# Assignment and Report Cover Sheet

1. Complete all of the details below and sign.
2. **No assignment will be accepted unless this form is completed in full, signed and dated**.
3. Hand this in to your lecturer in person or to ACBT reception or submit it electronically together with your assessment

| | |
|---|---|
| **STUDENT ID NUMBER: PIHHC183** | **NAME OF STUDENT: (PRINT CLEARLY)**<br>FAMILY NAME **DAYANANDA** OTHER NAME(S) **HIRUSHA** |
| **UNIT CODE / UNIT NAME: CSG2341 – INTELLIGENT SYSTEM** | **NAME OF LECTURER: MR. NIROSHAN BALASURIYA** |
| **TITLE/TOPIC OF ASSIGNMENT: ASSIGNMENT PART B – PROJECT IMPLEMENTATION AND PRESENTATION** | |
| **"I certify that the attached assignment is my own work and that any material drawn from other sources has been fully acknowledged".**<br>**Signed:** *Hirusha Dayananda* **Date: 21st September 2021** | |

## PENALTIES FOR LATE ASSIGNMENTS

1. If a student is unable to submit a within-semester assessment task (i.e. assignment) on or by the due date, the penalty will be 5% per working day. The mark will be zero after 5 working days. An exception may be granted if the student applies for an extension and provides an *Explained Absence* form together with:
   Medical certificate (signed by lecturer and given to the Welcome Centre by student), or
   Written explanation (signed by lecturer and given to the Manager Student Services by student), in the case of personal circumstances which have the potential to significantly affect the performance of the student.
2. Evidence must be submitted within 3 days ( By hard copy or an email to the Manager Examinations /Lecturer is also recommended).
3. Lecturers will undertake to provide feedback within 2 weeks of due date.

## ACADEMIC MISCONDUCT AND PLAGIARISM

Academic misconduct of any form is unacceptable. Academic misconduct includes, but is not limited to:

Plagiarism
Unauthorised collaboration;
Cheating in assessments
Fraudulently submitting work of another person
Theft of other students' work
Any other fraudulent practices

"**Plagiarism**" means to knowingly or unknowingly present as one's own work the ideas or writings of another without appropriate acknowledgment or referencing. This includes, but is not limited to:
Paraphrasing text without adequately stating the source;
Copying of visual representations (cartoons, line drawings, photos, paintings and software code)

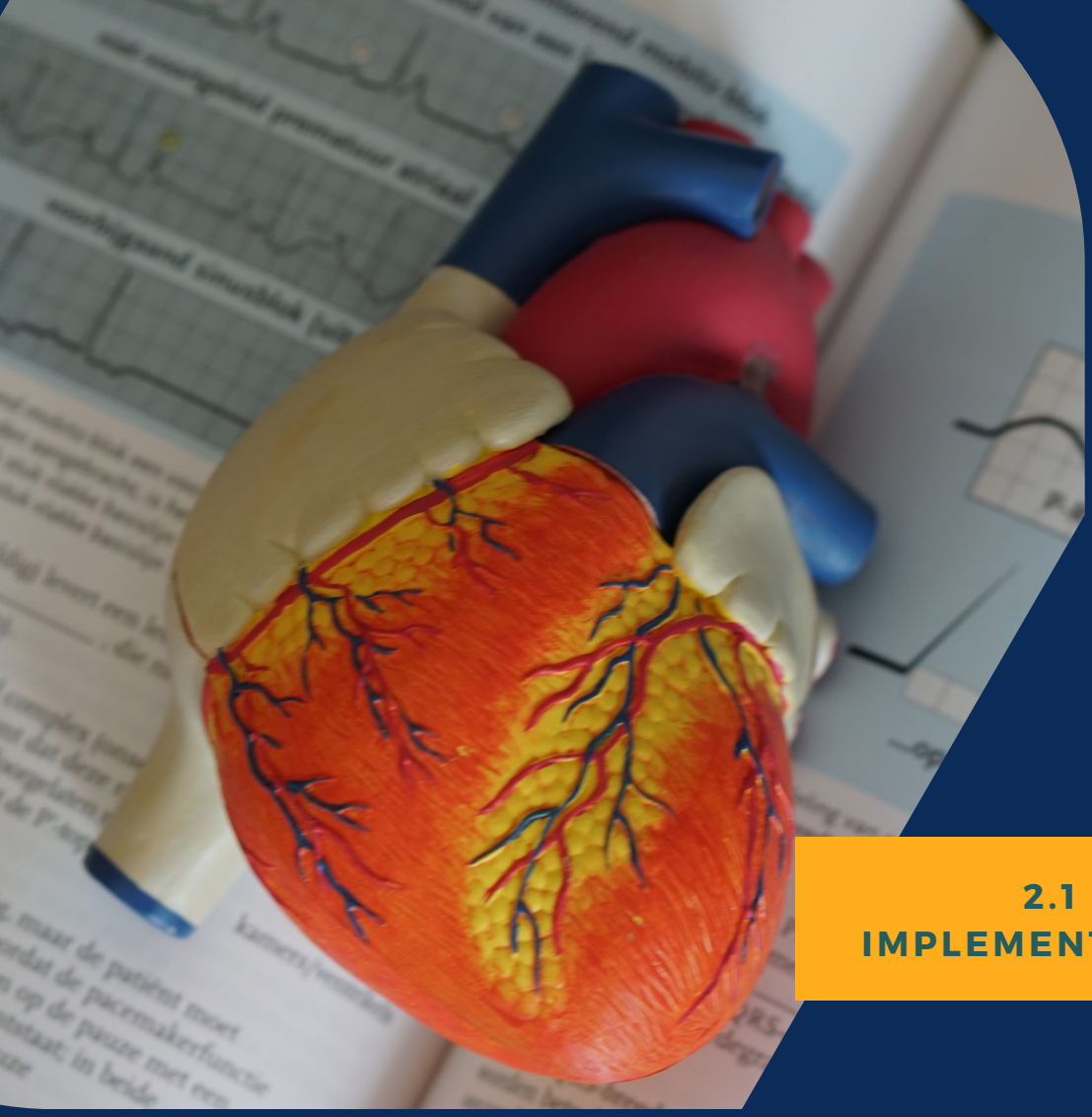
**All forms of cheating, plagiarism or collusion are regarded seriously and may result in penalties including loss of marks, exclusion from the unit or cancellation of enrolment. Lecturers submit assessments into *Turnitin* – a programme which scans and records your work to match it against electronic works of others on the internet. Further information see: Assessment Policy and/or the Student Misconduct Policy at:**
**http://www.acbt.net/policies**

✂==========✂==========✂==========✂=======

## ASSIGNMENT RECEIPT BY LECTURER / RECEPTIONIST

*To be completed by the student as proof of submission:.*

| | |
|---|---|
| **UNIT TITLE:** | **NAME OF STUDENT:**<br>FAMILY NAME OTHER NAME(S) |
| **LECTURER'S/RECEPTIONIST'S SIGNATURE:** | **NAME OF LECTURER / RECEPTIONIST:** |
| **DATE:** | **TOPIC OF ASSIGNMENT:** |



21st September 2021

2.1 PROJECT IMPLEMENTATION REPORT

# PROJECT IMPLEMENTATION REPORT ON MEDICAL IMAGE ANALYSIS

## For Cardiovascular Disease Diagnosis


**WRITTEN BY**

Hirusha Dayananda
Maheesha De Silva
Naveen Samarasinghe


**PRESENTED TO**

CSGI2341D - Intelligent systems

# Table of Contents

# 1. Abstract

This report was produced to meet the requirements of explaining the Machine Learning Algorithm that we all successfully constructed in order for the Topic we chose in "Medical image analysis for disease diagnosis in Cardiovascular (CVD)" in Assignment 1, which was the section in which we incorporated all of the descriptions of the techniques as well as more details on Machine Learning in relation to Cardiovascular. The main purpose of the implementation is to show the detections and make it prove how we got it each results that we discussed on the earlier project part 1.

In Assignment Part 2, we successfully implemented a python-based code that detects whether patients have Cardiovascular heart disease or not based on the given dataset with images for 10 years. We chose Jupyterlab to implement our code in, it was really easy for us when implementing because it was an open source application and commands such as importing and reading data were really easy because we can run it one by one. In reference to Assignment 1 Subsection 2, I have said that of all the approaches, I chose the "Logistic Regression" theory in constructing the Code to forecast heart problems. Logistic regression is a machine learning classification strategy for evaluating a dataset that includes one or more independent variables (IVs) that influence the outcome as well as a categorical dependent variable. There are 8 main process that were used when implementing the code, they are:

1. Imported the Libraries
2. Imported and Reading the Dataset
3. Data Analysis and Preprocessing the Dataset
4. Modelling of the Data
5. Feature Extraction
6. Exploratory Visual Analysis (Data Visualization)
7. Image Analysis
8. Imported the Logistic Regression
9. Model Evaluation
10. Overall Model Evaluation

As a result of the performance evaluation, we were able to effectively construct the detection code with the images and a data set, and even though some errors popped up, we were able to succeed by attempting to find everything such as the error that displayed us what the error was, and in the end, we were able to identify that there were 92 Patients with the risk of CVD Disease and 108 without the risk. The model we employed has a 55% accuracy rate.

## 2. Introduction/Background

Between 1990 and 2013, the number of fatalities from cardiovascular diseases grew by 41%, rising from 12.3 million to 17.3 million worldwide. Furthermore, the same condition is responsible for half of all deaths in the United States and other industrialized countries ***(Mozaffarian et al., 2016).*** As a result, early diagnosis of heart disorders is essential in order to avoid health issues. Machine learning has been widely employed in the modern healthcare field for employing data models to diagnose and forecast the existence of diseases. Machine learning is widely employed in practically every field around the world, including the healthcare industry. Machine learning is an artificial intelligence (AI) technology that allows systems to automatically learn and improve from experience without being explicitly programmed ***(Medwin Publishers, 2016).*** Furthermore, machine learning is the activity of parsing data, learning from it, and then making a conclusion or detection about something in the world ***(Abduljabbar et al., 2019)***.

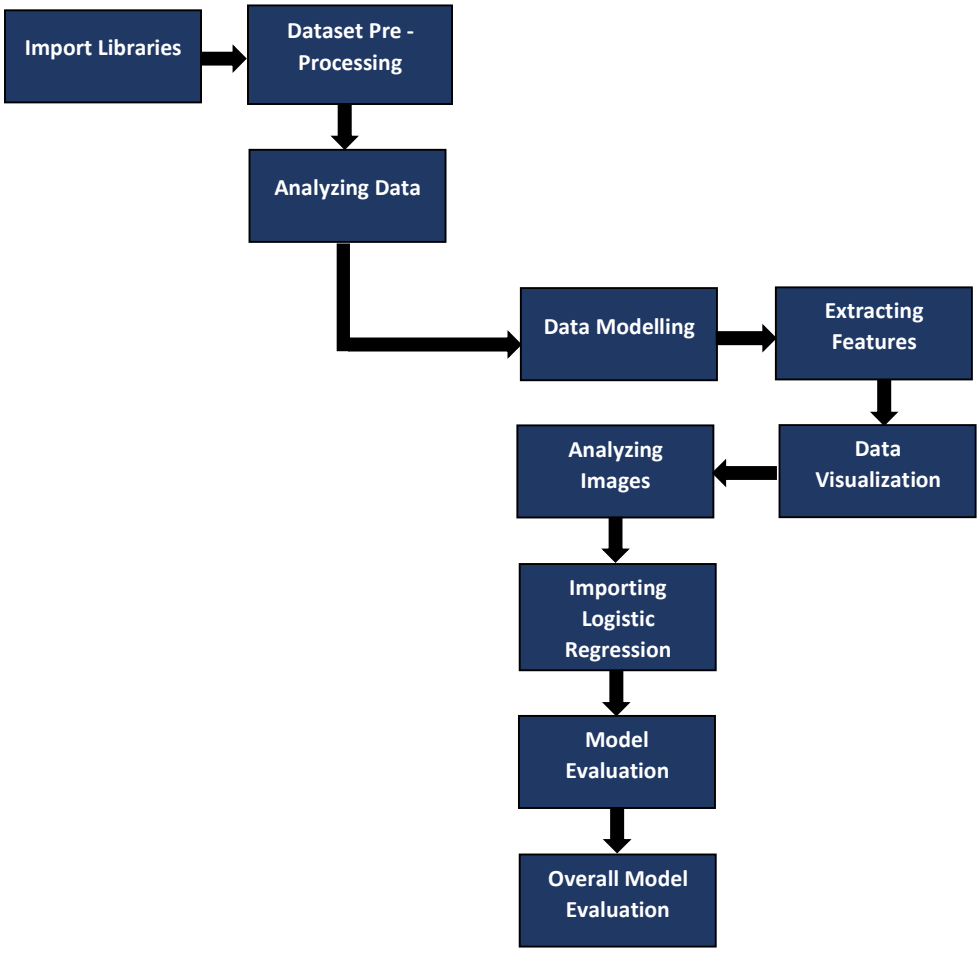
As mentioned in the Assignment we used the Logistic Regression theory to implement this because it is less difficult to apply, analyze, and train. Logistic Regression isn't used if the number of observations is less than the number of features; otherwise, it may result in overfitting which was easier. It makes no assumptions about class distributions in feature space. ***(A. S. T. Nishadi, 2019)*** The dataset used for the following is based on a set that we generated to meet the requirements and implement the algorithm to get the detections. This study's classification purpose is to detect whether the patient is at risk of future heart disease. The collection is made up of 200 records of patient data and 18 characteristics. The data analysis is done in Python using JupyterLab, a more flexible and sophisticated data science application platform.

The logistic regression function is also known as the sigmoid function, which aids in graph representation. The data should be imported first and then trained in this approach. The logistic regression algorithm is shown in graphs that indicate the difference between the qualities by applying equations. We must estimate the best and approximation coefficient from the training data and represent ***(Ramakrishnan et al., n.d.)***. It also gives high accuracy by utilizing several ways.

