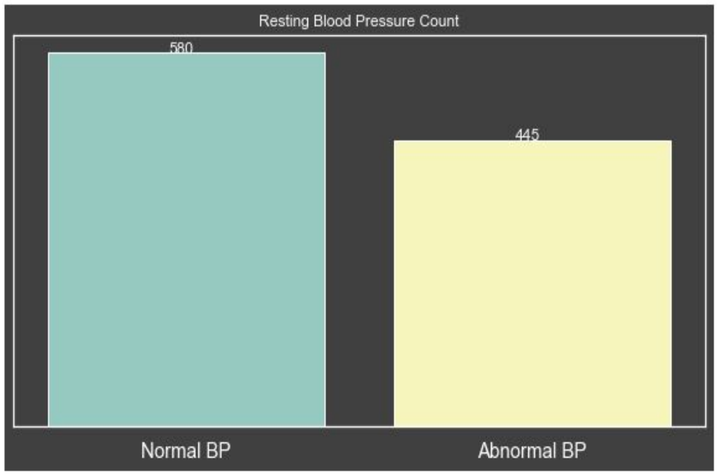
```
In [32]: serum_chol=[]
         for k in data['Cholestrol']:
             if k > 200:
                 serum_chol.append(1) #not healthy
             else:
                 serum_chol.append(0) #healthy

         ax = sns.countplot(x=serum_chol,palette='bwr')

         plt.title("Serum Cholestrol")
         plt.ylabel("")
         plt.yticks([])
         plt.yticks([])
         plt.xlabel("")

         for p in ax.patches:
             ax.annotate(p.get_height(), (p.get_x()+0.35, p.get_height()+0.5))
         ax.set_xticklabels(["Serum Cholestrol > 200 mg/dL","Serum Cholestrol < 200 mg/dL"]);
```



5.6.11-here we are checking for high blood pressure and normal blood pressure based on resting blood pressure levels
Image 5.6.11



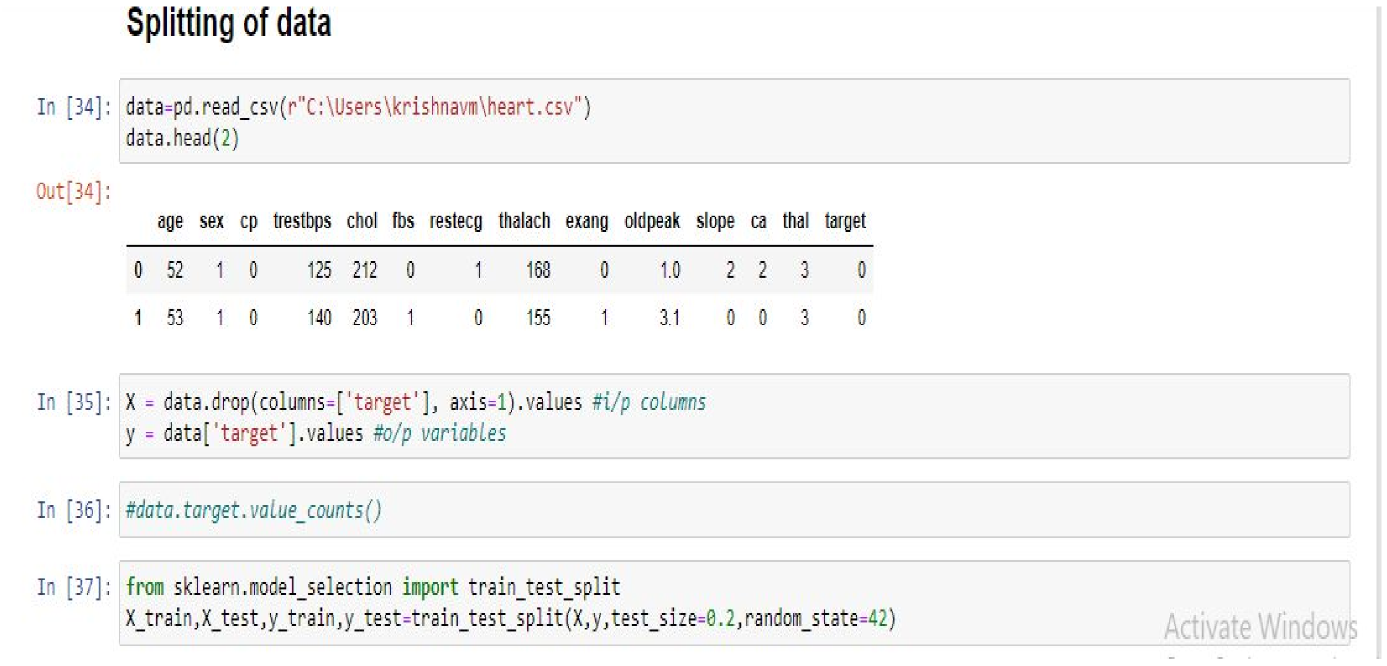
```
In [33]: # Resting Blood Pressure
         bp=[]
         for k in data['RestingBloodPressure']:
             if (k > 130):
                 bp.append(1) #high bp
             else:
                 bp.append(0) #normal

         ax = sns.countplot(x=bp,palette='Set3')

         plt.title("Resting Blood Pressure Count")
         plt.ylabel("")
         plt.yticks([])
         plt.xlabel("")

         for p in ax.patches:
             ax.annotate(p.get_height(), (p.get_x()+0.35, p.get_height()+0.5))

         ax.set_xticklabels(["Normal BP","Abnormal BP"]);
```



