# [The documentation for heart disease dataset]

Name : ahmed Mohamed salem

Name: Eslam Mohamed Arafa

[DATA SCIENCE PROJECT ]

**Heart Disease dataset:**

## 1. Introduction:

- A heart attack (Cardiovascular diseases) occurs when the flow of blood to the heart muscle suddenly becomes blocked. From WHO statistics every year 17.9 million dying from heart attack. The medical study says that human life style is the main reason behind this heart problem. Apart from this there are many key factors which warns that the person may/may not getting chance of heart attack.This dataset contain some medical information of patients which tells whether that person getting a heart attack chance is less or more. Using the information explore the dataset and classify the target variable using different Machine Learning models and find out which algorithm suitable for this dataset, **because people know the causes of disease, maintain their health, and avoid symptoms that cause heart problems.**
- **https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset/code.**

- **The dataset consists of a 16×1026 rows and coulmns.**

| Variable (column) | **meaning** |
|---|---|
| age | the time of life at which some particular qualification, power, or capacity arises or rests |
| Sex: | 0:female.<br>1: male. |
| Cp: | chest_pain_type<br>• Value 0: typical angina<br>• Value 1: atypical angina<br>• Value 2: non-anginal pain<br>• Value 3: asymptomatic |
| Trestbps: | Resting blood pressure |
| Chol: | serum cholestoral in mg/dl |
| Fbs: | (fasting blood sugar > 120 mg/dl)<br>• 1 = true;<br>• 0 = false |
| Restecg: | : resting electrocardiographic results<br>• Value 0: normal<br>• Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)<br>• Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria |
| Thalach: | maximum heart rate achieved |
| Exang: | exercise induced angina<br>• 1 = yes<br>• 0 = no |

| | |
|---|---|
| Oldpeak: | ST depression induced by exercise relative to rest |
| Slope: | 1. slope: the slope of the peak exercise ST segment<br>• Value 0: upsloping<br>• Value 1: flat<br>• Value 2: downsloping |
| Ca: | 1. . number of major vessels (0-3) colored by flourosopy |
| Thal: | thalassemia<br>• 1 = fixed defect<br>• 2 = normal<br>• 3 = reversable defect |
| Smoking: | Have you smoked at least 100 cigarettes in your entire life? |
| BMI: | Body Mass Index (BMI). |
| Target: | arget (the lable):<br>• 0 = no disease,<br>• 1 = disease |
| | |

**State of interest is** : blood sugar – thalassemia – Target – slope - blood pressure

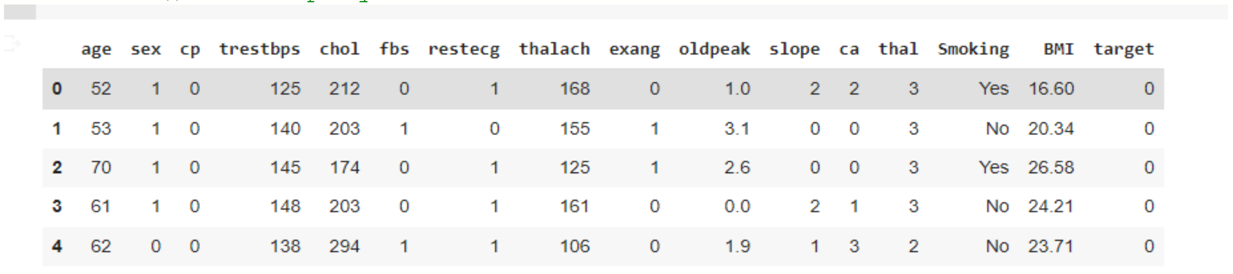
## 2. Loading libraries and importing data:

**1.**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import ComplementNB
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import matplotlib.pyplot as plt
```

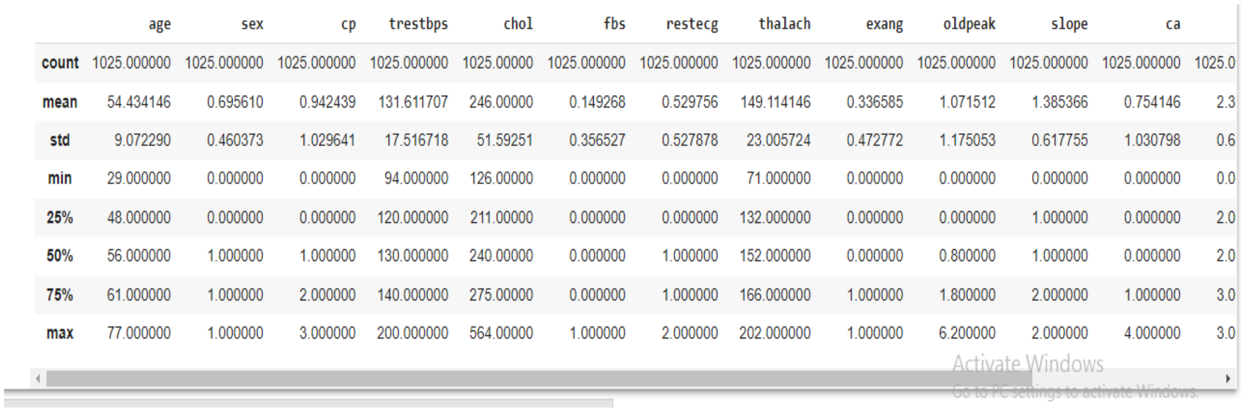
```
2. df=pd.read_csv("heart disease.csv")  # read the dataset.
```