We began by first incorporating the necessary libraries into the code and creating a dataset in order to implement it to the future detections and found the count of the detections and many other with proceeding the feature extraction process and later on we created codes for visualization to display such as the histograms for each distributions to show it clearly along with pair plots, and finally we inserted the Logistic Regression theory to the code to find the accuracies of the models and other evaluation statistics and last we got a graph with the ROC.In order to start with we to install Jupyter and to import the libraries we have to make sure that we have previously installed our libraries and Jupyter through the Anaconda Prompt Shell the following are the libraries we used throughout our implementation.

1. Pandas
2. Numpy
3. StatsModels
4. Scipy.Stats
5. Sklearn
6. Matpotlib.pyplot
7. Seaborn
8. Keras
9. Matpotlib.image
10. Tensorflow

## 2.1 The Methodology




**[Figure 1 – Workflow of building the Machine Learning Model.]**

Throughout the implementation we used 18 attributes of data and they are:

1. **The Medical Image No** – The Number which represents the Patients Medical Image Scan of the Heart.
2. **Sex** – 0 represents the Male, 1 represents the Female.
3. **Age** – Age of the Patient.
4. **Current Smoker** – 0 represents the patient is not a smoker, 1 represents the patients is a smoker.
5. **Cigs Per Day** – The Number of Cigarettes consumed by the Patient.
6. **BP Meds** - 0 represents the patient does not take Blood Pressure Medicine, 1 represents the patients takes Blood Pressure Medicines.
7. **Prevalent Stroke** - 0 represents the patient haven't had a prev Stroke before,1 represents the patient had a prev Stroke before.

8. **Prevalent Hyp -** 0 represents the patient haven't had a prev hypertension before,1 represents the patient had a prev hypertension before.
9. **Diabetes -** 0 represents the patient doesn't have diabetes, 1 represents the patients have diabetes**.**
10. **Fam Check Up –** represents the patient has not done a Family Checkup, 1 represents the patient has done a family checkup**.**
11. **Total Cholesterol** – The total number of Cholesterol the Patient has.
12. **Troponin Level** – The total level of Troponin the Patient has.
13. **Rest ECG** – This detects whether the patients has done a ECG or not.
14. **Systolic BP** – The level of SysBP in the Patient.
15. **Diastolic BP** – The level of in the Patient.
16. **Body Mass Index** – The number of Body Mass Index of the patient.
17. **Heart Rate** – The Rate of the Patients Heart.
18. **Ten Year CVD Target** – The detections based on 1 and 0 (0 represents the patients who do not have the ten-year risk of CVD and 1 represents the patients who have a risk of ten-year CVD disease.)

This report was created to fulfil the criteria of demonstrating the Machine Learning Algorithm that we should successfully built in order for the Topic we chose, "Medical Image analysis for disease identification in Cardiovascular (CVD)." Assignment 1 was the section in which we included all of the approach descriptions as well as further information on Machine Learning in connection to Cardiovascular. The major goal of the implementation is to demonstrate the detections and authenticate how we obtained them, as stated in the previous project part 1.

# 3. Approach/ Implementation Details

# The Implementation

## 3.1 Importing the Libraries

**Detecting Cardiovascular Heart Diseases through Medical Image Analysis in Logistic Regression of Machine Learning Algorithms by Python Jupyterlab**

*1. Student Details: Hirusha Dayananda (PIHHC183), Maheesha De Silva (DEMRC183) and Naveen Samarasinghe (SANDC182)*

*2. Unit Details: CSG2341D - Intelligent System*

*3. Assignment Details: Assignment 2 - Project Implementation*

*4. Date: 16th of September 2021*

**1. Importing the Libraries**

In [*]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import scipy.stats as st
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from matplotlib import pyplot
from matplotlib.image import imread
```

Here we have loaded the Libraries into Jypyter Lab in order to build a machine learning model successfully. In accession to that, the required libraries are used as supportive applications are to be loaded as well.

1. Panda – Used for data manipulation and Analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.ts name is a play on the phrase "Python data analysis" itself.
2. Numpy – An open source numerical library. NumPy contains a multi-dimensional array and matrix data structures. It can be utilized to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.
3. Statsmodel API - Statsmodels is a Python package that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator.
4. Scipy.stats - This module contains a large number of probability distributions, summary and frequency statistics, correlation functions and statistical tests, masked statistics, kernel density estimation, quasi-Monte Carlo functionality, distributions, and more.
5. Sklearn- The library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction and considered as the most important library in python.
6. Matplotlib.pyplot – It is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

7. Seaborn- Seaborn is an open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works easily with dataframes and the Pandas library. The graphs created can also be customized easily.
8. Keras – Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.
9. Matplotlib.image - The image module in matplotlib library is used for working with images in Python. The image module also includes two useful methods which are imread which is used to read images and which is used to display the image.
10. Tensorflow – An open-source library for numerical computation and large-scale machine learning that ease Google Brain TensorFlow, the process of acquiring data, training models, serving detection, and refining future results.

### 3.2 Importing and Reading the Dataset

**2. Importing and Reading the Dataset**

```
In [137]: df = pd.read_csv('Cardiovascular Disease Prediction Dataset.csv')
          df.head()
```

Out[137]:

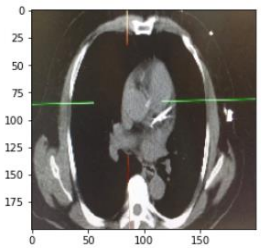
| | Medical Image No | sex | age | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | prevalentHyp | diabetes | FamCheckUp | totChol | troponinLevel | restECG | sysBP | dia |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2396 | 1 | 30 | 1 | 13 | 0 | 1 | 1 | 1 | 1 | 177 | 7.0 | 81 | 132 | |
| 1 | 2523 | 1 | 39 | 0 | 12 | 0 | 0 | 1 | 1 | 0 | 166 | 9.0 | 180 | 128 | |
| 2 | 2317 | 1 | 69 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 237 | 11.0 | 83 | 124 | |
| 3 | 2524 | 0 | 31 | 0 | 5 | 0 | 0 | 1 | 1 | 1 | 208 | 6.0 | 58 | 122 | |
| 4 | 2867 | 0 | 44 | 0 | 13 | 0 | 0 | 1 | 0 | 1 | 180 | 13.0 | 88 | 122 | |

Next, it has loaded the cardiac detection data into Jupyter Lab using the Cardiovascular Disease Prediction Dataset CSV file in order to develop the logistic regression model. We have set the main variable to “df” which is to be declared inside the function of the reading the csv file. The main variable is “df “here. Next line “head” is a method used to return to top (5 by default) rows of a data frame or series. It is useful for quickly testing if your object has the right type of data in it.

**Previewing of The Data**

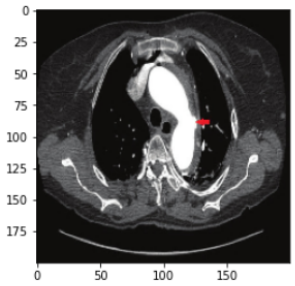
```
In [4]: src = 'Medical Image No 2396.jpg'
        print (src)
        photo = load_img(src, target_size=(200, 200))
        pyplot.imshow(photo)
        print(photo.size)
```

```
Medical Image No 2396.jpg
(200, 200)
```



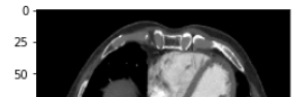
In [3]:
```
src = 'Medical Image No 2523.jpg'
print (src)
photo = load_img(src, target_size=(200, 200))
pyplot.imshow(photo)
print(photo.size)
```

```
Medical Image No 2523.jpg
(200, 200)
```




In [21]:
```
src = 'Medical Image No 2317.jpg'
print (src)
photo = load_img(src, target_size=(200, 200))
pyplot.imshow(photo)
print(photo.size)
```

```
Medical Image No 2317.jpg
(200, 200)
```




Included in the code are 5 medical images of the inside of the heart taken from scans of patients to detect whether they have or will have CVD Disease in the future or not, and they were successfully inserted to extract them and split them into training and testing. These photos are still being analyzed based on the patients' medical image numbers.

### 3.3 Data Analysis and Preprocessing the Dataset

Data Analysis was carried out using Jupyter Lab using Python. The following steps were implemented in order to process the logistics regression. In order to build up a more accurate Machine Learning model, data preprocessing is required. Data pre-processing is the process of cleaning the data. It will remove all the NAN values from our data. This process is also known as Data Wrangling. This includes the identification of missing data, noisy data and inconsistent data and all others below are involved in data preprocessing. With the aid of importing pandas, we can quickly evaluate a dataset and obtain an overview of the type and amount of data we're working with, as well as some key statistics.

**3. Data Analysis and Preprocessing the Dataset**

**Sample of the DataSet**

In [68]:
```
df.sample(2)
```

Out[68]:

| | Medical Image No | sex | age | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | prevalentHyp | diabetes | FamCheckUp | totChol | troponinLevel | restECG | sysBP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **144** | 2272 | 1 | 61 | 1 | 10 | 0 | 1 | 0 | 1 | 1 | 176 | 5.0 | 85 | 130 |
| **16** | 2119 | 0 | 58 | 0 | 16 | 0 | 1 | 1 | 1 | 1 | 202 | 6.0 | 80 | 128 |

The two rows of the DataFrame that we imported were chosen at random and are shown above. Data Samples returns the data set of any randomly picked two rows, including all 18 attributes, because we only entered two rows here.

### 3.3.1 Shape of the Dataset

**Shape of the DataSet**

```
In [69]: df.shape
Out[69]: (200, 18)
```

The shape returns the number of rows and columns in the dataset. In our case the data set contains 200 records and 18 attributes.

### 3.3.2 Key Values of the Dataset

**Key Values of the DataSet**

```
In [70]: df.keys()
Out[70]: Index(['Medical Image No', 'sex', 'age', 'currentSmoker', 'cigsPerDay',
                'BPMeds', 'prevalentStroke', 'prevalentHyp', 'diabetes', 'FamCheckUp',
                'totChol', 'troponinLevel', 'restECG', 'sysBP', 'diaBP', 'BMI',
                'heartRate', 'TenYearCVDTarget'],
               dtype='object')
```

The keys method contains the names of the dataset's columns. The data set in our scenario has 18 columns.

### 3.3.3 Information on the Dataset

**Information of the DataSet**

```
In [71]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Medical Image No  200 non-null    int64
 1   sex               200 non-null    int64
 2   age               200 non-null    int64
 3   currentSmoker     200 non-null    int64
 4   cigsPerDay        200 non-null    int64
 5   BPMeds            200 non-null    int64
 6   prevalentStroke   200 non-null    int64
 7   prevalentHyp      200 non-null    int64
 8   diabetes          200 non-null    int64
 9   FamCheckUp        200 non-null    int64
 10  totChol           200 non-null    int64
 11  troponinLevel     200 non-null    float64
 12  restECG           200 non-null    int64
 13  sysBP             200 non-null    int64
 14  diaBP             200 non-null    int64
 15  BMI               200 non-null    float64
 16  heartRate         200 non-null    int64
 17  TenYearCVDTarget  200 non-null    int64
dtypes: float64(2), int64(16)
memory usage: 28.2 KB
```

The info method returns all extra information such as the range index, count of data columns, non-null count, and data type of each column of the dataset. In our instance, the data collection comprises 200 non-null counts and data types with 2 float columns and 16 integer columns. The total memory use is also calculated at 28.2 kb.

### 3.3.4 Description of the Dataset

**Description of the DataSet**

```
In [72]: df.describe()
```

Out[72]:

| | Medical Image No | sex | age | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | prevalentHyp | diabetes | FamCheckUp | totChol | trop |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | 200.000000 | |
| mean | 2321.395000 | 0.465000 | 55.540000 | 0.470000 | 11.195000 | 0.045000 | 0.470000 | 0.360000 | 0.530000 | 0.580000 | 196.410000 | |
| std | 322.563568 | 0.500025 | 17.129668 | 0.500352 | 5.642354 | 0.207824 | 0.500352 | 0.481205 | 0.500352 | 0.494797 | 25.574051 | |
| min | 1789.000000 | 0.000000 | 30.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 150.000000 | |
| 25% | 2043.500000 | 0.000000 | 40.000000 | 0.000000 | 6.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 174.000000 | |
| 50% | 2318.500000 | 0.000000 | 53.000000 | 0.000000 | 11.500000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 196.500000 | |
| 75% | 2591.000000 | 1.000000 | 71.000000 | 1.000000 | 16.250000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 219.000000 | |
| max | 2888.000000 | 1.000000 | 85.000000 | 1.000000 | 20.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 240.000000 | |

The descriptions method returns statistics about the dataset's numerical columns. Such as the count, mean, standard deviation, minimum, 25%, 50%, 75%, and finally the maximum.

### 3.3.5 Explanation on the Columns of the Dataset

**Explanation of the Columns in the DataSet**

```
In [73]: fo = ["Number representing the patients medical image","0: Male, 1: Female","Age of the patient","0: No, 1: Yes"," The Amount of


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

```
Medical Image No:                       Number representing the patients medical image
sex:                    0: Male, 1: Female
age:                    Age of the patient
currentSmoker:                  0: No, 1: Yes
cigsPerDay:                      The Amount of Cigs taken in a day
BPMeds:                 0: Not taking BP Meds, 1: Taking BP Meds
prevalentStroke:                        0: Not Had Stroke, 1: Had Stroke
prevalentHyp:                   0: No Prevalent Strokes, 1: Have Prevalent Strokes
diabetes:                       0: No Diabetes, 1: Diabetes
FamCheckUp:                     0: Not Had Family Check Up, 1: Family Check Up
totChol:                        Total Number of Cholestral
troponinLevel:                  Number of Troponin Level
```

For further clarification, the descriptions above include explanations of each column, such as what the numbers indicate.

### 3.3.6 Identifying Missing Values

**Identifying Missing Values**

```
In [76]: df.isna().sum()
```

```
Out[76]: Medical Image No    0
sex                 0
age                 0
currentSmoker       0
cigsPerDay          0
BPMeds              0
prevalentStroke     0
prevalentHyp        0
diabetes            0
FamCheckUp          0
totChol             0
troponinLevel       0
restECG             0
sysBP               0
diaBP               0
BMI                 0
heartRate           0
TenYearCVDTarget    0
dtype: int64
```

```
In [27]: count=0
for i in df.isnull().sum(axis=1):
    if i>0:
        count=count+1
print('The Total Number of Rows With Missing Values is ', count)
print('Since it is only',round((count/len(df.index))*100), 'percent of the entire dataset the rows with missing values are exclu
```

```
The Total Number of Rows With Missing Values is  0
Since it is only 0 percent of the entire dataset the rows with missing values are excluded.
```

In additionally, the number of missing values has been discovered for cleaning up the existing dataset. The total number of missing values depending on the attributes is listed below. From the output above we got the total number of rows with missing value is 0. In this case, since it is only 0 % of the entire dataset, the rows with missing values are excluded. The Pandas Data Frame was used to calculate the total percentage of missing values in the column.