6- Feature Selection:-

6.1- as a part of feature selection we will be splitting the data into training data and testing data and perform further operations on it .
Image 6.1-

TRAINING THE MODEL:
Method 1:
● Splitting the data : after the preprocessing is done then the data is split into train and test sets
● In Machine Learning in order to access the performance of the classifier. You train the classifier using 'training set' and then test the performance of your classifier on unseen 'test set'. An important point to note is that during training the classifier only uses the training set . The test set must not be used during training the classifier. The test set will only be available during testing the classifier
● training set - a subset to train a model.(Model learns patterns between Input and Output)
● test set - a subset to test the trained model.(To test whether the model has correctly learnt)
● The amount or percentage of Splitting can be taken as specified (i.e. train data = 75% , test data =25% or train data = 80% , test data= 20%)
● First we need to identify the input and output variables and we need to separate the input set and output set ● In scikit learn library we have a package called model_selection in which train_test_split method is available .we need to import this method
● This method splits the input and output data to train and test based on the percentage specified by the user and assigns them to four different variables(we need to mention the variables)



Splitting of data

```
In [34]: data=pd.read_csv(r"C:\Users\krishnavm\heart.csv")
         data.head(2)
```

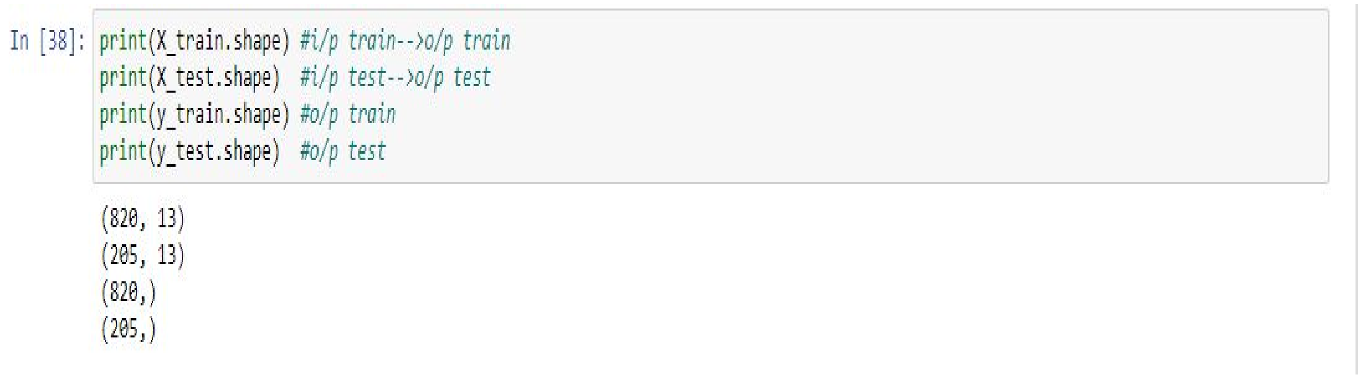
Out[34]:

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  ca  thal  target
0   52    1   0       125   212    0        1      168      0      1.0      2   2     3       0
1   53    1   0       140   203    1        0      155      1      3.1      0   0     3       0
```

```
In [35]: X = data.drop(columns=['target'], axis=1).values #i/p columns
         y = data['target'].values #o/p variables
```

```
In [36]: #data.target.value_counts()
```

```
In [37]: from sklearn.model_selection import train_test_split
         X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

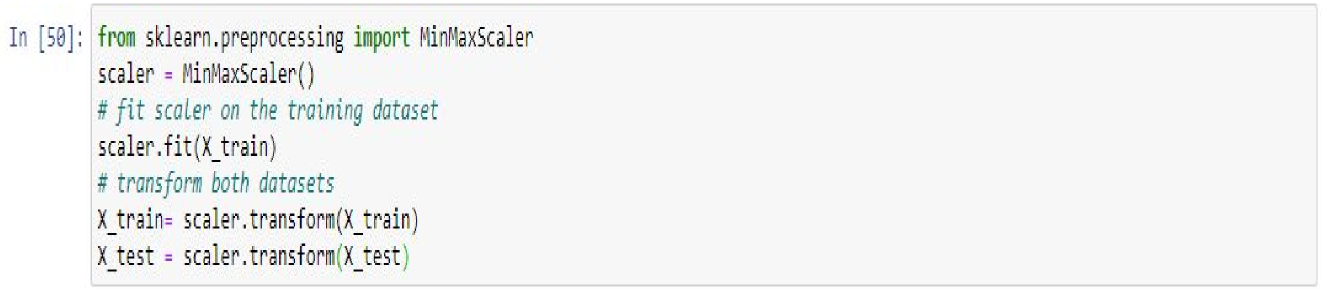


```
In [38]: print(X_train.shape) #i/p train-->o/p train
         print(X_test.shape)  #i/p test-->o/p test
         print(y_train.shape) #o/p train
         print(y_test.shape)  #o/p test

(820, 13)
(205, 13)
(820,)
(205,)
```

Observation- we have split the data into training and testing data and printed the train and test shapes successfully.

6.2- now to obtain better results we perform feature scaling on data so that any inequalities present will be removed .

Image 6.2-



```
In [50]: from sklearn.preprocessing import MinMaxScaler
         scaler = MinMaxScaler()
         # fit scaler on the training dataset
         scaler.fit(X_train)
         # transform both datasets
         X_train= scaler.transform(X_train)
         X_test = scaler.transform(X_test)
```

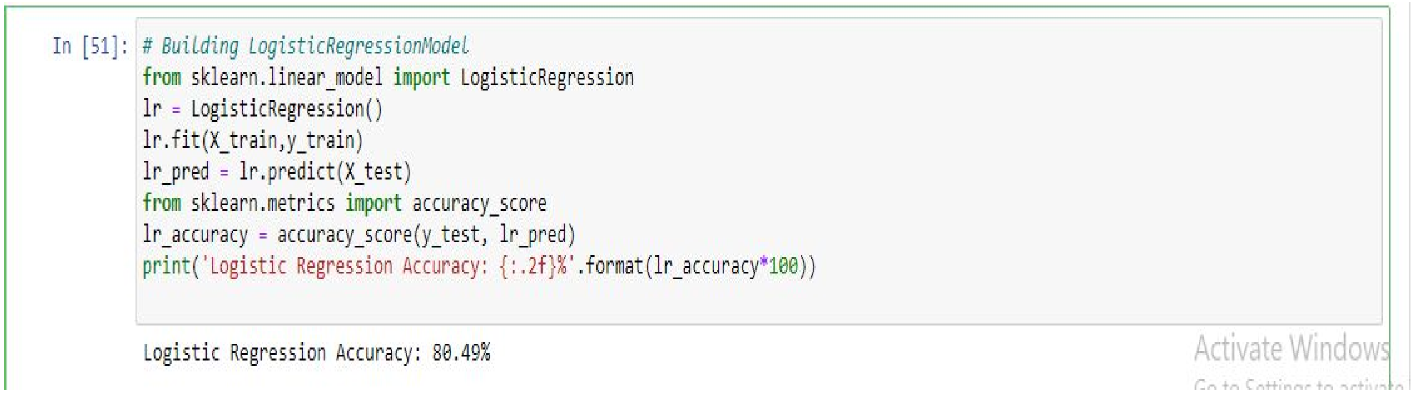
We have successfully performed scaling on the data now further operations can be applied and models can be built on this data.
Scaling will help in obtaining better accuracy results.