```
3.df.head()  # display the first five rows of the dataset.
```



| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | Smoking | BMI | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 52 | 1 | 0 | 125 | 212 | 0 | 1 | 168 | 0 | 1.0 | 2 | 2 | 3 | Yes | 16.60 | 0 |
| 1 | 53 | 1 | 0 | 140 | 203 | 1 | 0 | 155 | 1 | 3.1 | 0 | 0 | 3 | No | 20.34 | 0 |
| 2 | 70 | 1 | 0 | 145 | 174 | 0 | 1 | 125 | 1 | 2.6 | 0 | 0 | 3 | Yes | 26.58 | 0 |
| 3 | 61 | 1 | 0 | 148 | 203 | 0 | 1 | 161 | 0 | 0.0 | 2 | 1 | 3 | No | 24.21 | 0 |
| 4 | 62 | 0 | 0 | 138 | 294 | 1 | 1 | 106 | 0 | 1.9 | 1 | 3 | 2 | No | 23.71 | 0 |

```
4. df.describe ()
```



| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.00000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.0 |
| mean | 54.434146 | 0.695610 | 0.942439 | 131.611707 | 246.00000 | 0.149268 | 0.529756 | 149.114146 | 0.336585 | 1.071512 | 1.385366 | 0.754146 | 2.3 |
| std | 9.072290 | 0.460373 | 1.029641 | 17.516718 | 51.59251 | 0.356527 | 0.527878 | 23.005724 | 0.472772 | 1.175053 | 0.617755 | 1.030798 | 0.6 |
| min | 29.000000 | 0.000000 | 0.000000 | 94.000000 | 126.00000 | 0.000000 | 0.000000 | 71.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0 |
| 25% | 48.000000 | 0.000000 | 0.000000 | 120.000000 | 211.00000 | 0.000000 | 0.000000 | 132.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 2.0 |
| 50% | 56.000000 | 1.000000 | 1.000000 | 130.000000 | 240.00000 | 0.000000 | 1.000000 | 152.000000 | 0.000000 | 0.800000 | 1.000000 | 0.000000 | 2.0 |
| 75% | 61.000000 | 1.000000 | 2.000000 | 140.000000 | 275.00000 | 0.000000 | 1.000000 | 166.000000 | 1.000000 | 1.800000 | 2.000000 | 1.000000 | 3.0 |
| max | 77.000000 | 1.000000 | 3.000000 | 200.000000 | 564.00000 | 1.000000 | 2.000000 | 202.000000 | 1.000000 | 6.200000 | 2.000000 | 4.000000 | 3.0 |

- **Age** :
  - The average age in the data set is 54.5 years
  - The oldest is 77 years, whereas the youngest is 29 years old
- **Cholesterol**:
  - The average registered cholestrol level is 247.15
  - Maximum level is 564 and the minimum level is 126.
- **Resting blood pressure:**
  - 131 mean, 200 max and 94 min

- **Max heart rate achieved:**
  - The abverage max heart rate registered is 149.5 bpm. The Maximum and the minumum are 202 and 71bpm respectively.
- **St_depression:**
  - The average value of st_dpression is 1.06. Max is 6.2 and the minimum is 0.
