# Internship Project : Heart disease Prediction

**SUBMITTED BY:**

***NAME: PARTHIV SARMA***

***EMAIL ID: parthivk2@gmail.com***

***BATCH: ML WITH PYTHON***

***CERTIFICATION CODE: TCRIG02R68***

***GROUP: OWN***

***QUESTION:***

Please perform the below steps in a Google Colab or Jupyter Notebook as per your convenience:

I**n this assignment , we need to prediction of Heart disease detection**

**Features:**

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to

this date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4.

We will create a model with the following steps:

● Import the relevant packages

● Download and explore the dataset

● Prepare the dataset for training

● Use any prediction algorithm based upon the EDA

● Train the model to fit the data

● Make predictions using the trained model

● Create a test case and generate a predicted result from the system

**SOURCE CODE AND OUTPUTS:**

**from google.colab import drive**

**drive.mount('/content/drive')**

**# Commented out IPython magic to ensure Python compatibility.**

**#Importing the required Libraries and Dataset**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**# %matplotlib inline**

**from sklearn.model\_selection import train\_test\_split**

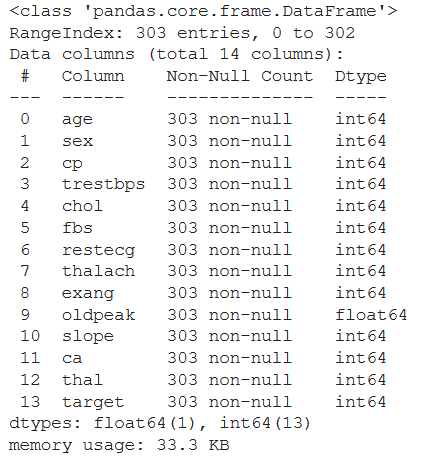
**from sklearn.preprocessing import StandardScaler**

**import warnings**

**warnings.filterwarnings('ignore')**

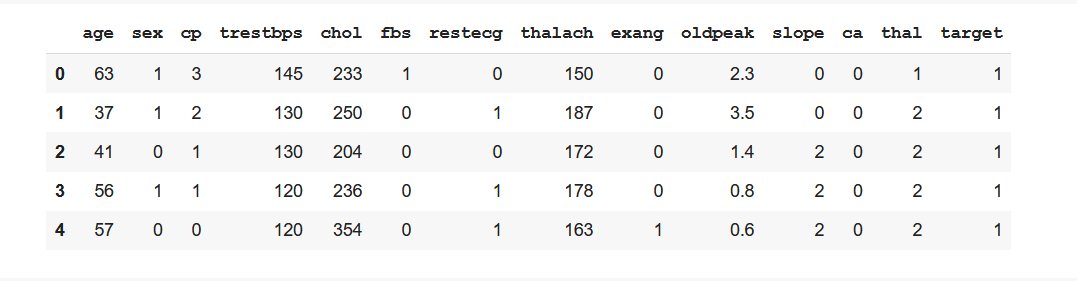
**heart\_df=pd.read\_csv('/content/drive/MyDrive/ML with PY/final project/heart.csv')**