7- Building Model and Evaluation

7.1-Logistic Regression model

in the following image we can see that we built the logistic regression model on the data set

Image 7.1-



```
In [51]: # Building LogisticRegressionModel
         from sklearn.linear_model import LogisticRegression
         lr = LogisticRegression()
         lr.fit(X_train,y_train)
         lr_pred = lr.predict(X_test)
         from sklearn.metrics import accuracy_score
         lr_accuracy = accuracy_score(y_test, lr_pred)
         print('Logistic Regression Accuracy: {:.2f}%'.format(lr_accuracy*100))

         Logistic Regression Accuracy: 80.49%
```

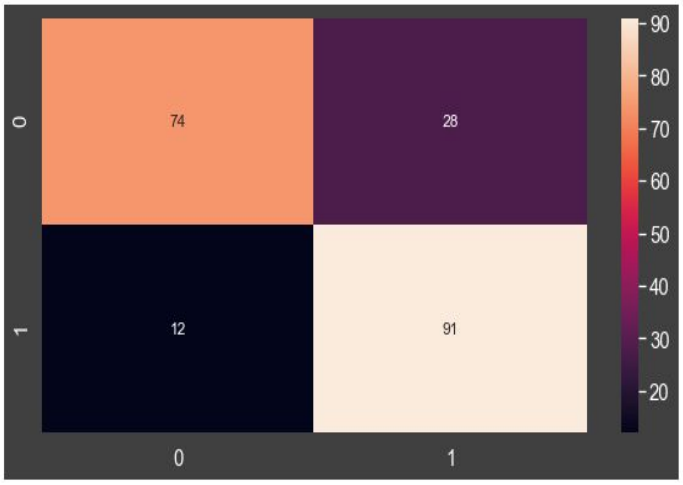
Observation- we have built a logistic regression model and have also performed prediction on the trained model .

We have obtained an accuracy score of 80.49%

7.1.1-confusion matrix for lr model

Image 7.1.1

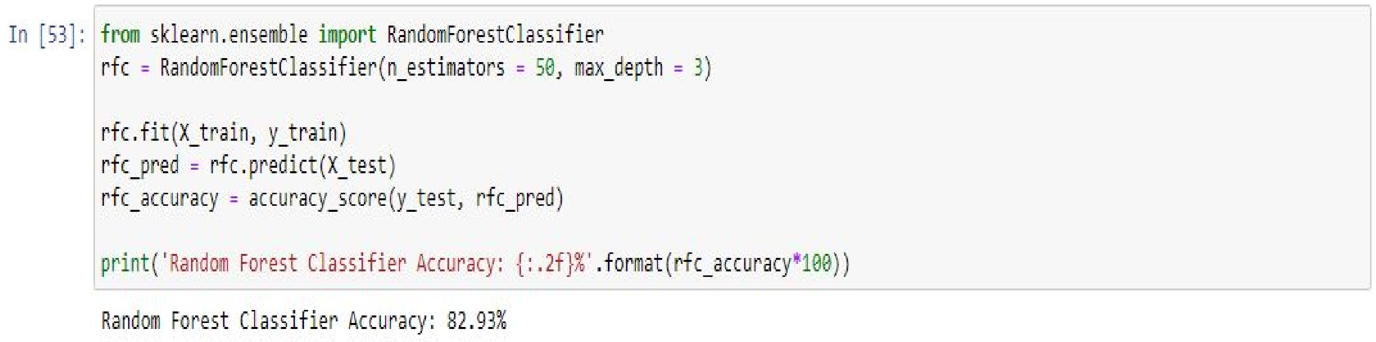
```
In [52]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test,lr_pred)
         sns.heatmap(cm,annot=True);
```