- Number of major blood vessels:
  - A maximum of 3 and a minimum of 0 major blood vessels are observed. The mean value is 0.68.

**Thalassemia :**

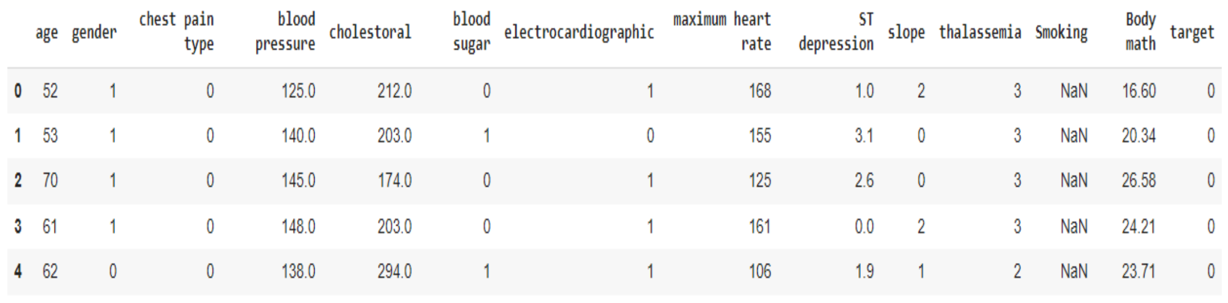
**The mean is** 2.323902 and the count is 1025.000000 and std is 0.620660 and the max =3.000000

**Blood sugar:** std= 0.356527 and max= 1.000000 and mean= 0.149268

**BMI: std=** 6.558601 **and mean =** 29.093639 **and max=** 75.820000 and min = 14.690000

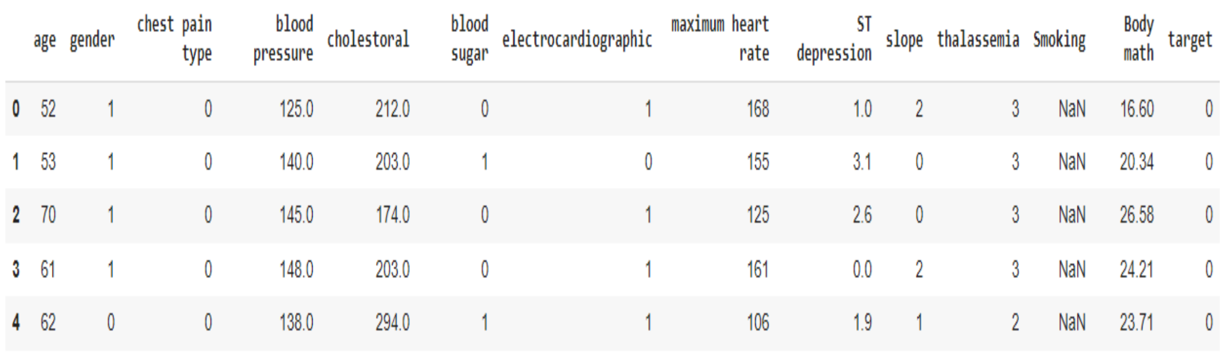
## 3. Data cleaning:

1. 
```
df=df.drop(['exercise induced','num of major vessels'],axis=1)
```



| | age | gender | chest pain type | blood pressure | cholestoral | blood sugar | electrocardiographic | maximum heart rate | ST depression | slope | thalassemia | Smoking | Body math | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 52 | 1 | 0 | 125.0 | 212.0 | 0 | 1 | 168 | 1.0 | 2 | 3 | NaN | 16.60 | 0 |
| **1** | 53 | 1 | 0 | 140.0 | 203.0 | 1 | 0 | 155 | 3.1 | 0 | 3 | NaN | 20.34 | 0 |
| **2** | 70 | 1 | 0 | 145.0 | 174.0 | 0 | 1 | 125 | 2.6 | 0 | 3 | NaN | 26.58 | 0 |
| **3** | 61 | 1 | 0 | 148.0 | 203.0 | 0 | 1 | 161 | 0.0 | 2 | 3 | NaN | 24.21 | 0 |
| **4** | 62 | 0 | 0 | 138.0 | 294.0 | 1 | 1 | 106 | 1.9 | 1 | 2 | NaN | 23.71 | 0 |

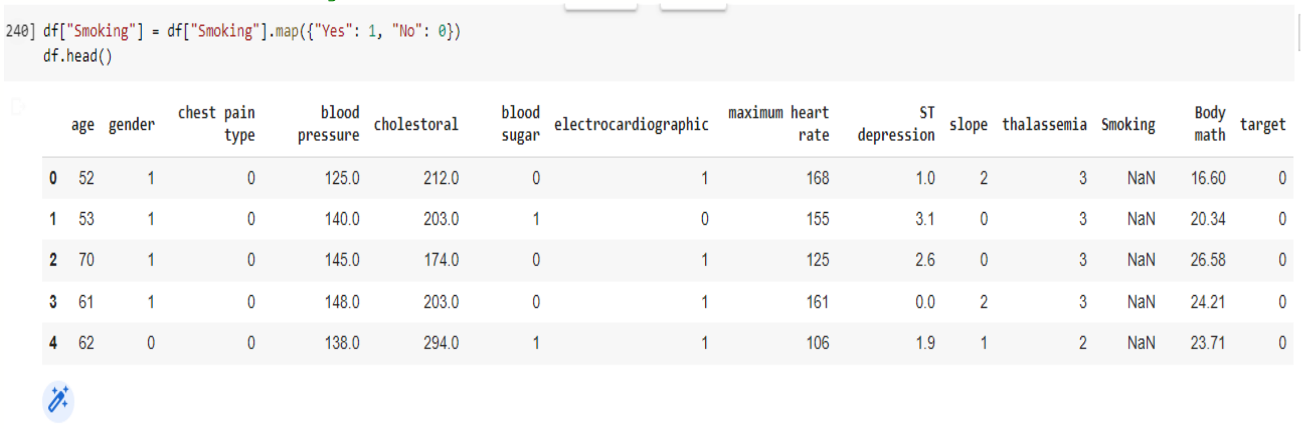
2.
```
df=df.rename(columns={'sex':'gender','cp':'chest pain type','trestbps':'blood pressure','chol':'cholestoral','fbs':'blood sugar','restecg':'electrocardiographic','thalach':'maximum heart rate','exang':'exercise induced','oldpeak':'ST depression','ca':'num of major vessels','thal':'thalassemia','BMI':'Body math'},inplace= False)

#we rename the name of some columns to be clear enough.
```



| | age | gender | chest pain type | blood pressure | cholestoral | blood sugar | electrocardiographic | maximum heart rate | ST depression | slope | thalassemia | Smoking | Body math | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 52 | 1 | 0 | 125.0 | 212.0 | 0 | 1 | 168 | 1.0 | 2 | 3 | NaN | 16.60 | 0 |
| 1 | 53 | 1 | 0 | 140.0 | 203.0 | 1 | 0 | 155 | 3.1 | 0 | 3 | NaN | 20.34 | 0 |
| 2 | 70 | 1 | 0 | 145.0 | 174.0 | 0 | 1 | 125 | 2.6 | 0 | 3 | NaN | 26.58 | 0 |
| 3 | 61 | 1 | 0 | 148.0 | 203.0 | 0 | 1 | 161 | 0.0 | 2 | 3 | NaN | 24.21 | 0 |
| 4 | 62 | 0 | 0 | 138.0 | 294.0 | 1 | 1 | 106 | 1.9 | 1 | 2 | NaN | 23.71 | 0 |

3. df.duplicated().value_counts()