**heart\_df.info()**

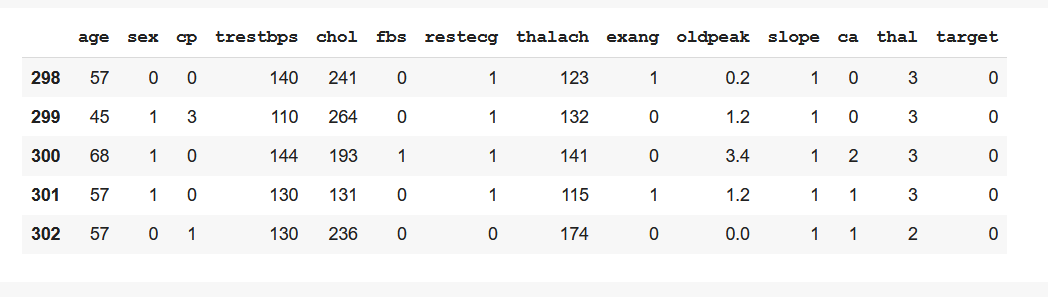


**#Exploring the dataset**

**heart\_df.head()**



**heart\_df.tail()**



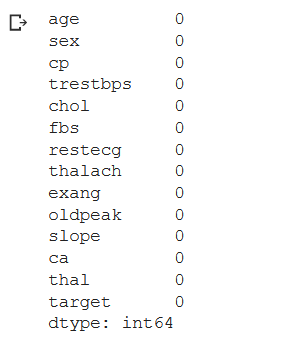
**heart\_df.\_len\_()**



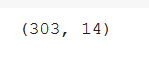
**heart\_df.columns.\_len\_()**



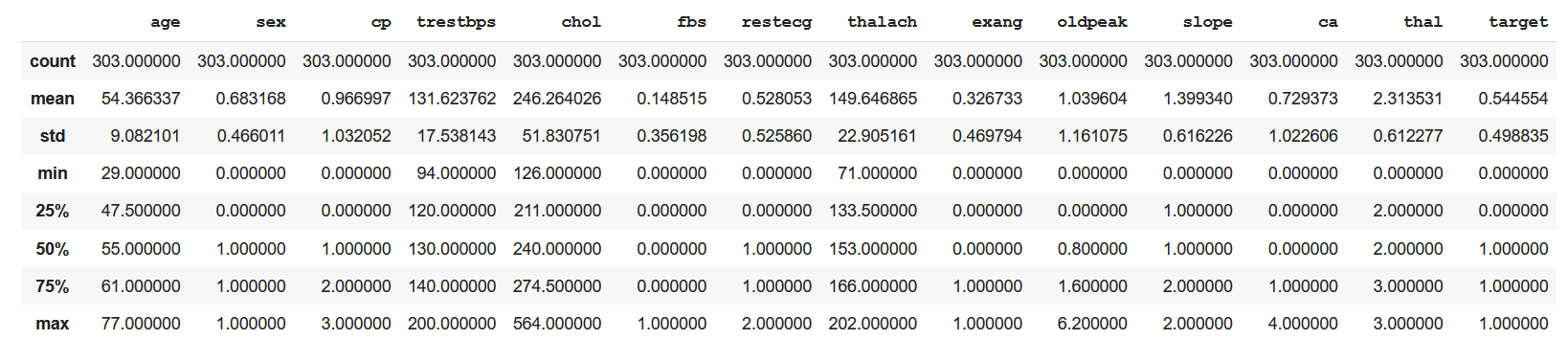
**heart\_df.isnull().sum()**



**heart\_df.shape**

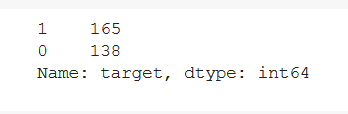


**heart\_df.describe()**

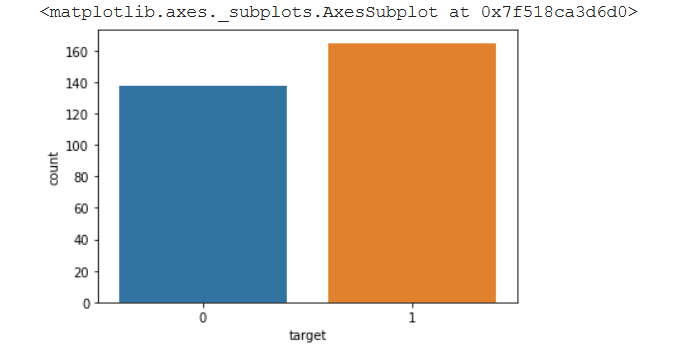


**#Exploratory Data Analysis**

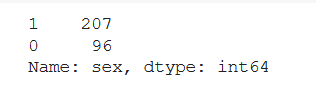
**heart\_df['target'].value\_counts()**



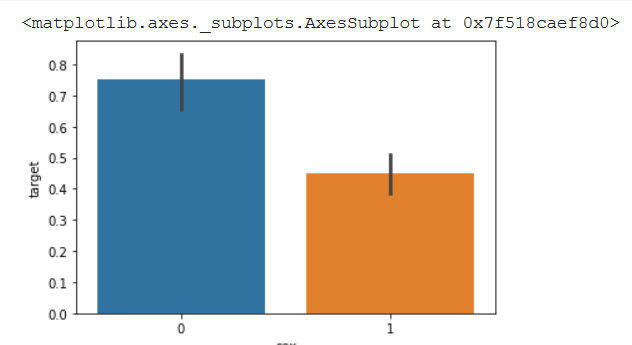