We have plotted the corresponding confusion matrix for the model

7.2:- RandomForest model

Image 7.2- We have built a random forest model as shown below

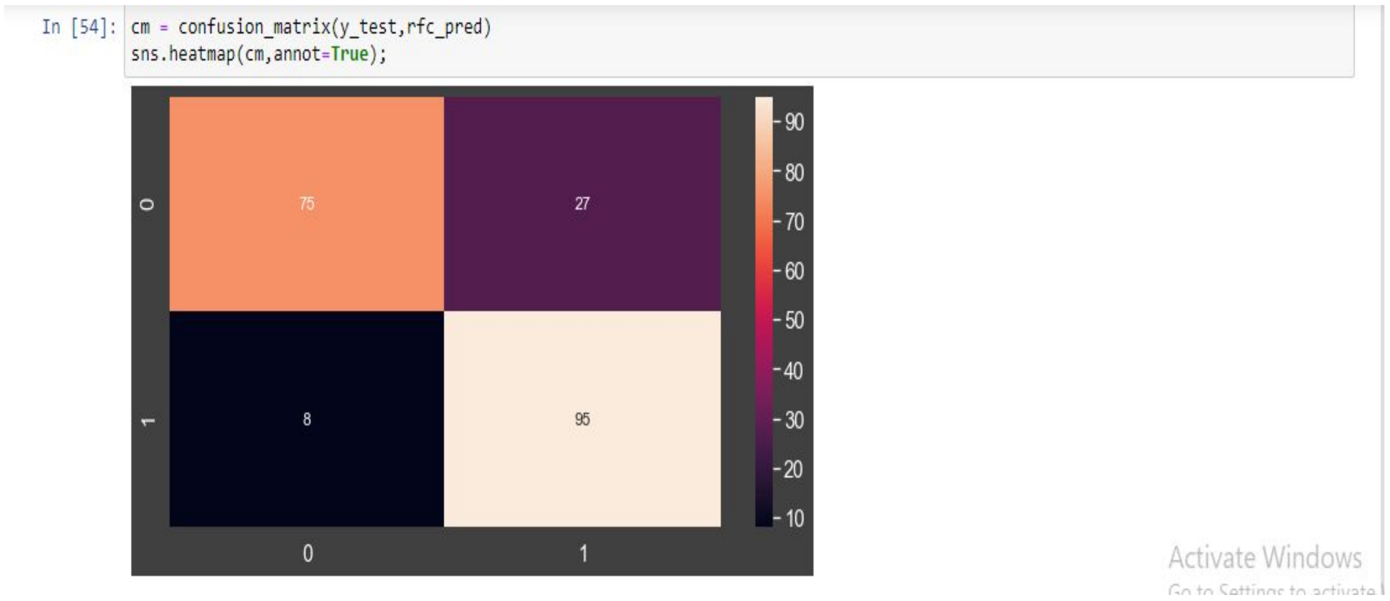


```
In [53]: from sklearn.ensemble import RandomForestClassifier
         rfc = RandomForestClassifier(n_estimators = 50, max_depth = 3)

         rfc.fit(X_train, y_train)
         rfc_pred = rfc.predict(X_test)
         rfc_accuracy = accuracy_score(y_test, rfc_pred)

         print('Random Forest Classifier Accuracy: {:.2f}%'.format(rfc_accuracy*100))

         Random Forest Classifier Accuracy: 82.93%
```

Observation - the random forest model has been successfully built on the dataset and has obtained a better accuracy of 82.93% compared to the previous model