### 3.3.7 Removing NULL values from the Dataset

**Removing NULL Values from the Dataset**

```
In [77]: df.dropna(axis = 0, inplace = True)
print(df.shape)
```

```
(200, 18)
```

The following code employs the Pandas dropna() method, which was used to examine the drop rows/columns having Null values.

### 3.3.8 Description of the Target

**Description of the Target**

```
In [78]: df["TenYearCVDTarget"].describe()

Out[78]: count    200.000000
         mean       0.460000
         std        0.499648
         min        0.000000
         25%        0.000000
         50%        0.000000
         75%        1.000000
         max        1.000000
         Name: TenYearCVDTarget, dtype: float64
```

The descriptions method specifically returns statistics concerning the Ten-Year CVD Targets. For example, the count, mean, standard deviation, minimum, 25%, 50%, 75%, and ultimately the maximum.

### 3.3.9 Count of the Target

**Count of the Target**

```
In [79]: df['TenYearCVDTarget'].value_counts()

Out[79]: 0    108
         1     92
         Name: TenYearCVDTarget, dtype: int64
```

The count returns the total number of patients who are at a risk of CVD Diseases and who are not because the value count is assigned to the attribute “TenYearCVDTarget”. The following shows there are 108 patients without the risk of CVD diseases and 92 with the risk of heart diseases analyzing through the medical images and the dataset.

### 3.3.10 Uniqueness of the Target

**Uniqueness of the Target**

```
In [80]: df["TenYearCVDTarget"].unique()

Out[80]: array([0, 1], dtype=int64)
```

To obtain the unique values of the column, we can utilize the unique() function on a variable of interest. Assume we wish to determine the unique values of the data frame's column 'TenYearCVDTarget.' This would return all of the Targets in the data frame such as show above there 0’s and 1’s only. We can utilize pandas' unique function on the column of interest.

### 3.3.11 The Correlation between Columns

**The Correlation between Coloumns**

```
In [81]: print(df.corr()["TenYearCVDTarget"].abs().sort_values(ascending=False))
```

```
TenYearCVDTarget    1.000000
restECG             0.140881
heartRate           0.125340
troponinLevel       0.121955
BMI                 0.067816
diaBP               0.061140
sysBP               0.047354
prevalentStroke     0.045025
prevalentHyp        0.044309
BPMeds              0.041618
age                 0.037177
totChol             0.029998
sex                 0.024539
cigsPerDay          0.014153
Medical Image No    0.007650
FamCheckUp          0.007317
currentSmoker       0.004824
diabetes            0.004824
Name: TenYearCVDTarget, dtype: float64
```

The corr() calculates the pairwise correlation of all columns in related to the Dataset. Any missing values are immediately filtered out. It is disregarded for any non-numeric data type columns in the dataframe.

### 3.4 Modelling the Data

The following outcomes are used to indicate the logistic regression. Logistic regression is mainly used to for detection and also calculating the probability of success.

**4. Modelling the Data**

```
In [74]: from statsmodels.tools import add_constant as add_constant
         df_constant = add_constant(df)
         df_constant.head()
```

Out[74]:

| | const | Medical Image No | sex | age | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | prevalentHyp | diabetes | FamCheckUp | totChol | troponinLevel | restECG | sys |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 2396 | 1 | 30 | 1 | 13 | 0 | 1 | 1 | 1 | 1 | 177 | 7.0 | 81 | |
| **1** | 1.0 | 2523 | 1 | 39 | 0 | 12 | 0 | 0 | 1 | 1 | 0 | 166 | 9.0 | 180 | |
| **2** | 1.0 | 2317 | 1 | 69 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 237 | 11.0 | 83 | |
| **3** | 1.0 | 2524 | 0 | 31 | 0 | 5 | 0 | 0 | 1 | 1 | 1 | 208 | 6.0 | 58 | |
| **4** | 1.0 | 2867 | 0 | 44 | 0 | 13 | 0 | 0 | 1 | 0 | 1 | 180 | 13.0 | 88 | |

The first stage in the modeling process will be to add a constant value to the cleaned dataset.

## 3.5 Feature Extraction

And for the next stage, we're looking at the parameter and the outcome of our logistic regression model.

### 5. Feature Extraction

In [75]:
```
st.chisqprob = lambda chisq, df: st.chi2.sf(chisq, df)
cols=df_constant.columns[:2]
model=sm.Logit(df.TenYearCVDTarget,df_constant[cols])
result=model.fit()
result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.689914
         Iterations 4
```

Out[75]:

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | TenYearCVDTarget | **No. Observations:** | 200 |
| **Model:** | Logit | **Df Residuals:** | 198 |
| **Method:** | MLE | **Df Model:** | 1 |
| **Date:** | Thu, 16 Sep 2021 | **Pseudo R-squ.:** | 4.241e-05 |
| **Time:** | 16:05:15 | **Log-Likelihood:** | -137.98 |
| **converged:** | True | **LL-Null:** | -137.99 |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 0.9139 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | -0.0496 | 1.033 | -0.048 | 0.962 | -2.075 | 1.976 |
| **Medical Image No** | -4.77e-05 | 0.000 | -0.108 | 0.914 | -0.001 | 0.001 |

According to the logistic data above, P >= 0.05 indicates a modest statistically significant link with the likelihood of heart disease. As a result, the feature extraction method was employed to eliminate the qualities with the greatest P values. The procedure will be repeated until all of the attributes have P values less than 0.05.

In [14]:
```
def back_feature_elem (data_frame,dep_var,col_list):
    """ Takes in the dataframe, the dependent variable and a list of column names, runs the regression repeatedly eleminating fea
    P-value above alpha one at a time and returns the regression summary with all p-values below alpha"""

    while len(col_list)>-1 :
        model=sm.Logit(dep_var,data_frame[col_list])
        result=model.fit(disp=0)
        largest_pvalue=round(result.pvalues,10).nlargest(5)
        if largest_pvalue[0]<(2):
            return result
            break
        else:
            col_list=col_list.drop(largest_pvalue.index)

result=back_feature_elem(df_constant,df.TenYearCVDTarget,cols)
result.summary()
```

Out[14]:

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | TenYearCVDTarget | **No. Observations:** | 200 |
| **Model:** | Logit | **Df Residuals:** | 198 |
| **Method:** | MLE | **Df Model:** | 1 |
| **Date:** | Thu, 16 Sep 2021 | **Pseudo R-squ.:** | 4.241e-05 |
| **Time:** | 15:56:39 | **Log-Likelihood:** | -137.98 |
| **converged:** | True | **LL-Null:** | -137.99 |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 0.9139 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **const** | -0.0496 | 1.033 | -0.048 | 0.962 | -2.075 | 1.976 |
| **Medical Image No** | -4.77e-05 | 0.000 | -0.108 | 0.914 | -0.001 | 0.001 |

The output seen above is the outcome of feature extraction. For the cardiac dectection data, the logistic regression equation is as follows.

$$\text{logit}(P) = \log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 * \text{Sex} + \beta_2 * \text{age} + \beta_3 * \text{cigsPerDay} + \beta_4 * \text{totChol} + \beta_5 * \text{sysBP} + \beta_6 * \text{glucose}$$

### 3.5.1 Interpreting the Data

The following methods indicates the accuracy measurements.

**Interpreting the Data**

```
In [131]: params = np.exp(result.params)
          conf = np.exp(result.conf_int())
          conf['OR'] = params
          pvalue=round(result.pvalues,3)
          conf['pvalue']=pvalue
          conf.columns = ['CI 95%(2.5%)', 'CI 95%(97.5%)', 'Odds Ratio','pvalue']
          print ((conf))
```

```
                  CI 95%(2.5%)  CI 95%(97.5%)  Odds Ratio  pvalue
const                 0.125576       7.211144    0.951600   0.962
Medical Image No      0.999088       1.000817    0.999952   0.914
```

The accuracy of OR is estimated by using 95% confidence interval (CI). A large CI represents the low level of precision of OR and also small CI represents the higher precision of OR. However, 95% CI does not indicate the statistical significance unlike the p value.

- According to the fitted model, the odds of diagnosed with heart disease of males (78.8%) is higher than the females.
- Further, the odds of diagnosis with CHD is increase approximately 7% for a one-year age increase (1.067571)
- In addition to that, additional cigarette has risk of 2% increase in odds of CHD.
- Furthermore, odds of sysBP has 1.7% increase in every unit increase.
- No significance changes in the total cholesterol level and glucose level.

### 3.6 Exploratory Visual Analysis (Data Visualization)

The following visualization are derived through the Jupyter Lab for display predicators.

#### 3.6.1 Distribution of each Dataset

The purpose of this step is to see the distribution of each data. From the histogram above, we can understand better the distribution of the data.

6. Exploratary Visual Analysis (Data Visualization)

Distribution of each Dataset

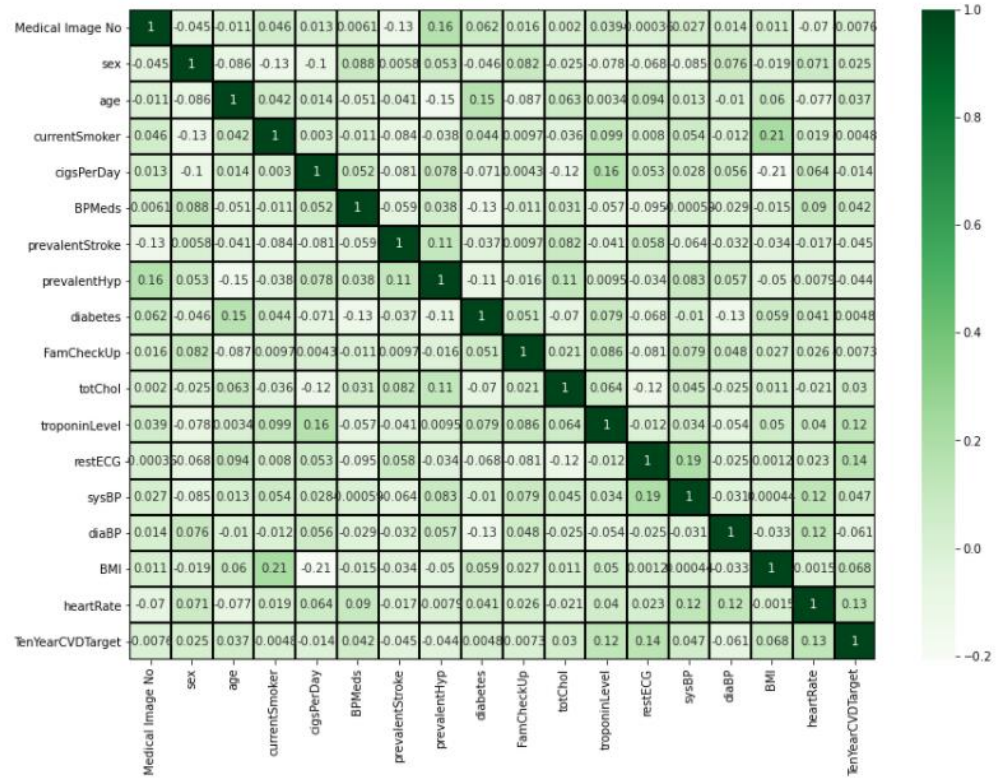
```
In [132]: def draw_histograms(dataframe, features, rows, cols):
              fig=plt.figure(figsize=(20,20))
              for i, feature in enumerate(features):
                  ax=fig.add_subplot(rows,cols,i+1)
                  dataframe[feature].hist(bins=20,ax=ax,facecolor='midnightblue')
                  ax.set_title(feature+" Distribution",color='DarkRed')

              fig.tight_layout()
              plt.show()
          draw_histograms(df,df.columns,7,3)
```

#### 3.6.2 Correlation Matrix of the Dataset

The correlation matrix in this case is a tabular data set that represents the 'correlations' between pairs of variables in the data. Each row and column represent a variable, and each number in this matrix reflects the correlation coefficient between the variables represented by the respective row and column. A pairs plot shows the distribution of single variables as well as the relationships between two variables in the following data set.