**sns.countplot(heart\_df['target'])**



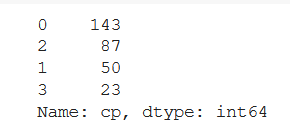
**heart\_df['sex'].value\_counts()**



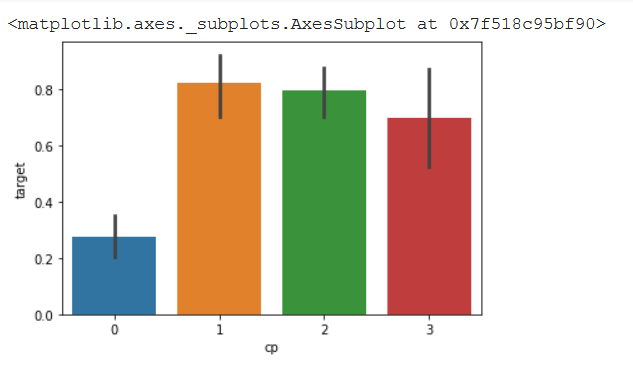
**sns.barplot(heart\_df["sex"],heart\_df["target"])**



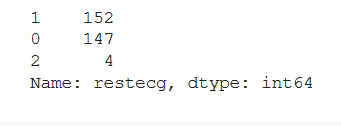
**heart\_df.cp.value\_counts()**



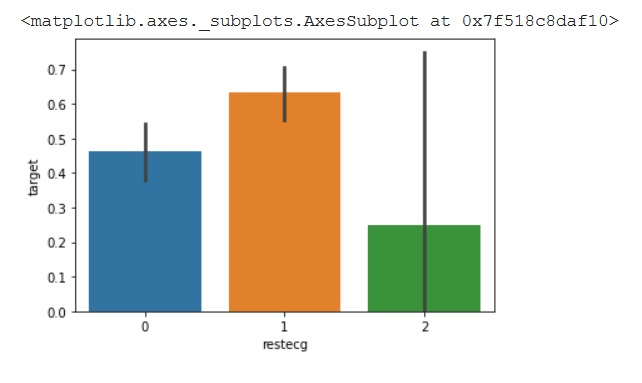
**sns.barplot(heart\_df["cp"],heart\_df['target'])**



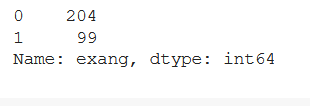
**heart\_df.restecg.value\_counts()**



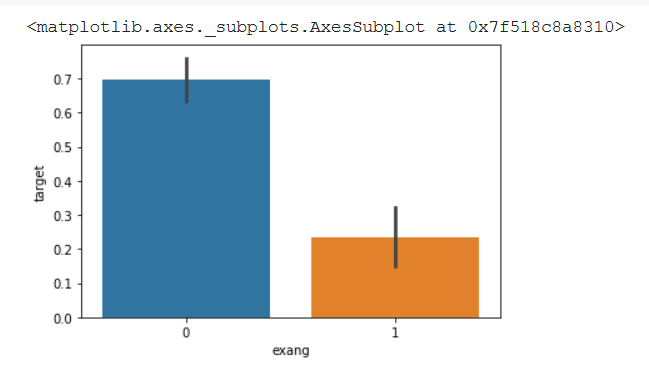
**sns.barplot(heart\_df["restecg"],heart\_df['target'])**