```
df = df.drop_duplicates()  # we just drope the duplicates values in the da
tasets.
df.shape.
df.isnull().sum()  #check if there are null values or not.
#There is no missing values in this dataset!
```

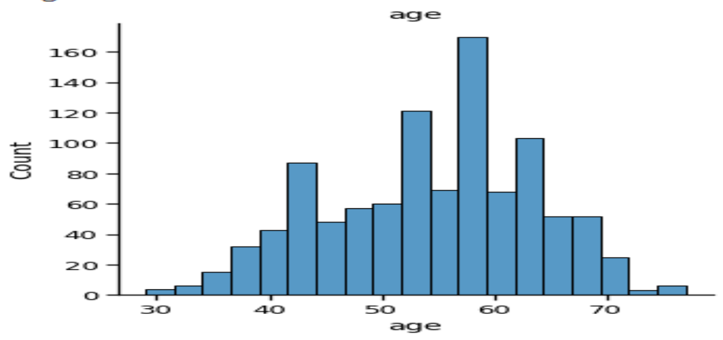


```
240] df["Smoking"] = df["Smoking"].map({"Yes": 1, "No": 0})
     df.head()
```

| | age | gender | chest pain type | blood pressure | cholestoral | blood sugar | electrocardiographic | maximum heart rate | ST depression | slope | thalassemia | Smoking | Body math | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 52 | 1 | 0 | 125.0 | 212.0 | 0 | 1 | 168 | 1.0 | 2 | 3 | NaN | 16.60 | 0 |
| 1 | 53 | 1 | 0 | 140.0 | 203.0 | 1 | 0 | 155 | 3.1 | 0 | 3 | NaN | 20.34 | 0 |
| 2 | 70 | 1 | 0 | 145.0 | 174.0 | 0 | 1 | 125 | 2.6 | 0 | 3 | NaN | 26.58 | 0 |
| 3 | 61 | 1 | 0 | 148.0 | 203.0 | 0 | 1 | 161 | 0.0 | 2 | 3 | NaN | 24.21 | 0 |
| 4 | 62 | 0 | 0 | 138.0 | 294.0 | 1 | 1 | 106 | 1.9 | 1 | 2 | NaN | 23.71 | 0 |