**Correlation Matrix of the Dataset**

```
In [83]: plt.figure(figsize = (14, 10))
         sns.heatmap(df.corr(), cmap='Greens',annot=True, linecolor='Black', linewidths=1.0)
         plt.show()
```

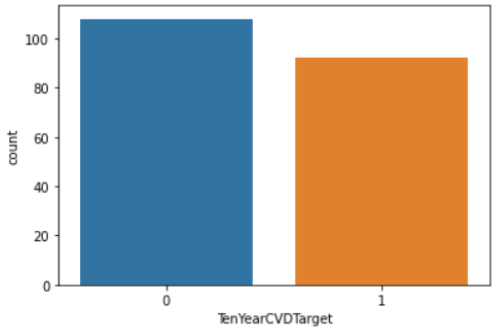



## 3.6.3 Count Plot based on the diagnosis of the Patients

**Countplot based on the CVDTarget whether the Patients are at risk of having Cardiovacular Heart Disease or not.**

```
In [84]: sns.countplot(x='TenYearCVDTarget',data=df)
```

```
Out[84]: <AxesSubplot:xlabel='TenYearCVDTarget', ylabel='count'>
```



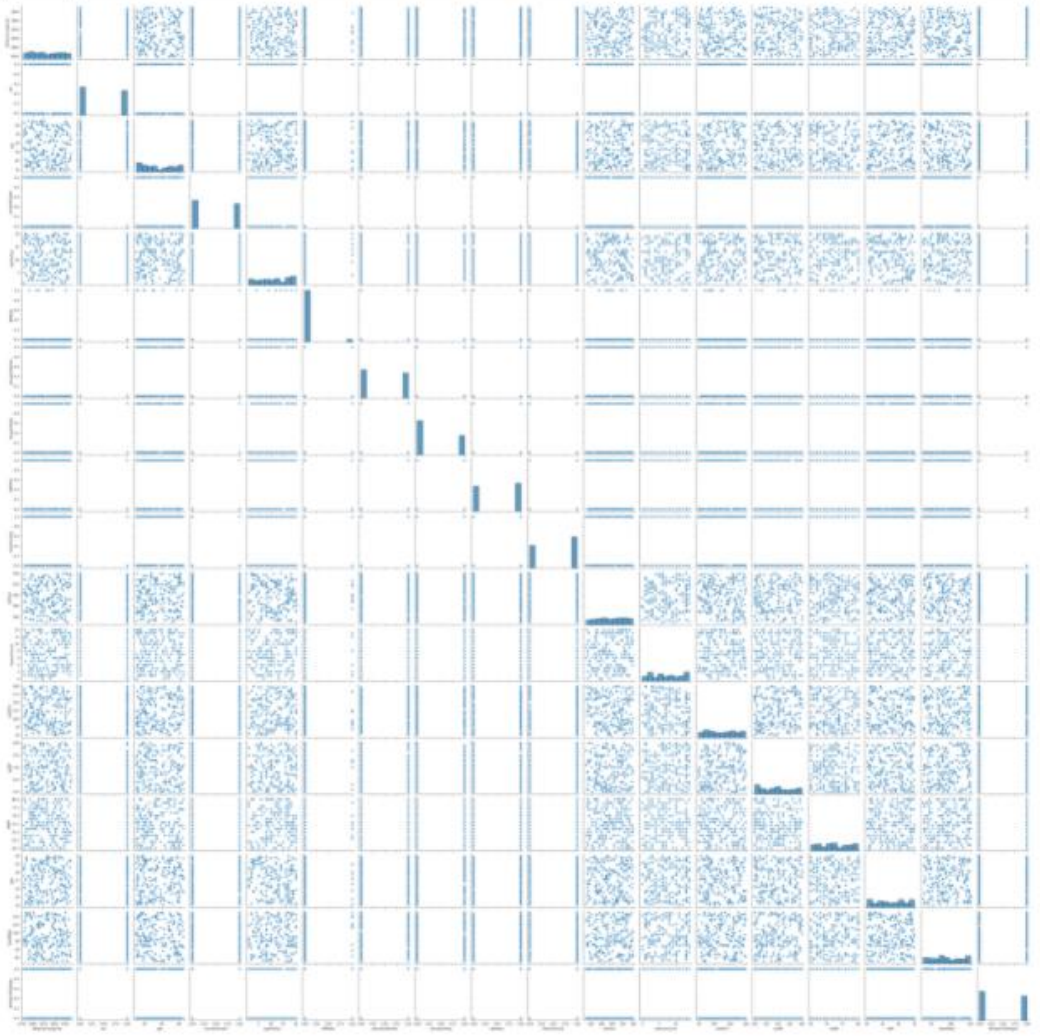

From the Figure above, we can conclude that there are 108 Patients with NO CARDIOVASCULAR HEART DISEASE and 92 WITH THE RISK OF CARDIOVASCULAR HEART DISEASE

### 3.6.4 Pair plot

A pair plot shows the distribution of single variables as well as the relationships between two variables in the following data set.

Pairplot of the Data Set

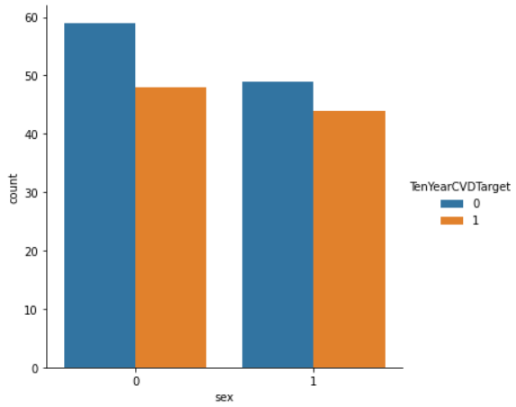
```
In [107]: sns.pairplot(df)
          plt.show()
```



### 3.6.5 Other Count plots and Sub plots

**Countplot of the Patients based on their Sex to identify which gender has high predictions of Heart Diseases**

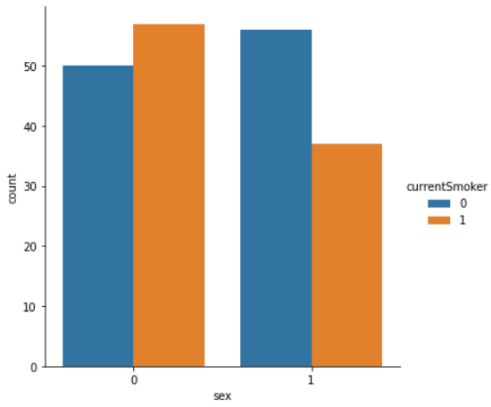
```
In [85]: sns.catplot(data=df, kind='count', x='sex',hue='TenYearCVDTarget')
         plt.show()
```




From the Figure above, we can conclude that Males (0) have a higher risk of CVD Diseases than Females (1)

**Countplot of the Patients based on their Sex whether they are Current Smoker or Not**

```
In [86]: sns.catplot(data=df, kind='count', x='sex',hue='currentSmoker')
         plt.show()
```

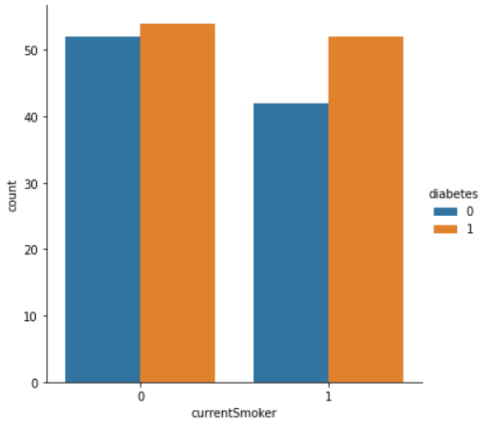



From the figure we can conclude that we have males who smokes higher than the Females.

- Male (0) and Female (1)
- Not smoker (0) and Smoker (1)

**Countplot based on the Diabetes Level of the Patients whether they are Current Smoker or Not**

In [87]:
```
sns.catplot(data=df, kind='count', x='currentSmoker',hue='diabetes')
plt.show()
```

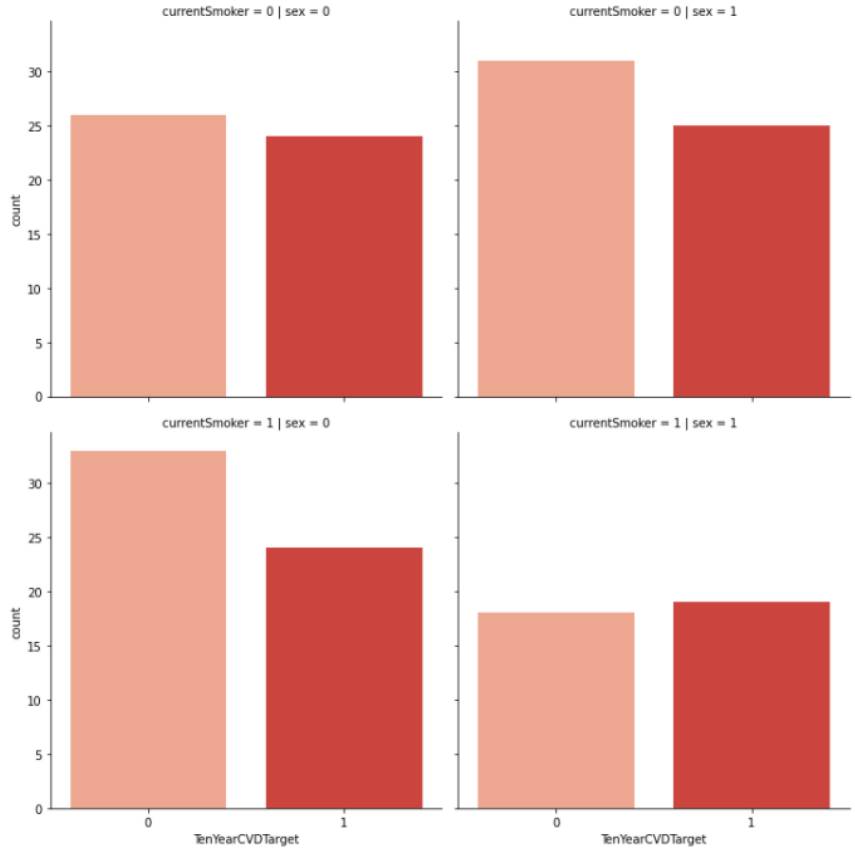



Based on the findings, we may conclude that diabetic individuals smoke more than non-diabetic patients.

- No Diabetes (0) Diabetes (1)
- Not smoker (0) Smoker (1)

**Countplot - The Subplots of the patients affecting with CVD Dieases on basis of their sex and current smoking**

In [88]:
```
sns.catplot(data=df, kind='count', x='TenYearCVDTarget', col='sex',row='currentSmoker', palette='Reds')
plt.show()
```



### 3.6.6 Percentage of Patients with CVD Heart Problems

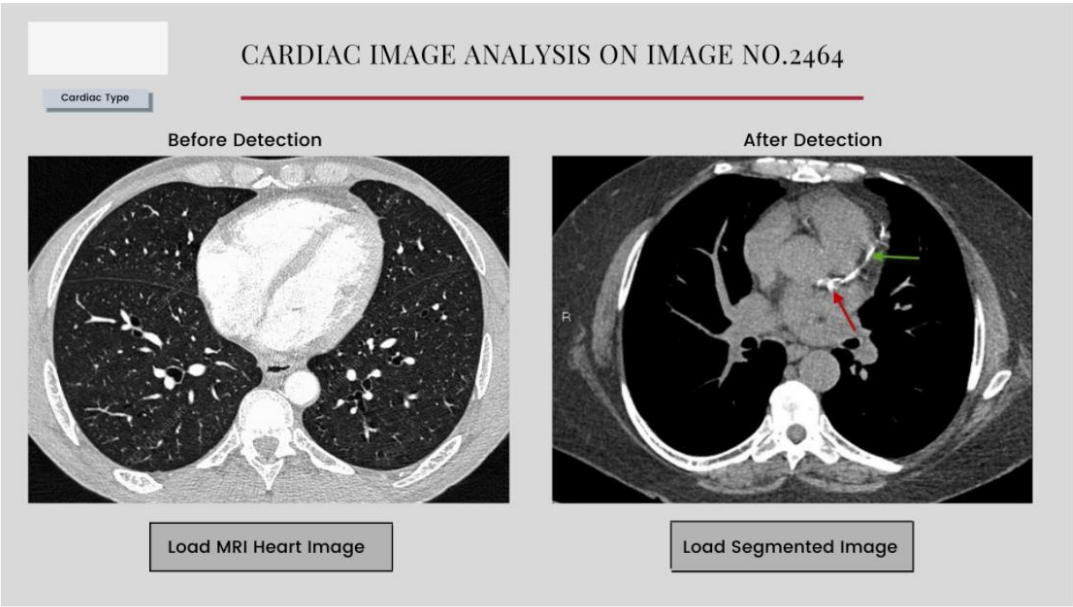
```
In [51]: print("Percentage of patience without Cardiovascular Heart Problems within 10 years: "+str(round(TenYearCVDTarget_temp[0]*100/20
         print("Percentage of patience with Cardiovascular Heart Problems within 10 years: "+str(round(TenYearCVDTarget_temp[1]*100/200,2)

         #Alternatively,
         # print("Percentage of patience with heart problems: "+str(y.where(y==1).count()*100/303))
         # print("Percentage of patience with heart problems: "+str(y.where(y==0).count()*100/303))

         # #Or,
         # countNoDisease = len(df[df.target == 0])
         # countHaveDisease = len(df[df.target == 1])

Percentage of patience without Cardiovascular Heart Problems within 10 years: 54.0
Percentage of patience with Cardiovascular Heart Problems within 10 years: 46.0
```

The following is a cell that we implemented separately to indicate the count from 100 and to show the percentage of each diagnosed patient.

### 3.7 Image Analysis

**7. Image Analysis**

**An Image Based on analysing the Cardiac Disease of the Medical Image number 2464 of a Patient**




The figure below represents a neural network technique being used to detect the detection of a heart before and after it has been diagnosed. It is also possible to load the segmented image.

### 3.8 Machine Learning Section (Importing the LR Algorithm)

Machine learning techniques are used to identify the pre-processed data. The data collection includes 17 attributes and a detcted value. The identification of dependent variables is the foundation of the ML model. Because the goal variable is categorical, it employed binary logistic regression, which is one of the classification algorithms.

#### 3.8.1 Separating the Dataset into Feature and Target Data

**8. Machine Learning Section (Importing the LR Algorithm)**

**Seperating the Dataset into Feature and Target Data**

In [91]:
```
X = df.iloc[:,0:17]
y = df.iloc[:,17:18]
```

In [92]:
```
X.head()
```

Out[92]:

| | Medical Image No | sex | age | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | prevalentHyp | diabetes | FamCheckUp | totChol | troponinLevel | restECG | sysBP | dia |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2396 | 1 | 30 | 1 | 13 | 0 | 1 | 1 | 1 | 1 | 177 | 7.0 | 81 | 132 | |
| 1 | 2523 | 1 | 39 | 0 | 12 | 0 | 0 | 1 | 1 | 0 | 166 | 9.0 | 180 | 128 | |
| 2 | 2317 | 1 | 69 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 237 | 11.0 | 83 | 124 | |
| 3 | 2524 | 0 | 31 | 0 | 5 | 0 | 0 | 1 | 1 | 1 | 208 | 6.0 | 58 | 122 | |
| 4 | 2867 | 0 | 44 | 0 | 13 | 0 | 0 | 1 | 0 | 1 | 180 | 13.0 | 88 | 122 | |

In [93]:
```
y.head()
```

Out[93]:

| | TenYearCVDTarget |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 1 |
| 3 | 0 |
| 4 | 0 |

Here the iloc method is used to designate rows and columns by their integer index, such as separating the Target from the other data columns.