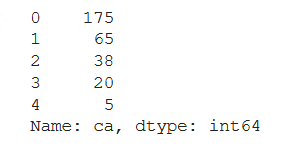
**heart\_df.exang.value\_counts()**



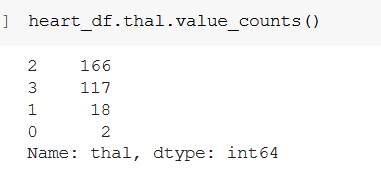
**sns.barplot(heart\_df["exang"],heart\_df['target'])**

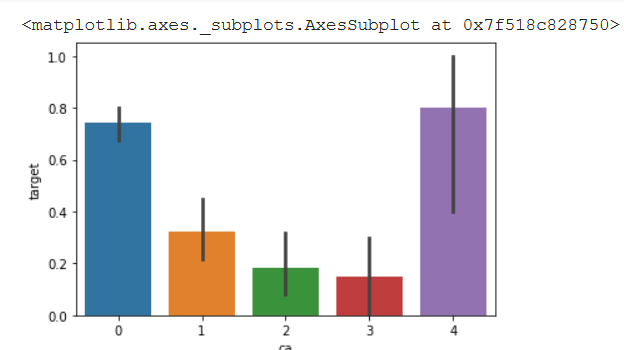


**heart\_df.ca.value\_counts()**

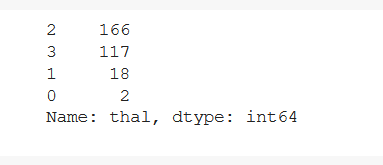


**sns.barplot(heart\_df["ca"],heart\_df['target'])**

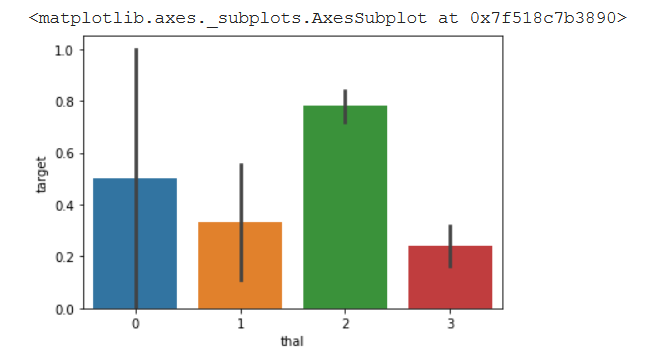




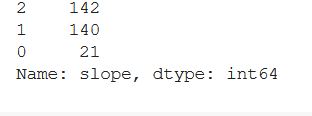
**heart\_df.thal.value\_counts()**



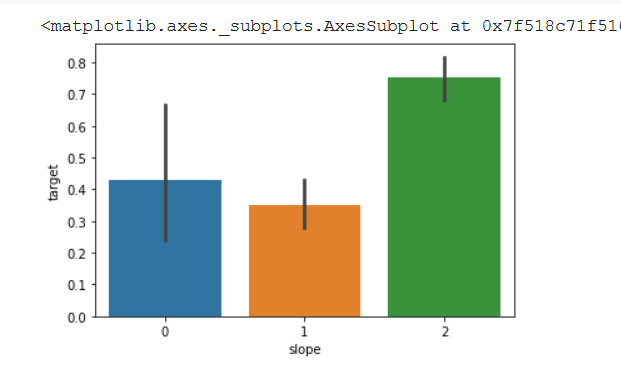
**sns.barplot(heart\_df["thal"],heart\_df['target'])**