```
for f in df:
        plt.figure(figsize = (10,10))
        sns.displot(df[f])
        plt.title(f)
        plt.show();
# check if there are outliers on dataset or not using displot
```

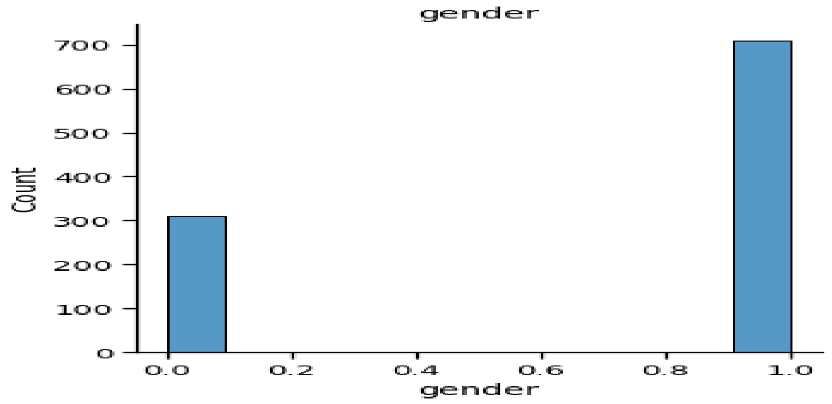
```
<Figure size 720x720 with 0 Axes>
```

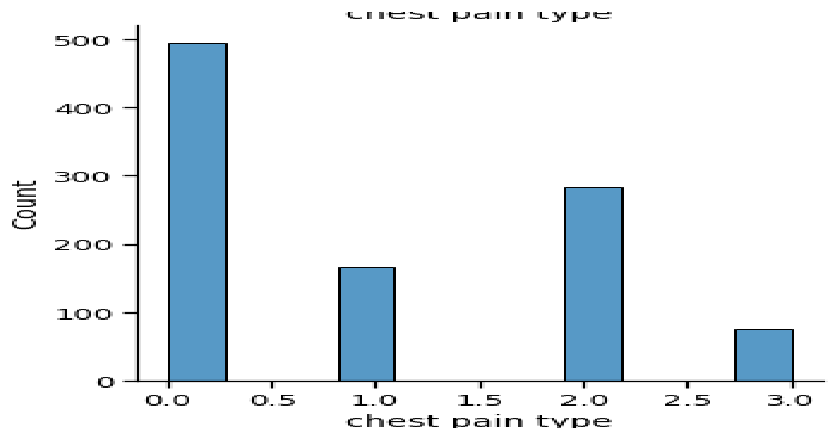


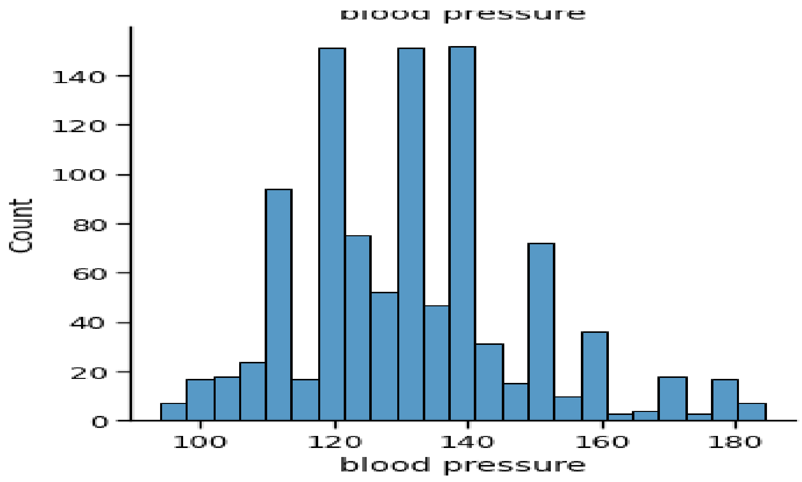
```
<Figure size 720x720 with 0 Axes>
```

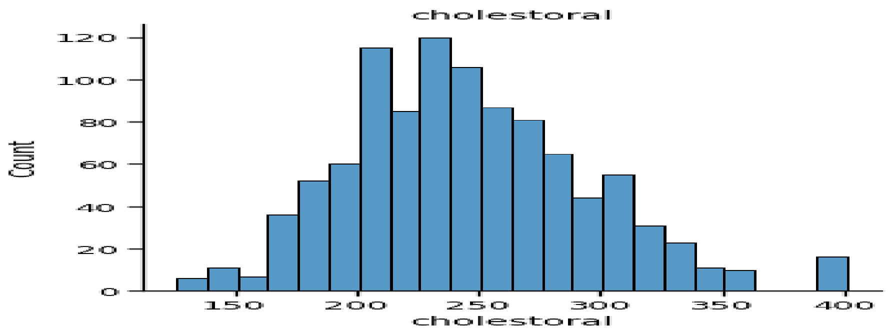
gender

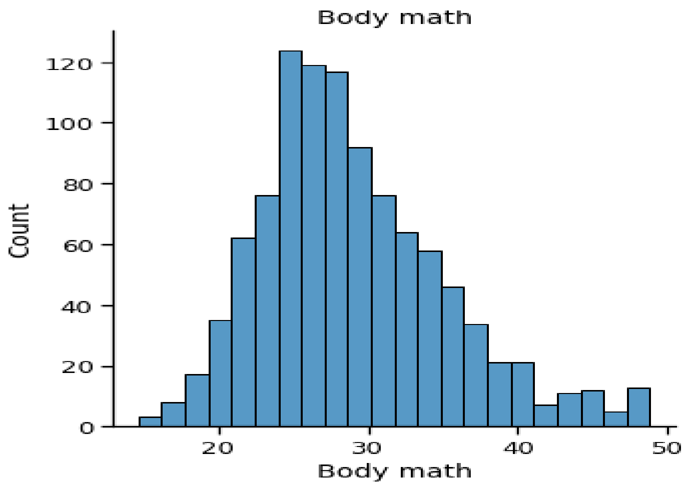
```
<Figure size 720x720 with 0 Axes>
```


gender
Count
700
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
gender

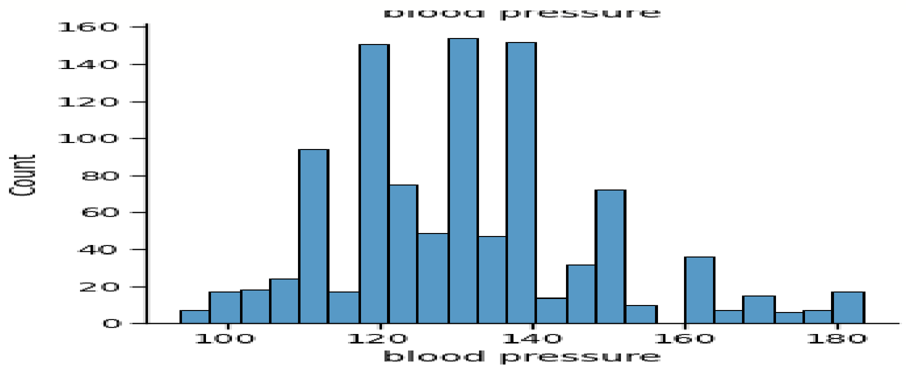

chest pain type
Count
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
chest pain type

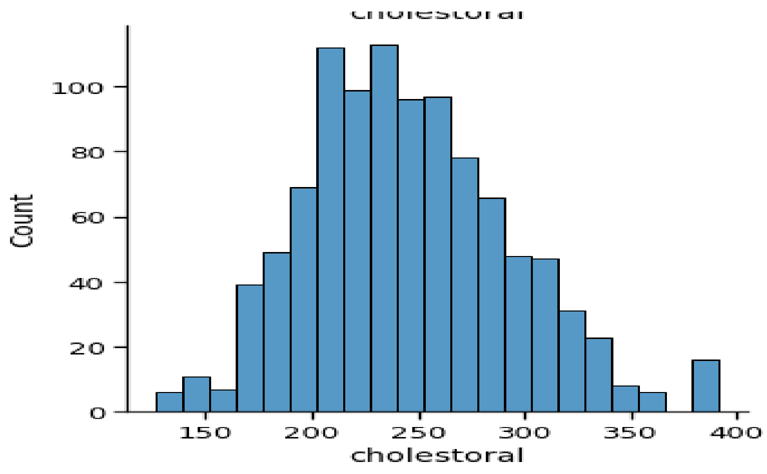

blood pressure
Count
140
120
100
80
60
40
20
0
100
120
140
160
180
blood pressure

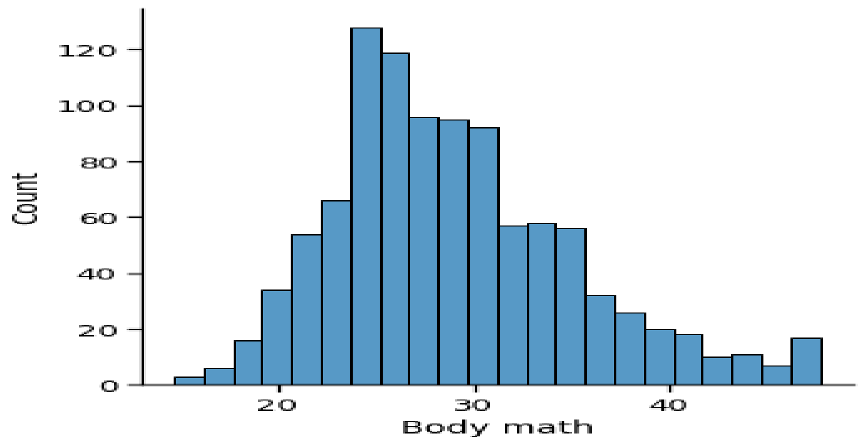