#### 3.8.2 Assigning the Data for Training and Testing

**Assigning the Data for Training and Testing**

In [94]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.5, random_state=21)
```

Data set was separated into training and testing sets for evaluation process. This has been done using scikit- learn library.

### 3.8.3 Applying the ML Model – Logistic Regression

**Applying the ML Model - Logistic Regression**

```
In [95]: from sklearn.linear_model import LogisticRegression
         logreg = LogisticRegression()
```

Next the LR theory is been applied accordingly with importing the Logistic theory as seen above. Logreg is a variable is assigned to the Logistic Regression theory which is imported in the first line.

### 3.8.4 Training the Dataset

**Training the Dataset**

```
In [96]: logreg.fit(X_train, y_train)
```

```
C:\Users\rasha\anaconda3\anaconda\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y wa
s passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
C:\Users\rasha\anaconda3\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Out[96]: LogisticRegression()
```

The dataset and images are being trained with the LR theory so that it can detect the CVD Disease outcome accurately.

### 3.8.5 Testing the Dataset

**Testing the Dataset**

```
In [97]: y_pred = logreg.predict(X_test)
```

Test data is used to assess the accuracy and efficiency of the algorithm used to train the machine - specifically, how well it can detect new answers based on its previous training.

### 3.9 Model Evaluation

Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data.

**9. Model Evaluation**

**Accuracy of the Logistic Regression Model**

```
In [135]: score = logreg.score(X_test, y_test)
          print("Prediction Accuracy Score is: {:.2f}%".format(score*100))

          Prediction Accuracy Score is: 55.00%
```

The most essential, accuracy describes how well a model can classify data. It is the number of right guesses as a percentage of all detections. To compute the accuracy of a classification task, we use the sklearn package. In our situation, the forecasts of the diagnosis are almost correct because more than half of them are correct (55%).

### 3.9.1 Confusion Matrix

**Confusion Matrix**

```
In [99]: from sklearn.metrics import confusion_matrix, classification_report
         cm = confusion_matrix(y_test, y_pred)
         print("Confusion Matrix is:\n",cm)

         Confusion Matrix is:
          [[37 16]
          [29 18]]
```

A Confusion matrix is a N x N matrix that is used to assess the effectiveness of a classification model, where N is the number of target classes. The matrix compares the actual target values to the detections of the machine learning model. The rows represent the target variable's expected values as seen above.

### 3.9.2 Classification Report

**Classification Report**

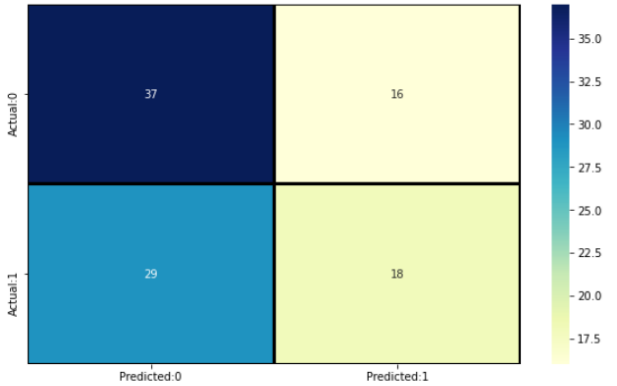
```
In [100]: print("Classification Report is:\n\n",classification_report(y_test,y_pred))

          Classification Report is:

                        precision    recall  f1-score   support

                     0       0.56      0.70      0.62        53
                     1       0.53      0.38      0.44        47

              accuracy                           0.55       100
             macro avg       0.55      0.54      0.53       100
          weighted avg       0.55      0.55      0.54       100
```

The Classification report here is used to measure the quality of predictions from the classification algorithm. The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives.

### 3.9.3 Plot of the Confusion Matrix

**Plot of the Confusion Matrix**

In [101]:
```
conf_matrix = pd.DataFrame(data = cm,
                           columns = ['Predicted:0', 'Predicted:1'],
                           index =['Actual:0', 'Actual:1'])
plt.figure(figsize = (10, 6))
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "YlGnBu", linecolor="Black", linewidths=1.5)
plt.show()
```




The Confusion matrix shows 38+18 = 56 Correct Predictions and 29+15 = 44 Incorrect Ones.

*1. True Positives: 18*

*2. True Negatives: 38*

*3. False Positives:15*

*4. False Negatives:29*

On the confusion matrix plot, the rows correspond to the predicted class (Output Class) and the columns correspond to the true class (Target Class). In a classification problem, this has been used to convey a summary of prediction results, including correct and wrong predictions. Furthermore, this was applied not only to faults but also to different categories of errors. The following parameters are shown by the segments of the confusion matrix.

- True Positives (TP): cases which are predicted yes (they have the disease), and they do have the disease.
- True Negatives (TN): cases which are predicted no, and they do not have the disease.
- False Positives (FP): cases which are predicted yes, but they do not actually have the disease (Type I error).
- False Negatives (FN): cases which are predicted no, but they actually do have the disease (Type II error).

According to the outcome of the confusion matrix, Correct predictions (38+18) =56, Incorrect predictions (29+15) =44

Therefore,

- True Positives: 18
- True Negatives: 38
- False Positives: 15(Type I error)
- False Negatives: 29(Type II error)

### 3.10 Overall Model Evaluation

This is a list of rates that are often computed from a confusion matrix for a binary classifier:

**10. Overall Model Evaluation**

**Sensitivity and Specificity**

In [102]:
```
TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
```

In [103]:
```
print(' \n The acuuracy of the model = TP+TN/(TP+TN+FP+FN) = ',(TP+TN)/float(TP+TN+FP+FN),'\n\n',

'The Missclassification = 1-Accuracy = ',1-((TP+TN)/float(TP+TN+FP+FN)),'\n\n',

'Sensitivity or True Positive Rate = TP/(TP+FN) = ',TP/float(TP+FN),'\n\n',

'Specificity or True Negative Rate = TN/(TN+FP) = ',TN/float(TN+FP),'\n\n',

'Positive Predictive value = TP/(TP+FP) = ',TP/float(TP+FP),'\n\n',

'Negative predictive Value = TN/(TN+FN) = ',TN/float(TN+FN),'\n\n',

'Positive Likelihood Ratio = Sensitivity/(1-Specificity) = ',sensitivity/(1-specificity),'\n\n',

'Negative likelihood Ratio = (1-Sensitivity)/Specificity = ',(1-sensitivity)/specificity)
```

```
The acuuracy of the model = TP+TN/(TP+TN+FP+FN) =  0.55

The Missclassification = 1-Accuracy =  0.44999999999999996

Sensitivity or True Positive Rate = TP/(TP+FN) =  0.3829787234042553

Specificity or True Negative Rate = TN/(TN+FP) =  0.6981132075471698

Positive Predictive value = TP/(TP+FP) =  0.5294117647058824

Negative predictive Value = TN/(TN+FN) =  0.5606060606060606

Positive Likelihood Ratio = Sensitivity/(1-Specificity) =  1.2686170212765955

Negative likelihood Ratio = (1-Sensitivity)/Specificity =  0.8838412880966073
```

It has been checked the accuracy of the model using confusion matrix. With analyzing confusion matrix data, it is evident that the model is highly specific than sensitive. Further, the negative values in the model are predicted more accurately than the positives.

| Terms | Formula |
|---|---|
| Accuracy of the model (overall, how often the classifer correct) | (TP+TN)/(TP+TN+FP+FN) |
| Misclassification Rate (overall, how often it wrong or error rate) | (FP+FN)/(TP+TN+FP+FN) |
| Sensitivity or True Positive Rate (when it is actually yes, how often does it predict yes) | TP/(TP+FN) |
| Specificity or True Negative Rate (when it is actually no, how often does it predict no) | TN/(TN+FP) |

**[Table 1 –Formulas of the above calculations.]**

### 3.10.1 Predicted probabilities of No CVD Diseases (0) and CVD Heart Disease (1) – Threshold of 0.5

0 (CVD Heart Disease: No) and 1 (CVD Heart Disease: Yes) for the test data with a default classification threshold of 0.5.

**Predicted probabilities of No CVD Heart Diseases (0) and CVD Heart Diseases (1) - Threshold of 0.5**

In [104]:
```
y_pred_prob=logreg.predict_proba(X_test)[:,:]
y_pred_prob_df=pd.DataFrame(data=y_pred_prob, columns=['Prob of no heart disease (0)','Prob of Heart Disease (1)'])
y_pred_prob_df.head()
```

Out[104]:

| | Prob of no heart disease (0) | Prob of Heart Disease (1) |
|---|---|---|
| 0 | 0.650599 | 0.349401 |
| 1 | 0.694272 | 0.305728 |
| 2 | 0.692216 | 0.307784 |
| 3 | 0.735303 | 0.264697 |
| 4 | 0.690265 | 0.309735 |

### 3.10.2 Lowering the Threshold

According to the confusion matrix, if the number of False Negatives (FN) (Type II mistake) is very high, it can be rated as moderately risky because it involves ignoring the possibility of sickness when there is just one OR True. As a result, the threshold can be reduced to boost sensitivity.

**Lowering the threshold**

In [105]:
```
from sklearn.preprocessing import binarize
for i in range(1,5):
    cm2=0
    y_pred_prob_yes=logreg.predict_proba(X_test)
    y_pred2=binarize(y_pred_prob_yes,i/10)[:,1]
    cm2=confusion_matrix(y_test,y_pred2)
    print ('With',i/10,'threshold the Confusion Matrix is ','\n',cm2,'\n',
            'with',cm2[0,0]+cm2[1,1],'correct predictions and',cm2[1,0],'Type II errors( False Negatives)','\n\n',
          'Sensitivity: ',cm2[1,1]/(float(cm2[1,1]+cm2[1,0])),'Specificity: ',cm2[0,0]/(float(cm2[0,0]+cm2[0,1])),'\n\n\n')
```

```
With 0.1 threshold the Confusion Matrix is
 [[ 0 53]
 [ 0 47]]
 with 47 correct predictions and 0 Type II errors( False Negatives)

 Sensitivity:  1.0 Specificity:  0.0


With 0.2 threshold the Confusion Matrix is
 [[ 1 52]
 [ 0 47]]
 with 48 correct predictions and 0 Type II errors( False Negatives)

 Sensitivity:  1.0 Specificity:  0.018867924528301886


With 0.3 threshold the Confusion Matrix is
 [[11 42]
 [ 5 42]]
 with 53 correct predictions and 5 Type II errors( False Negatives)

 Sensitivity:  0.8936170212765957 Specificity:  0.20754716981132076


With 0.4 threshold the Confusion Matrix is
 [[24 29]
 [20 27]]
 with 51 correct predictions and 20 Type II errors( False Negatives)

 Sensitivity:  0.574468085106383 Specificity:  0.452830188679245
```

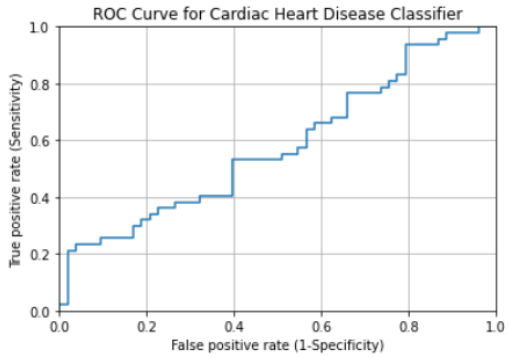
```
C:\Users\rasha\anaconda3\anaconda\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass threshold=0.1 as keyword args. From version 1.0 (renaming of 0.25) passing these as positional arguments will result in an error
  warnings.warn(f"Pass {args_msg} as keyword args. From version "
C:\Users\rasha\anaconda3\anaconda\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass threshold=0.2 as keyword args. From version 1.0 (renaming of 0.25) passing these as positional arguments will result in an error
  warnings.warn(f"Pass {args_msg} as keyword args. From version "
C:\Users\rasha\anaconda3\anaconda\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass threshold=0.3 as keyword args. From version 1.0 (renaming of 0.25) passing these as positional arguments will result in an error
  warnings.warn(f"Pass {args_msg} as keyword args. From version "
C:\Users\rasha\anaconda3\anaconda\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass threshold=0.4 as keyword args. From version 1.0 (renaming of 0.25) passing these as positional arguments will result in an error
  warnings.warn(f"Pass {args_msg} as keyword args. From version "
```

### 3.10.3 Graph on the ROC Curve

The ROC Curve is a straightforward figure that is used to depict the performance of a binary classifier. Furthermore, this illustrates the tradeoff between a classifier's true positive rate and false positive rate for various probability threshold settings.

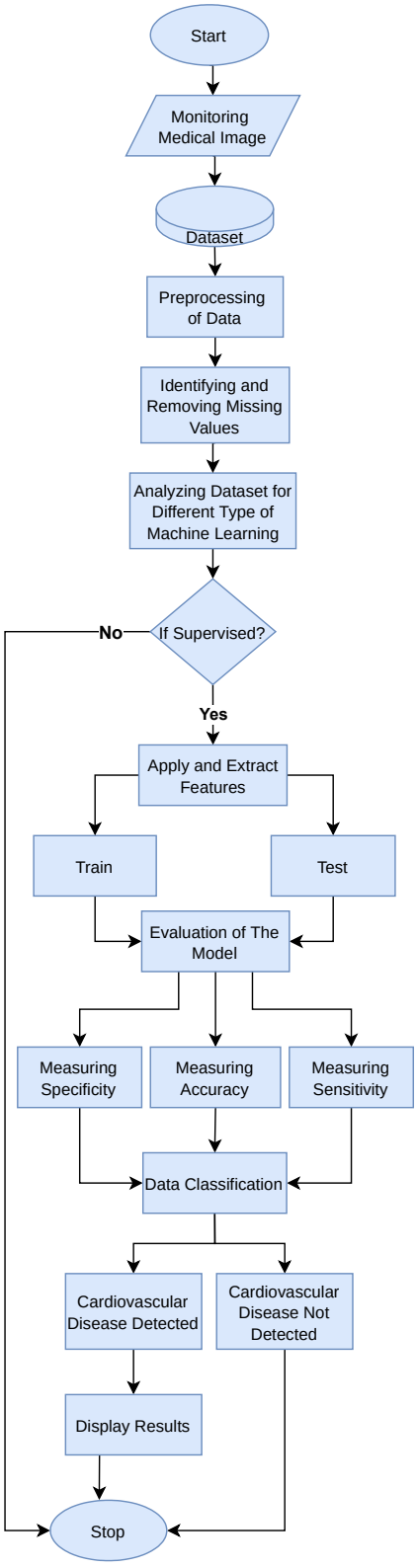
**Graph on the ROC Curve**

In [106]:
```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_yes[:,1])
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC Curve for Cardiac Heart Disease Classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)
```




At all thresholds, good classification accuracy models should have much more true positives than false positives.

## The Flowchart



# 4. Performance Evaluation

Dataset that were used consist of Numerical Values and Images that were taken to extract and be trained and tested the set of Dataset was found from the link https://archive.ics.uci.edu/ml/datasets/Heart+Disease And moreover, the below includes the nature, count and all the other information of the Dataset also the same set of data were used for splitting data for training and testing purposes.

- Shape/Count of the Dataset