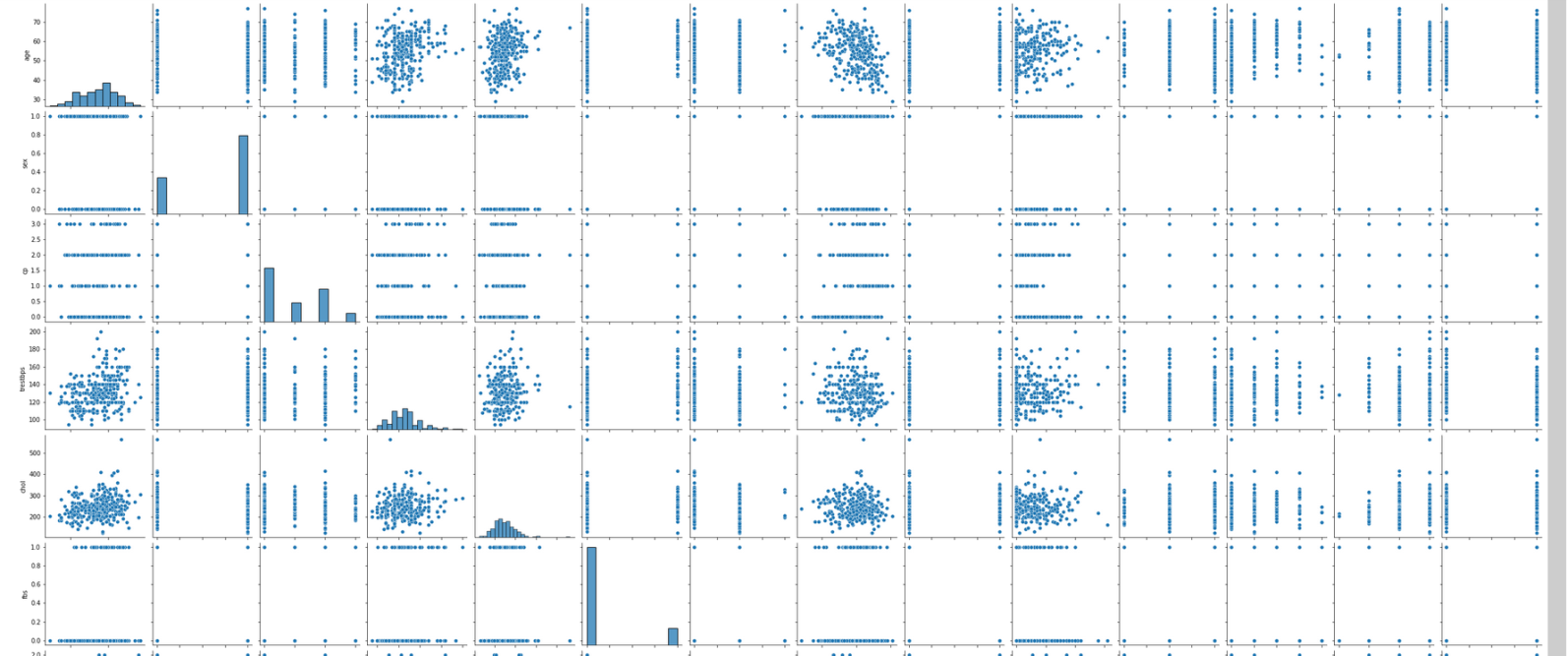
**heart\_df.slope.value\_counts()**

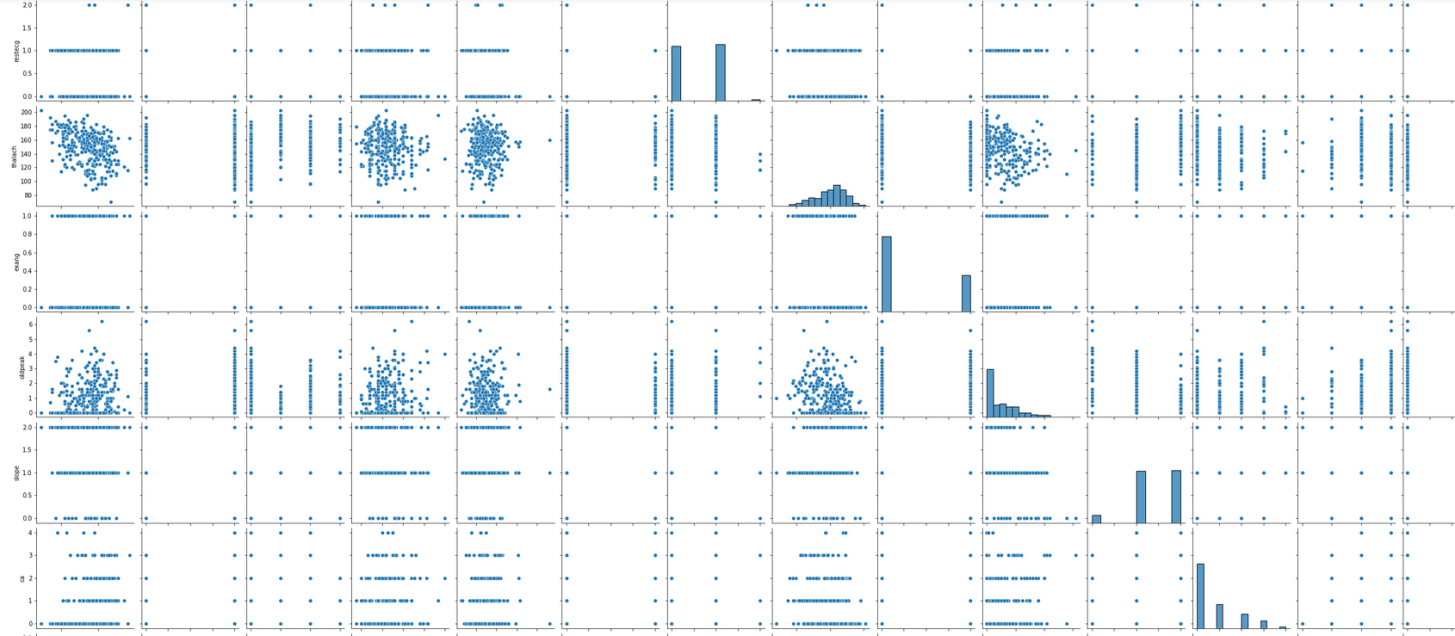


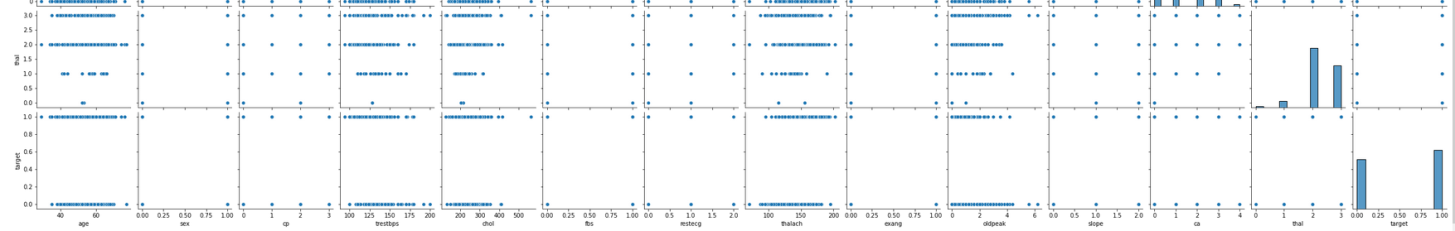
**sns.barplot(heart\_df["slope"],heart\_df['target'])**