```
  def rem_outlier(df, feature):
 lower_bound = df[feature].mean() - 3*df[feature].std()
 upper_bound = df[feature].mean() + 3*df[feature].std()

 df.loc[df[feature] < lower_bound , feature] = lower_bound
 df.loc[df[feature] > upper_bound, feature] = upper_bound
# creating function to remove columns that have outliers values.


for f in ['blood pressure', 'cholestoral','Body math']:
    rem_outlier(df, f)
    # here we just call function remove outliers to delete the columns tha
t we highlighted in the for loop.

for f in ['blood pressure','cholestoral','Body math']:
     plt.figure(figsize = (10,10))
     sns.displot(df[f])
     plt.title(f)
     plt.show();
```

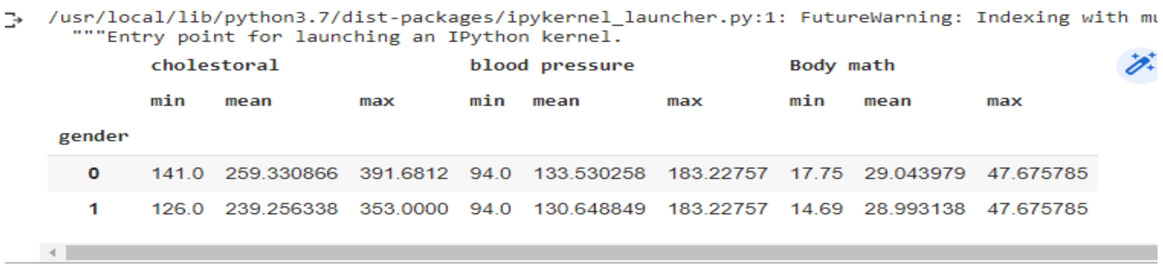






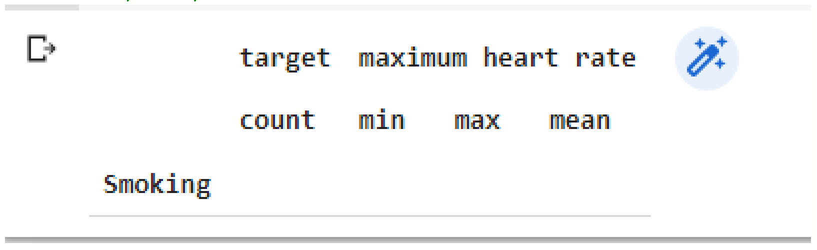
## 3. **Pandas’ aggregate functions**

```
data=df.groupby('gender')['cholestoral','blood pressure','Body math'].aggr
egate(['min','mean','max'])
data  # we can see the cholestoral's min,max and mean for females are grea
ter than males.
      # the blood suger's min and max valuse are equal in males and female
s but the blood's mean in females are greater than the males.
      # the Body math's mean and min in females is greater than males.
```



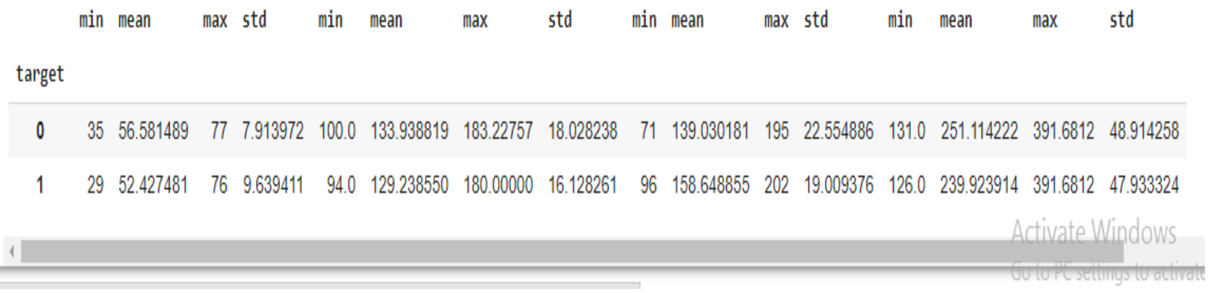
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Indexing with mu
  """Entry point for launching an IPython kernel.
         cholestoral                    blood pressure                  Body math
         min     mean        max        min    mean        max          min     mean       max
gender
  0      141.0   259.330866  391.6812   94.0   133.530258  183.22757    17.75   29.043979  47.675785
  1      126.0   239.256338  353.0000   94.0   130.648849  183.22757    14.69   28.993138  47.675785
```

```
dat=df.groupby('target')['age','blood pressure','maximum heart rate','chol
estoral'].aggregate(['min','mean','max','std'])
dat  # we select this columns and grouped by target to calculate the value
 of min,max,mean and std .4.
```



```
              target   maximum heart rate
              count    min    max    mean
     Smoking
```