7.2.1- corresponding confusion matrix has been displayed

Image 7.2.1-



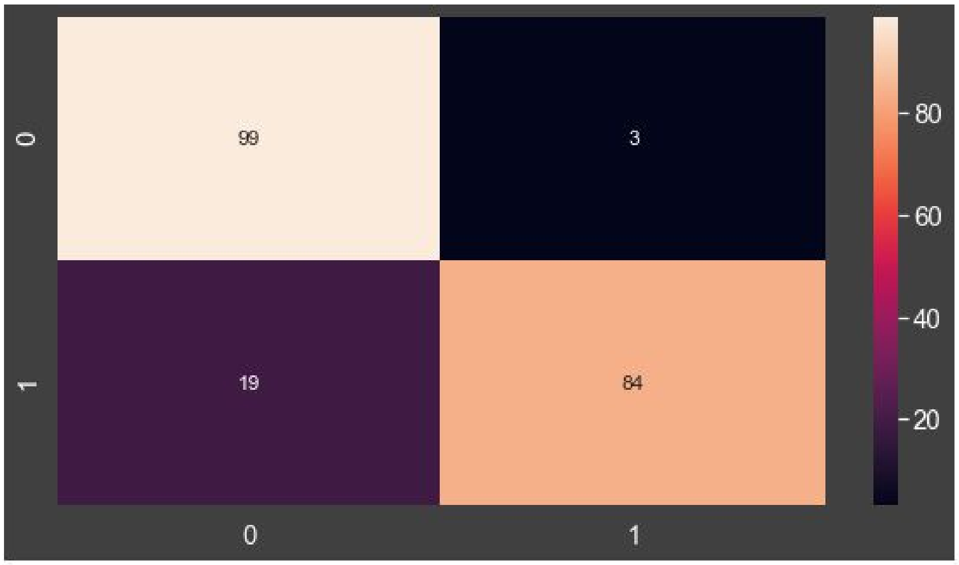
```
In [54]: cm = confusion_matrix(y_test,rfc_pred)
         sns.heatmap(cm,annot=True);
```

7.3- K Neighbors Classifier Model

Image 7.3-The following is an image of building knn classifier model on the data set

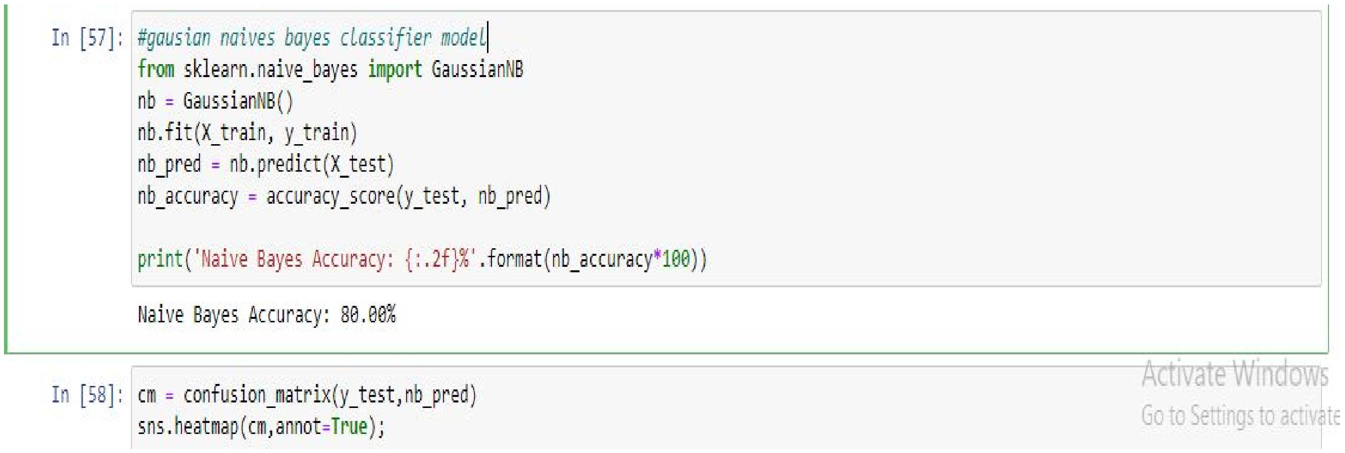
```
In [55]: from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors = 4)  # n_neighbors means k
         knn.fit(X_train, y_train)
         knn_pred = knn.predict(X_test)
         knn_accuracy = accuracy_score(y_test, knn_pred)
         print('KNeighborsClassifier Accuracy: {:.2f}%'.format(knn_accuracy*100))

         KNeighborsClassifier Accuracy: 89.27%
```

```
In [56]: cm = confusion_matrix(y_test,knn_pred)
         sns.heatmap(cm,annot=True);
```