**sns.pairplot(heart\_df)**







**#Scaling the dataset**

**from sklearn.preprocessing import StandardScaler**

**scaler=StandardScaler()**

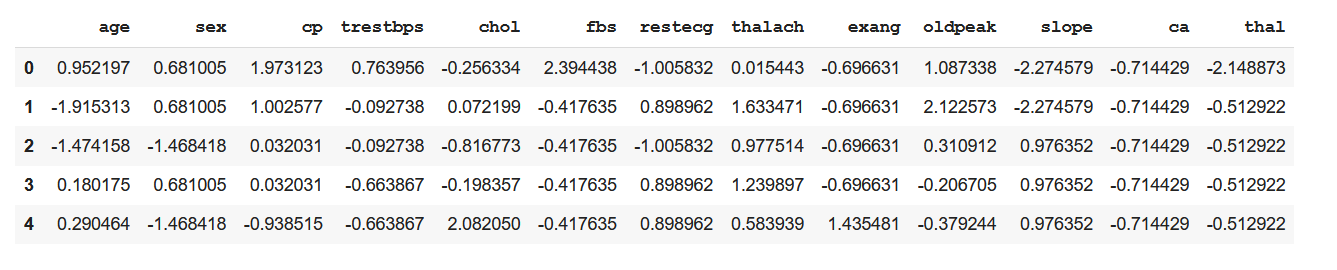
**scaler.fit(heart\_df.drop('target',axis=1))**



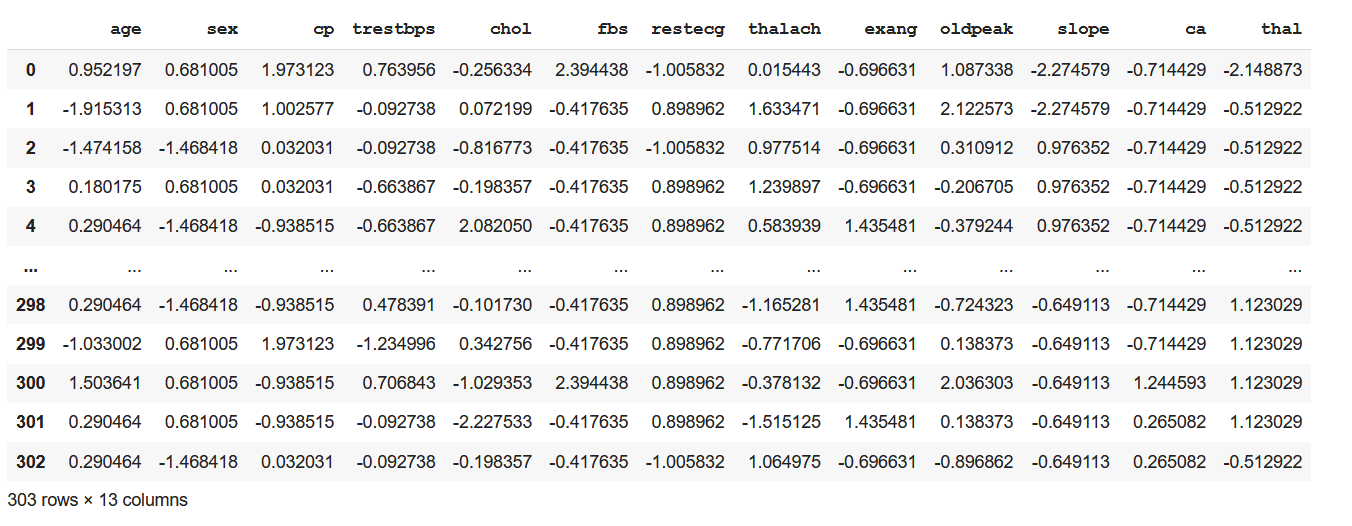
**scaler\_featured=scaler.transform(heart\_df.drop('target',axis=1))**

**heart\_feat=pd.DataFrame(scaler\_featured,columns=heart\_df.columns[:-1])**

**heart\_feat.head()**



**heart\_feat**



**heart\_feat.shape**



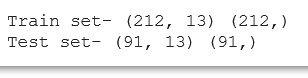
**#splitting dataset into train and test**

**from sklearn.model\_selection import train\_test\_split**

**X\_Train,X\_Test,Y\_Train,Y\_Test=train\_test\_split(heart\_feat,heart\_df['target'],test\_size=0.30,random\_state=42)**