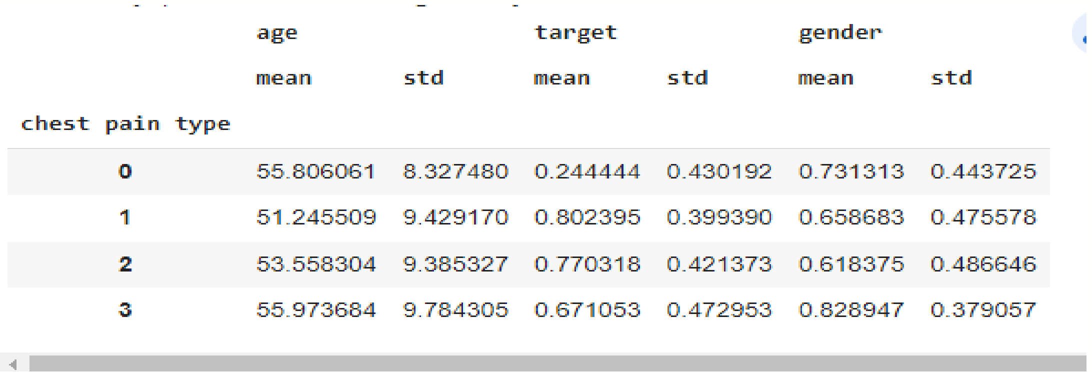
```
    # after implements this operation we can see the people that have hear
t disease and other columns that
```



| | min | mean | max | std | min | mean | max | std | min | mean | max | std | min | mean | max | std |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **target** | | | | | | | | | | | | | | | | |
| **0** | 35 | 56.581489 | 77 | 7.913972 | 100.0 | 133.938819 | 183.22757 | 18.028238 | 71 | 139.030181 | 195 | 22.554886 | 131.0 | 251.114222 | 391.6812 | 48.914258 |
| **1** | 29 | 52.427481 | 76 | 9.639411 | 94.0 | 129.238550 | 180.00000 | 16.128261 | 96 | 158.648855 | 202 | 19.009376 | 126.0 | 239.923914 | 391.6812 | 47.933324 |

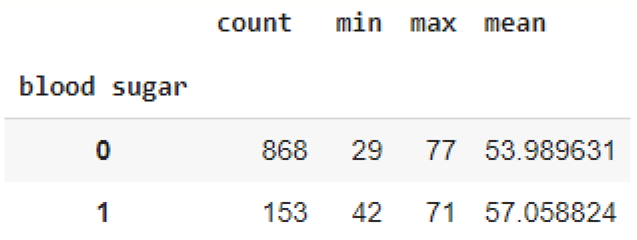
Activate Windows
Go to PC settings to activate

```
daf=df.groupby('chest pain type')['age','target','gender'].aggregate(['mea
n','std'])
daf
```



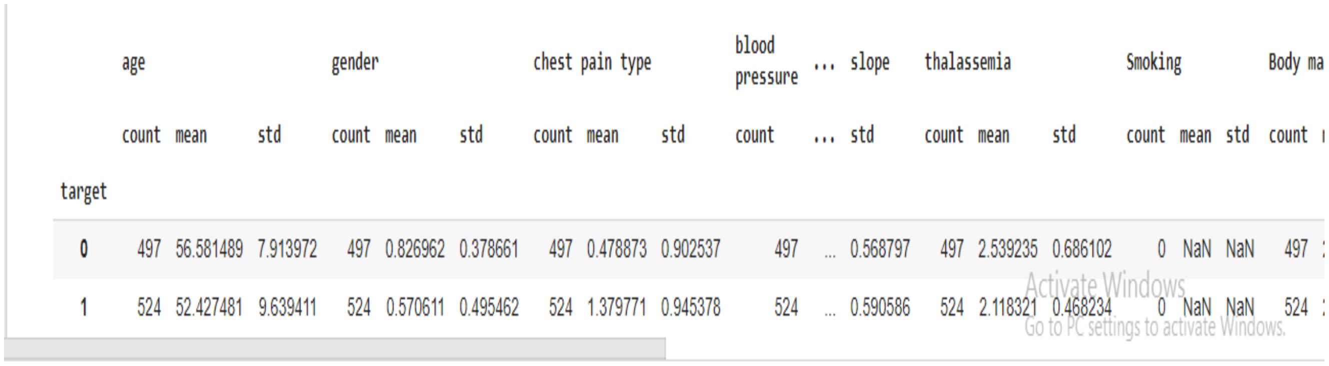
| | age | | target | | gender | |
|---|---|---|---|---|---|---|
| | mean | std | mean | std | mean | std |
| **chest pain type** | | | | | | |
| **0** | 55.806061 | 8.327480 | 0.244444 | 0.430192 | 0.731313 | 0.443725 |
| **1** | 51.245509 | 9.429170 | 0.802395 | 0.399390 | 0.658683 | 0.475578 |
| **2** | 53.558304 | 9.385327 | 0.770318 | 0.421373 | 0.618375 | 0.486646 |
| **3** | 55.973684 | 9.784305 | 0.671053 | 0.472953 | 0.828947 | 0.379057 |

```
dar=df.groupby('blood sugar').aggregate({'gender':'count','age':['min','ma
x','mean']})

dar
```