```
(200, 18)
```

- Key Values of the Dataset

```
Index(['Medical Image No', 'sex', 'age', 'currentSmoker', 'cigsPerDay',
       'BPMeds', 'prevalentStroke', 'prevalentHyp', 'diabetes', 'FamCheckUp',
       'totChol', 'troponinLevel', 'restECG', 'sysBP', 'diaBP', 'BMI',
       'heartRate', 'TenYearCVDTarget'],
      dtype='object')
```

- Info of the Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Medical Image No  200 non-null    int64
 1   sex               200 non-null    int64
 2   age               200 non-null    int64
 3   currentSmoker     200 non-null    int64
 4   cigsPerDay        200 non-null    int64
 5   BPMeds            200 non-null    int64
 6   prevalentStroke   200 non-null    int64
 7   prevalentHyp      200 non-null    int64
 8   diabetes          200 non-null    int64
 9   FamCheckUp        200 non-null    int64
 10  totChol           200 non-null    int64
 11  troponinLevel     200 non-null    float64
 12  restECG           200 non-null    int64
 13  sysBP             200 non-null    int64
 14  diaBP             200 non-null    int64
 15  BMI               200 non-null    float64
 16  heartRate         200 non-null    int64
 17  TenYearCVDTarget  200 non-null    int64
dtypes: float64(2), int64(16)
memory usage: 28.2 KB
```

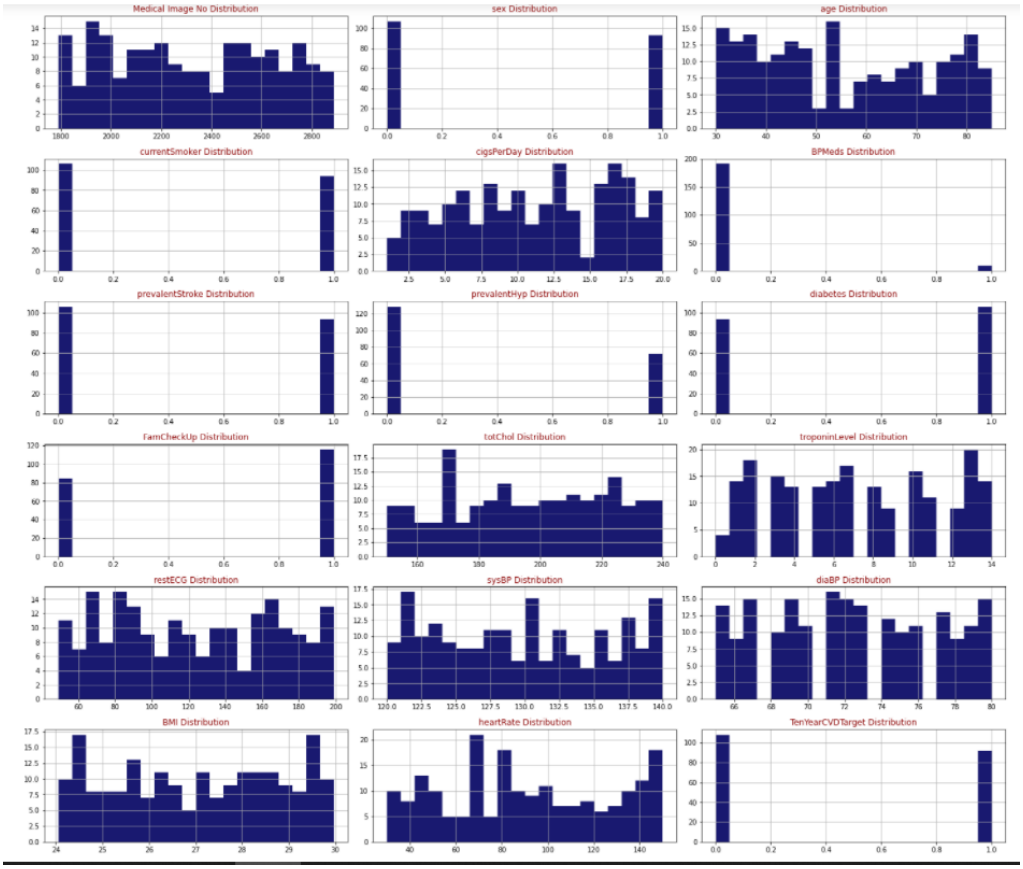
- Missing Values of the Dataset

```
Medical Image No    0
sex                 0
age                 0
currentSmoker       0
cigsPerDay          0
BPMeds              0
prevalentStroke     0
prevalentHyp        0
diabetes            0
FamCheckUp          0
totChol             0
troponinLevel       0
restECG             0
sysBP               0
diaBP               0
BMI                 0
heartRate           0
TenYearCVDTarget    0
dtype: int64
```

- Correlation between Columns

```
TenYearCVDTarget     1.000000
restECG              0.140881
heartRate            0.125340
troponinLevel        0.121955
BMI                  0.067816
diaBP                0.061140
sysBP                0.047354
prevalentStroke      0.045025
prevalentHyp         0.044309
BPMeds               0.041618
age                  0.037177
totChol              0.029998
sex                  0.024539
cigsPerDay           0.014153
Medical Image No     0.007650
FamCheckUp           0.007317
currentSmoker        0.004824
diabetes             0.004824
Name: TenYearCVDTarget, dtype: float64
```

- Distributions of the Dataset




In terms of comparing other models to Logistic Regression, we believe that LR is the best practical sort of Machine Learning Approach for the data set and images we provided for diagnosing purposes. Logistic Regression, as previously described, is a Machine Learning classification technique used to estimate the likelihood of a categorical dependent variable. It is a classification problem extension of the linear regression model. Unlike linear regression, which produces continuous numerical values, logistic regression produces a probability value that may be mapped to two or more discrete classes using the logistic sigmoid function. Below is a comparison between the Logistic Regression theory and Support Vector Machine for the same set of data and images.

## 4.1 Logistic Regression Comparison

**Assigning the Data for Training and Testing**

In [42]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.5, random_state=21)
```

**Applying the ML Model - Logistic Regression**

In [56]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

**Training the Dataset**

In [44]:
```
logreg.fit(X_train, y_train)
```

```
C:\Users\rasha\anaconda3\anaconda\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
C:\Users\rasha\anaconda3\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[44]:
```
LogisticRegression()
```

**Testing the Dataset**

In [45]:
```
y_pred = logreg.predict(X_test)
```

The image above is verification that the data and images were trained and tested using the Logistic Regression theory, which employs sigmoid functions and other techniques.

**Accuracy of the Logistic Regression Model**

In [66]:
```
score = sv.score(X_test, y_test)
print("Prediction Accuracy Score is: {:.2f}%".format(score*100))
```

```
Prediction Accuracy Score is: 53.00%
```

Finally, we can observe that with the Logistic Regression Theory, the accuracy score increases to 53%, indicating that half of the CVD predictions based on the LR theory are right.

## 4.2 Support Vector Machine Comparison

**Assigning the Data for Training and Testing**

In [39]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.5, random_state=21)
```

**Applying the ML Model - SVM**

In [55]:
```
from sklearn import svm
sv = svm.SVC()
```

**Training the Dataset**

In [56]:
```
sv.fit(X_train, y_train)
```

```
C:\Users\rasha\anaconda3\anaconda\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y wa
s passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

Out[56]:
```
SVC()
```

**Testing the Dataset**

In [59]:
```
y_pred_svm = sv.predict(X_test)
```

SVMs, as a first approximation, find a separation line (or hyperplane) between data of two classes. SVM is an algorithm that takes data as input and generates a line that, if possible, separates those classes. The image above is verification that the data and images were trained and tested using the Support Vector Machine. However, with the SVM Theory, the accuracy score drops to 33%, showing that half of the CVD predictions based on the LR theory are incorrect.

**Accuracy of the Support Vector Machine Model**

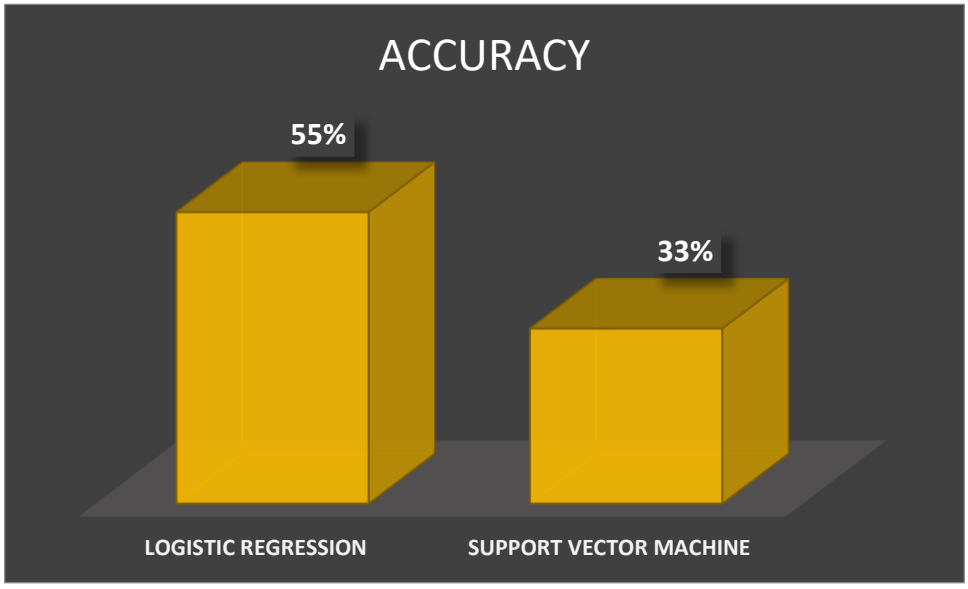
In [70]:
```
score = sv.score(X_test, y_test)
print("Prediction Accuracy Score is: {:.2f}%".format(score*100))
```

```
Prediction Accuracy Score is: 33.00%
```

However, with the SVM Theory, the accuracy score drops to 33%, showing that half of the CVD predictions based on the LR theory are incorrect.

**Demonstration of the Findings**

The SVM Machine Learning approach is used here to modify the theory with the same set of data is because linear SVMs and logistic regression often perform similarly in practice. The logistic regression is derived from extended linear regression. The Support Vector Machines algorithm is considerably more geometrically motivated. However, because SVM works well with unstructured and semi-structured data such as text and images, and logistic regression works with previously recognized independent variables, the LR Theory predictions are higher than the SVM. As a result, I believe the LR method is the most suitable Machine Learning approach to be taken when implementing heart diagnosing problems.




| | Precision | Recall |
|---|---|---|
| Logistic Regression | 0.56 | 0.70 |
| Support Vector Machine | 0.46 | 0.57 |

# 5. Conclusion

Nowadays most of the data is computerized, the data is distributed everywhere but we're not utilizing it properly. By Analyzing the available data, we can also use for unknown patterns. The motive of this future work is to predict heart diseases with high rate of accuracy by using the Classification Algorithms of Machine Learning. For predicting the heart disease with the help of different parameters, we can use Logistic Regression, Support Vector Machine, KNN, Decision Tree, naviebayes, sklearn in machine learning Algorithm. Moreover, the model could be improved by using more data and techniques. The future scope of the paper is the prediction of heart diseases by using advanced techniques, with a high rate of accuracy and algorithms in less time complexity.

The primary goal of this study is to assess the accuracy of the classification algorithm in predicting the risk of 10-year CVD using the Medical Image Analysis and set of data including patient's data. We discovered that Logistic Regression is a better appropriate algorithm to predict the risk. Following the feature extraction process, the following attributes are chosen based on P values less than 5%. The primary goal of this study is to predict cardiac disease with high accuracy by preprocessing all with all the methods. Furthermore, the Logistic Regression model has the highest accuracy of all algorithms at 55.00% for this following dataset.

Finally, through the implementation of the code, all of our team members gained so much knowledge through the Jupyter Lab, such as coming to the libraries, where we learned about many new libraries that we had not previously learned about and particularly what each of them does. We also learnt new concepts on the Python Jupyterlab, such as what head does, dropping, iloc, describing, making samples, inserting images, showing numerical statics, and all of the others, which allowed team members obtain a better level of knowledge and work very effectively in the future. We also made many mistakes, such as unknowingly assigning variables to different areas and not successfully importing Tensorflow, and somehow as a team we were all able to work together and bring solutions to all the errors that popped in us throughout the competition, which and as a fact now we know very well on considering the small part to be the hardest as which will benefit us a lot in future work.