**print('Train set-',X\_Train.shape,Y\_Train.shape)**

**print('Test set-',X\_Test.shape,Y\_Test.shape)**



**# Using K Nearest Neighbours algorithm using the KNeighboursClassifier**

**from sklearn.neighbors import KNeighborsClassifier**

**error\_rate = []**

**for i in range(1,50):**

**knn = KNeighborsClassifier(n\_neighbors=i)**

**knn.fit(X\_Train,Y\_Train)**

**pred\_i = knn.predict(X\_Test)**

**error\_rate.append(np.mean(pred\_i != Y\_Test))**

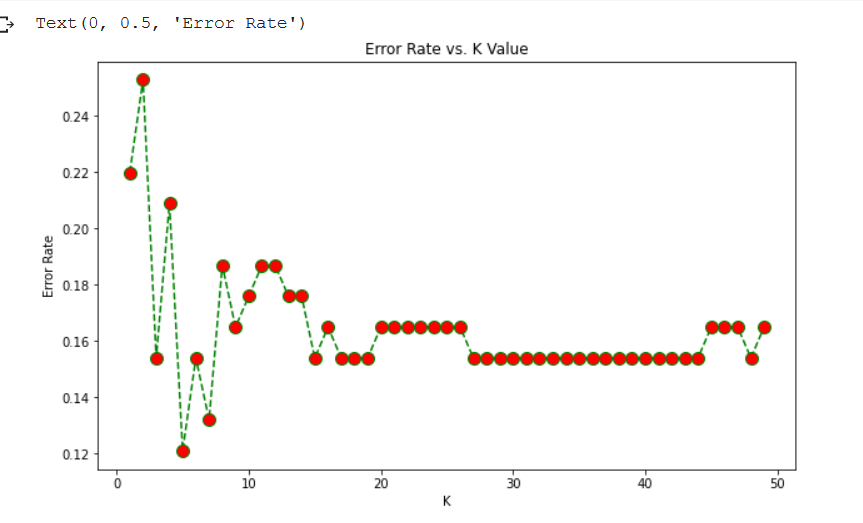
**plt.figure(figsize=(10,6))**

**plt.plot(range(1,50),error\_rate,color='green', linestyle='dashed', marker='o', markerfacecolor='red', markersize=10)**

**plt.title('Error Rate vs. K Value')**

**plt.xlabel('K')**

**plt.ylabel('Error Rate')**



**knn = KNeighborsClassifier(n\_neighbors = 5) #since graphs lowest value is seen around 5**

**knn.fit(X\_Train,Y\_Train)**

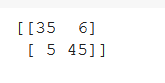


**prediction=knn.predict(X\_Test)**

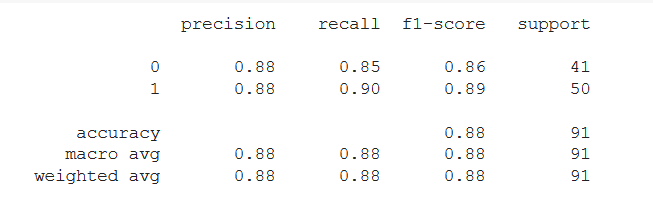
**#CONFUSION MATRIX AND CLASSIFICATION REPORT**

**from sklearn.metrics import classification\_report,confusion\_matrix**

**print(confusion\_matrix(Y\_Test,prediction))**



**print(classification\_report(Y\_Test,prediction))**

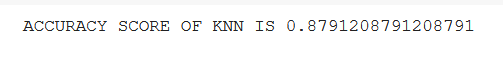


**#CALCULATING ACCURACY**

**from sklearn import metrics**

**knn\_accur=metrics.accuracy\_score(Y\_Test, prediction)**

**print("ACCURACY SCORE OF KNN IS",knn\_accur)**



***CONCLUSION:WE CONCLUDE THAT THE ACCURACY ACHIEVED USING KNN IS ABOUT 87.91 %***

***--------------------------------------------------------------------------------------------------***