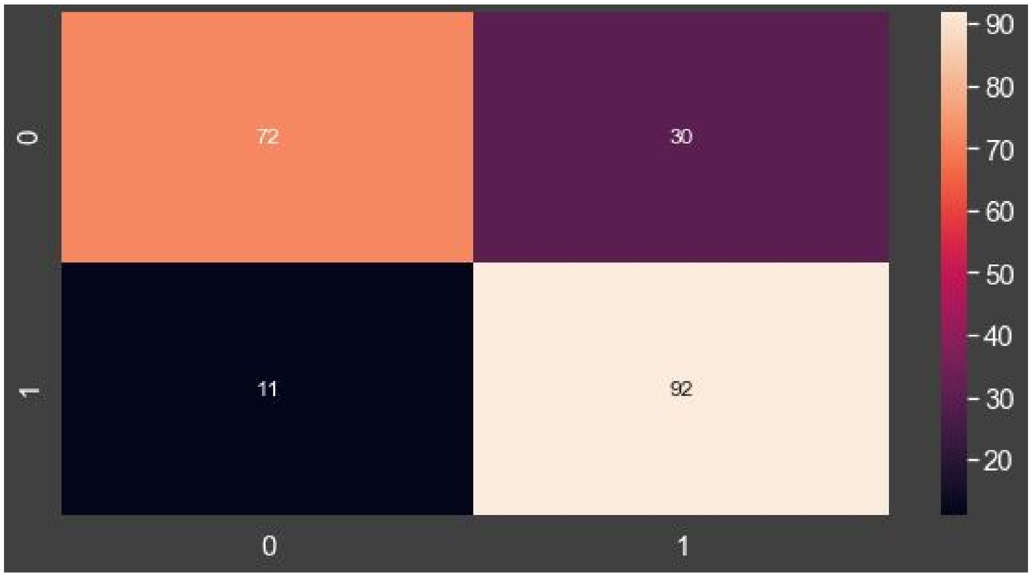


Observation- a knn model has been successfully built on the dataset and its corresponding confusion matrix has been represented
We observe that this model has yielded us with a high accuracy of 89.27%. Which is a fair result of feature scaling.

7.4- Gaussian Navie's Bayes Classifier
Image 7.4-



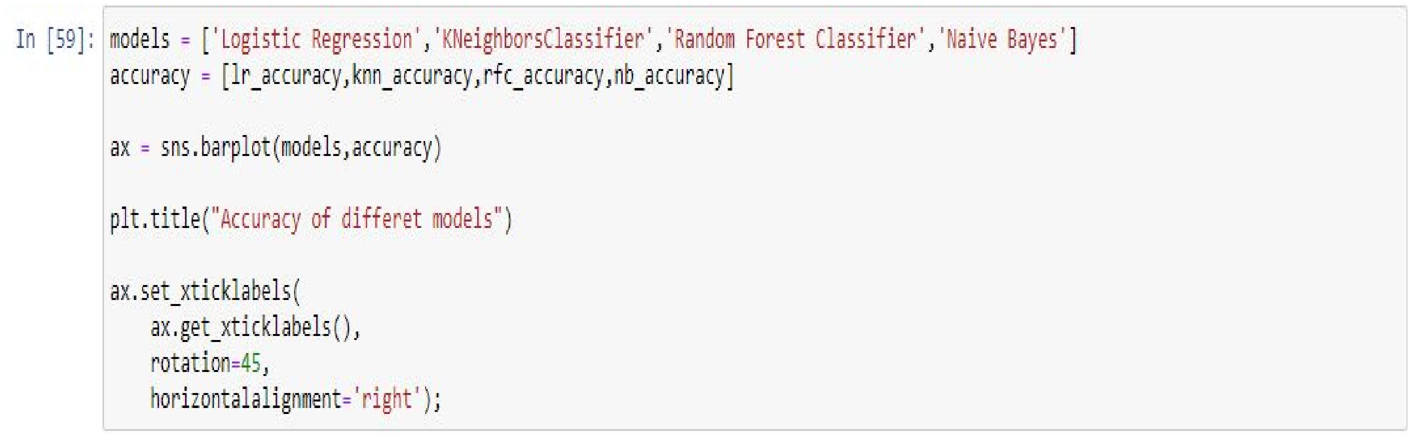
```
In [57]: #gausian naives bayes classifier model
         from sklearn.naive_bayes import GaussianNB
         nb = GaussianNB()
         nb.fit(X_train, y_train)
         nb_pred = nb.predict(X_test)
         nb_accuracy = accuracy_score(y_test, nb_pred)

         print('Naive Bayes Accuracy: {:.2f}%'.format(nb_accuracy*100))

         Naive Bayes Accuracy: 80.00%

In [58]: cm = confusion_matrix(y_test,nb_pred)
         sns.heatmap(cm,annot=True);
```