# 6. References


1. Mozaffarian, D., Benjamin, E. J., Go, A. S., Arnett, D. K., Blaha, M. J., Cushman, M., de Ferranti, S., Després, J.-P., Fullerton, H. J., Howard, V. J., Huffman, M. D., Judd, S. E., Kissela, B. M., Lackland, D. T., Lichtman, J. H., Lisabeth, L. D., Liu, S., Mackey, R. H., Matchar, D. B., & McGuire, D. K. (2016). Heart Disease and Stroke Statistics—2015 Update. *Circulation*, *131*(4). Retrieved on 12th September, 2021, from https://doi.org/10.1161/cir.0000000000000152

2. *Medwin Publishers*. (2016). Medwinpublishers.com. Retrieved on 13th September, 2021, from https://medwinpublishers.com/JOBD/

3. Abduljabbar, R., Dia, H., Liyanage, S., & Bagloee, S. A. (2019). Applications of Artificial Intelligence in Transport: An Overview. *Sustainability*, *11*(1), 189. Retrieved on 14th September 2021, from https://doi.org/10.3390/su11010189

4. S. T. Nishadi. (2019). *Predicting Heart Diseases In Logistic Regression Of Machine Learning Algorithms By Python Jupyterlab*. Retrieved on 13th September, 2021, from https://www.semanticscholar.org/paper/Predicting-Heart-Diseases-In-Logistic-Regression-Of-Nishadi/89028628526e5ffd8bedd6ea126a9de2423c104d

5. Ramakrishnan, N., Safith, S., Sonya, A., Kavitha, G., & Rahman, B. (n.d.). *JOURNAL OF CRITICAL REVIEWS A COMPREHENSIVE REVIEW ON PREDICTION OF HEART CONDITION USING MACHINE LEARNING ALGORITHM*. Retrieved on 16th September, 2021, form https://www.bibliomed.org/mnsfulltext/197/197-1597144194.pdf?1631796160


# 7. Appendix

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import scipy.stats as st
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from matplotlib import pyplot
from matplotlib.image import imread

df = pd.read_csv('Cardiovascular Disease Prediction Dataset.csv')
df.head()

src = 'Medical Image No 2396.jpg'
print (src)
photo = load_img(src, target_size=(200, 200))
pyplot.imshow(photo)
print(photo.size)

src = 'Medical Image No 2523.jpg'
print (src)
photo = load_img(src, target_size=(200, 200))
pyplot.imshow(photo)
print(photo.size)
```

```
src = 'Medical Image No 2317.jpg'
print (src)
photo = load_img(src, target_size=(200, 200))
pyplot.imshow(photo)
print(photo.size)

src = 'Medical Image No 2524.jpg'
print (src)
photo = load_img(src, target_size=(200, 200))
pyplot.imshow(photo)
print(photo.size)

src = 'Medical Image No 2867.jpg'
print (src)
photo = load_img(src, target_size=(200, 200))
pyplot.imshow(photo)
print(photo.size)

df.sample(2)

df.shape

df.keys()

df.info()

df.describe()

info = ["Number representing the patients medical image","0: Male, 1: Female","Age of the patient","0: No, 1: Yes"," The Amount of Cigs
taken in a day","0: Not taking BP Meds, 1: Taking BP Meds"," 0: Not Had Stroke, 1: Had Stroke","0: No Prevalent Strokes, 1: Have
Prevalent Strokes","0: No Diabetes, 1: Diabetes","0: Not Had Family Check Up, 1: Family Check Up","Total Number of
Cholestral","Number of Troponin Level"]
for i in range(len(info)):
   print(df.columns[i]+":\t\t\t"+info[i])

df.isna().sum()

count=0
for i in df.isnull().sum(axis=1):
   if i>0:
      count=count+1
print('The Total Number of Rows With Missing Values is ', count)
print('Since it is only',round((count/len(df.index))*100), 'percent of the entire dataset the rows with missing values are excluded.')

df.dropna(axis = 0, inplace = True)
print(df.shape)

df["TenYearCVDTarget"].describe()

df['TenYearCVDTarget'].value_counts()

df["TenYearCVDTarget"].unique()

print(df.corr()["TenYearCVDTarget"].abs().sort_values(ascending=False))

from statsmodels.tools import add_constant as add_constant
df_constant = add_constant(df)
df_constant.head()

st.chisqprob = lambda chisq, df: st.chi2.sf(chisq, df)
cols=df_constant.columns[:2]
model=sm.Logit(df.TenYearCVDTarget,df_constant[cols])
result=model.fit()
result.summary()
```

```
def back_feature_elem (data_frame,dep_var,col_list):
    """ Takes in the dataframe, the dependent variable and a list of column names, runs the regression repeatedly eleminating feature
with the highest
    P-value above alpha one at a time and returns the regression summary with all p-values below alpha"""

    while len(col_list)>-1 :
        model=sm.Logit(dep_var,data_frame[col_list])
        result=model.fit(disp=0)
        largest_pvalue=round(result.pvalues,10).nlargest(5)
        if largest_pvalue[0]<(2):
            return result
            break
        else:
            col_list=col_list.drop(largest_pvalue.index)

result=back_feature_elem(df_constant,df.TenYearCVDTarget,cols)
result.summary()

params = np.exp(result.params)
conf = np.exp(result.conf_int())
conf['OR'] = params
pvalue=round(result.pvalues,3)
conf['pvalue']=pvalue
conf.columns = ['CI 95%(2.5%)', 'CI 95%(97.5%)', 'Odds Ratio','pvalue']
print ((conf))

def draw_histograms(dataframe, features, rows, cols):
    fig=plt.figure(figsize=(20,20))
    for i, feature in enumerate(features):
        ax=fig.add_subplot(rows,cols,i+1)
        dataframe[feature].hist(bins=20,ax=ax,facecolor='midnightblue')
        ax.set_title(feature+" Distribution",color='DarkRed')

    fig.tight_layout()
    plt.show()
draw_histograms(df,df.columns,7,3)

plt.figure(figsize = (14, 10))
sns.heatmap(df.corr(), cmap='Greens',annot=True, linecolor='Black', linewidths=1.0)
plt.show()

sns.countplot(x='TenYearCVDTarget',data=df)

sns.pairplot(df)
plt.show()

sns.catplot(data=df, kind='count', x='sex',hue='TenYearCVDTarget')
plt.show()

sns.catplot(data=df, kind='count', x='sex',hue='currentSmoker')
plt.show()


sns.catplot(data=df, kind='count', x='currentSmoker',hue='diabetes')
plt.show()

sns.catplot(data=df, kind='count', x='TenYearCVDTarget', col='sex',row='currentSmoker', palette='Reds')
plt.show()
```

```
y = df["TenYearCVDTarget"]
sns.countplot(y)
TenYearCVDTarget_temp = df.TenYearCVDTarget.value_counts()
print(TenYearCVDTarget_temp)

print("Percentage of patience without Cardiovascular Heart Problems within 10 years:
"+str(round(TenYearCVDTarget_temp[0]*100/303,2)))
print("Percentage of patience with Cardiovascular Heart Problems within 10 years: "+str(round(TenYearCVDTarget_temp[1]*100/303,2)))
#Alternatively,
# print("Percentage of patience with heart problems: "+str(y.where(y==1).count()*100/303))
# print("Percentage of patience with heart problems: "+str(y.where(y==0).count()*100/303))
# #Or,
# countNoDisease = len(df[df.target == 0])
# countHaveDisease = len(df[df.target == 1])

X = df.iloc[:,0:17]
y = df.iloc[:,17:18]

X.head()

y.head()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.5, random_state=21)

from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()

logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

score = logreg.score(X_test, y_test)
print("Prediction Accuracy Score is: {:.2f}%".format(score*100))

from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix is:\n",cm)

print("Classification Report is:\n\n",classification_report(y_test,y_pred))

conf_matrix = pd.DataFrame(data = cm,
                    columns = ['Predicted:0', 'Predicted:1'],
                    index =['Actual:0', 'Actual:1'])
plt.figure(figsize = (10, 6))
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "YlGnBu", linecolor="Black", linewidths=1.5)
plt.show()

TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)

print(' \n The acuuracy of the model = TP+TN/(TP+TN+FP+FN) = ',(TP+TN)/float(TP+TN+FP+FN),'\n\n',
'The Missclassification = 1-Accuracy = ',1-((TP+TN)/float(TP+TN+FP+FN)),'\n\n',
'Sensitivity or True Positive Rate = TP/(TP+FN) = ',TP/float(TP+FN),'\n\n',
'Specificity or True Negative Rate = TN/(TN+FP) = ',TN/float(TN+FP),'\n\n',
'Positive Predictive value = TP/(TP+FP) = ',TP/float(TP+FP),'\n\n',
'Negative predictive Value = TN/(TN+FN) = ',TN/float(TN+FN),'\n\n',
'Positive Likelihood Ratio = Sensitivity/(1-Specificity) = ',sensitivity/(1-specificity),'\n\n',
'Negative likelihood Ratio = (1-Sensitivity)/Specificity = ',(1-sensitivity)/specificity)
```

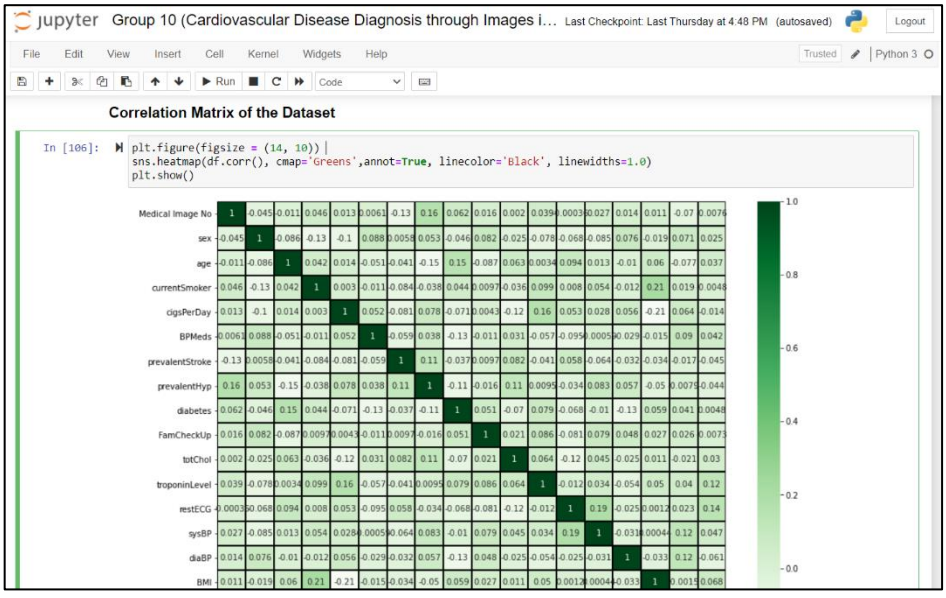
```
y_pred_prob=logreg.predict_proba(X_test)[:,:]
y_pred_prob_df=pd.DataFrame(data=y_pred_prob, columns=['Prob of no heart disease (0)','Prob of Heart Disease (1)'])
y_pred_prob_df.head()

from sklearn.preprocessing import binarize
for i in range(1,5):
  cm2=0
  y_pred_prob_yes=logreg.predict_proba(X_test)
  y_pred2=binarize(y_pred_prob_yes,i/10)[:,1]
  cm2=confusion_matrix(y_test,y_pred2)
  print ('With',i/10,'threshold the Confusion Matrix is ','\n',cm2,'\n',
       'with',cm2[0,0]+cm2[1,1],'correct predictions and',cm2[1,0],'Type II errors( False Negatives)','\n\n',
      'Sensitivity: ',cm2[1,1]/(float(cm2[1,1]+cm2[1,0])),'Specificity: ',cm2[0,0]/(float(cm2[0,0]+cm2[0,1])),'\n\n\n')