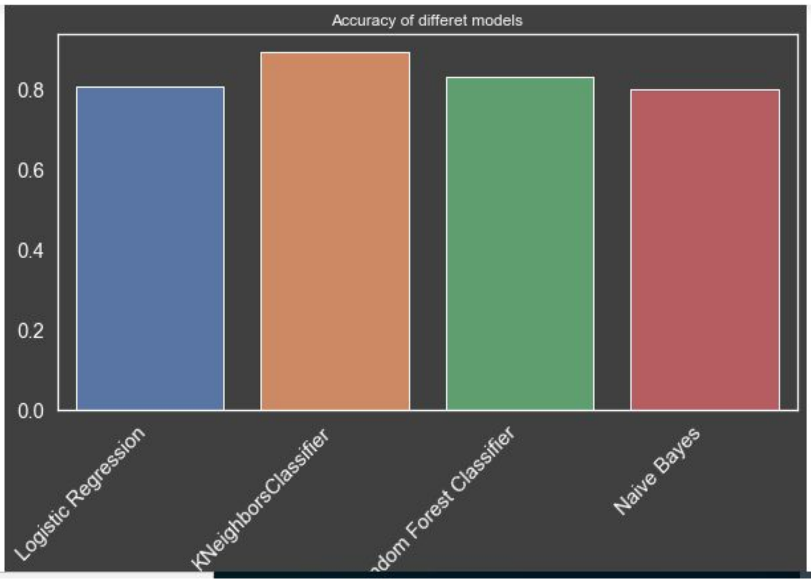
Observation - a successful model has been built and prediction is done on data followed by representing its confusion matrix.
This model has given us 80.00% accuracy

8.Concluding which model best suits this data set.

8.1- the following is a graph showing accuracy levels of different models
Image 8.1-



```
In [59]: models = ['Logistic Regression','KNeighborsClassifier','Random Forest Classifier','Naive Bayes']
         accuracy = [lr_accuracy,knn_accuracy,rfc_accuracy,nb_accuracy]

         ax = sns.barplot(models,accuracy)

         plt.title("Accuracy of differet models")

         ax.set_xticklabels(
             ax.get_xticklabels(),
             rotation=45,
             horizontalalignment='right');
```



Observation- Fromthe above image we can conclude that KNN model has given us the highest accuracy and that it is the best fit model for this data set.

CONCLUSION

It is concluded after performing thorough Exploratory Data analysis which include Stats models which are computed to get accuracy and also Heat maps ,countplots etc which are computed to get a clear understanding of the data set (which parameter has most abundant effect on the study case) and its come to point of getting the solution for the problem statement being , prediction of heart diseases occurrence. We have observed that men are more prone to these diseases in which majority are a result of typical angina chest pain along with high blood pressure and other influencing factors. Whereas on the other hand females are less prone to these circumstances.