from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob_yes[:,1])
plt.plot(fpr,tpr)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC Curve for Cardiac Heart Disease Classifier')
plt.xlabel('False positive rate (1-Specificity)')
plt.ylabel('True positive rate (Sensitivity)')
plt.grid(True)
```

## The user Interface

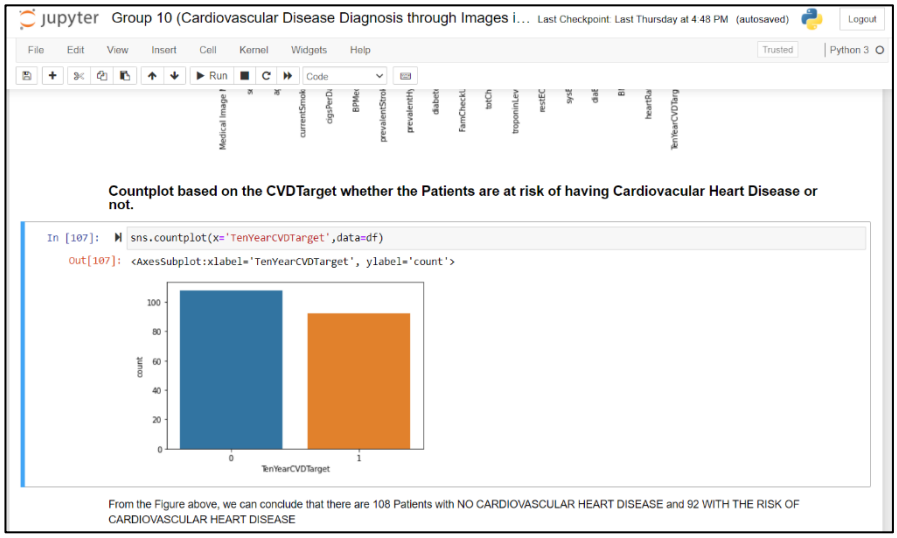
The interface we have been using throughout the additional code will be Python Jupyter Notebook which is an open-source web application, where the users have the access in generating codes and sharing with each other. The Computer language we taken to write the code is python because all of the teammates were having the knowledge of the language much. Jupyter Notebook and python language was chosen as the user interface because it is efficient, versatile, shared, and allows data visualization in the same context.

### Screenshots of The Different Steps in Addition to The Code

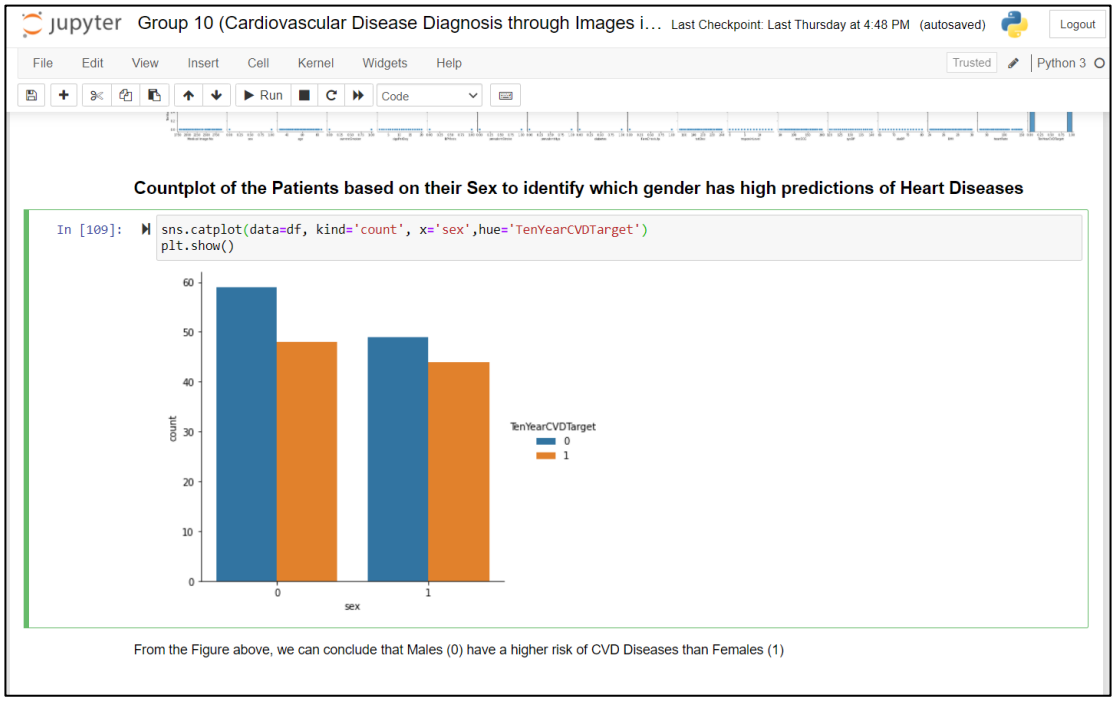
- **Correlation Matrix of The Dataset**



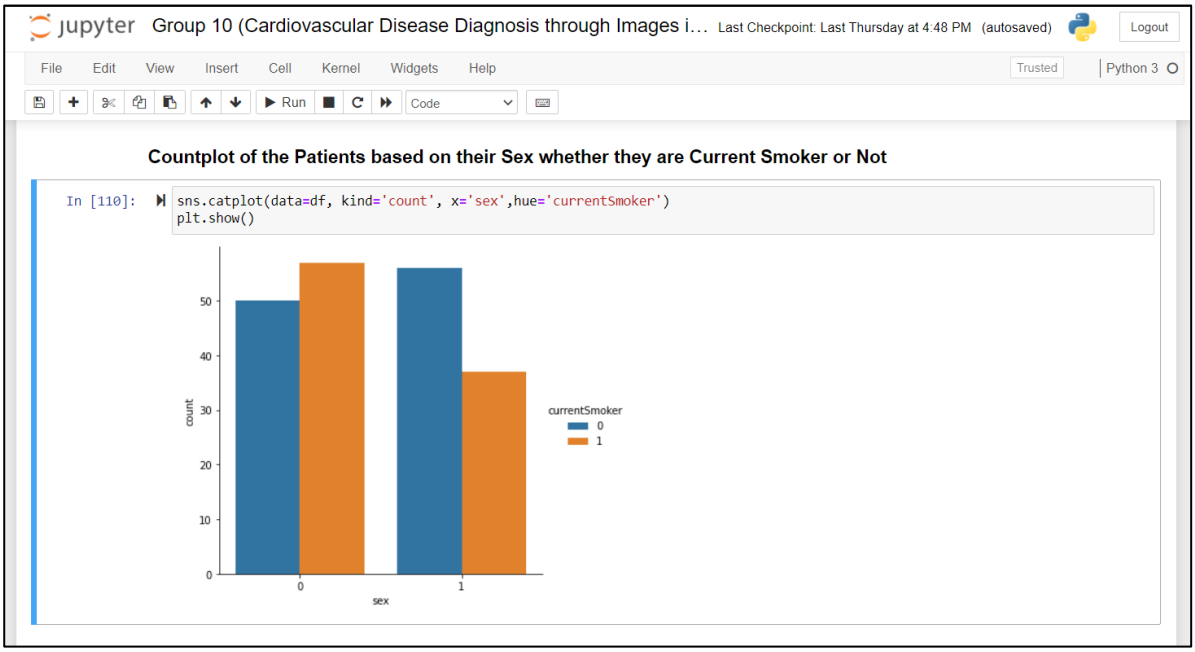
- **Countplots Based on The CVDTarget whether the Patients are at risk of having Cardiovacular Heart Disease or not.**



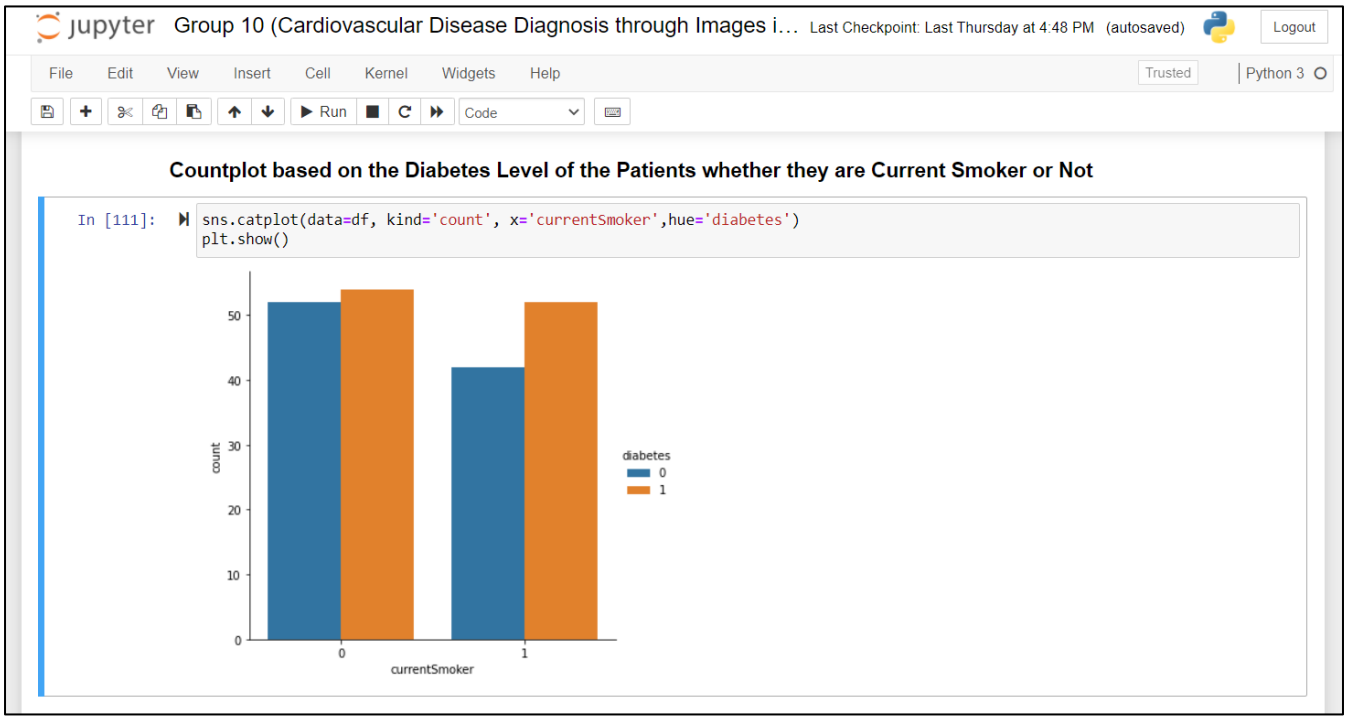

- **Countplot of the Patients based on their Sex to identify which gender has high predictions of heart diseases**



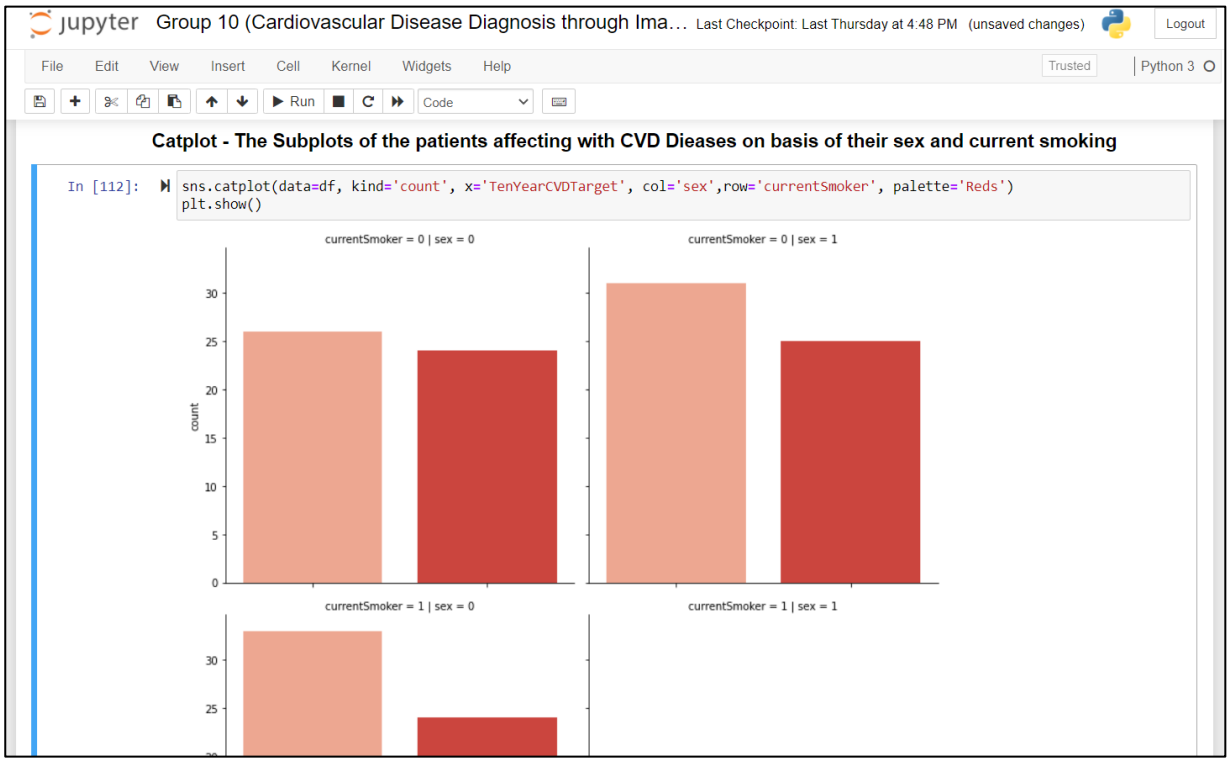
- **Countplot of the Patients based on their Sex whether they are Current Smoker or Not**



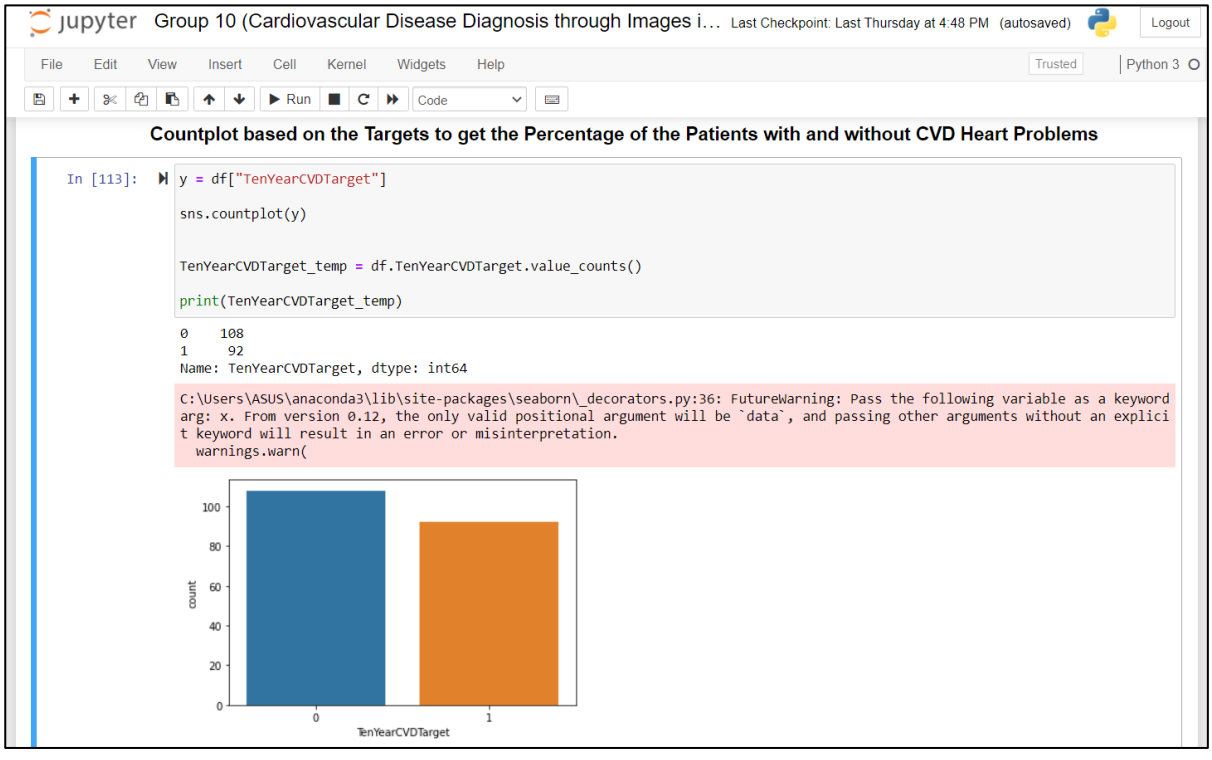

- **Countplot based on the Diabetes Level of the Patients whether they are Current Smoker or No**



- **Catplot - The Subplots of the patients affecting with CVD Diseases on basis of their sex and current smoking**




- **Countplot based on the Targets to get the Percentage of the Patients with and without CVD Heart Problems**




**- END OF THE PROJECT IMPLMENTATION REPORT -**