| | count | min | max | mean |
|---|---|---|---|---|
| **blood sugar** | | | | |
| **0** | 868 | 29 | 77 | 53.989631 |
| **1** | 153 | 42 | 71 | 57.058824 |

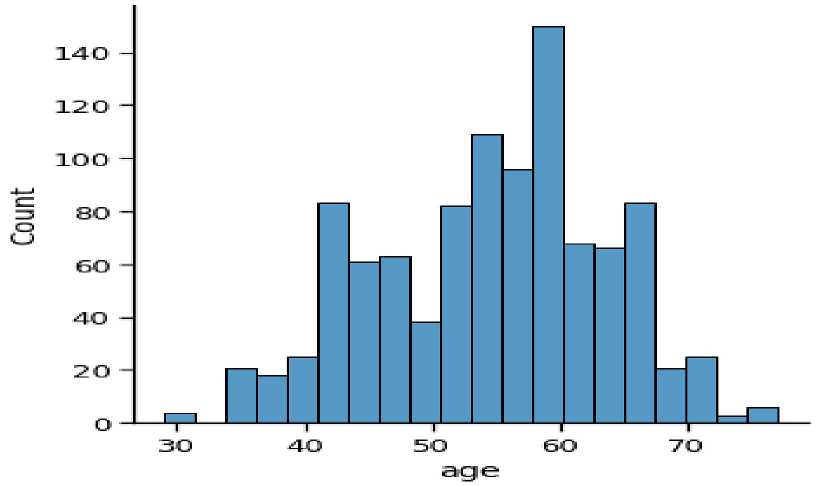
```
daw=df.groupby('target').aggregate(['count','mean','std'])
daw   # we calculate the count,mean and std of target according to all col
umns in the dataset.
```

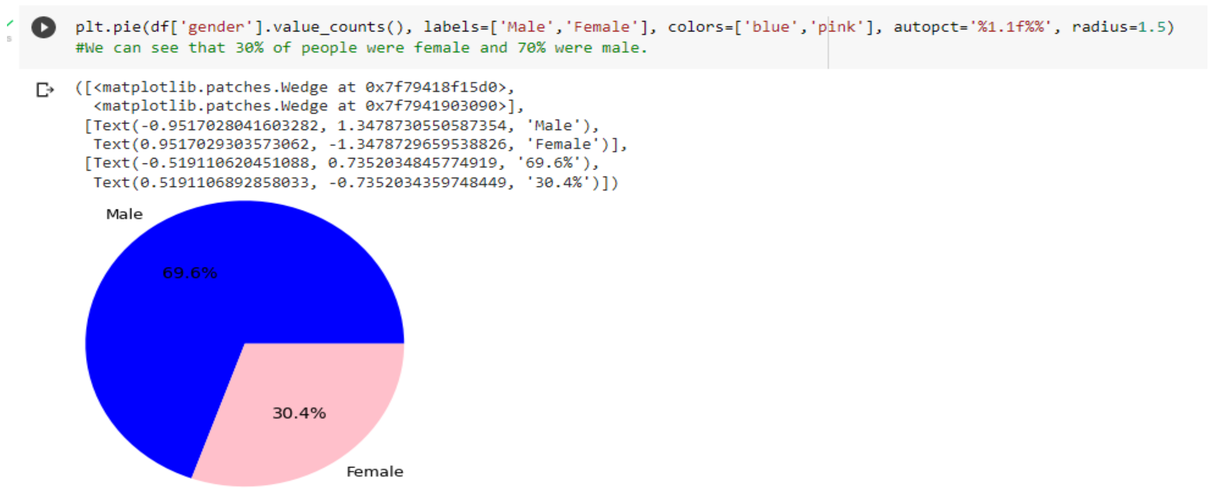


| | age | | | gender | | | chest pain type | | | blood pressure | ... | slope | thalassemia | | | Smoking | | | Body ma |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | count | mean | std | count | mean | std | count | mean | std | count | ... | std | count | mean | std | count | mean | std | count |
| target | | | | | | | | | | | | | | | | | | | |
| 0 | 497 | 56.581489 | 7.913972 | 497 | 0.826962 | 0.378661 | 497 | 0.478873 | 0.902537 | 497 | ... | 0.568797 | 497 | 2.539235 | 0.686102 | 0 | NaN | NaN | 497 |
| 1 | 524 | 52.427481 | 9.639411 | 524 | 0.570611 | 0.495462 | 524 | 1.379771 | 0.945378 | 524 | ... | 0.590586 | 524 | 2.118321 | 0.468234 | 0 | NaN | NaN | 524 |

## 4. Data visualization:

```
sns.displot(df['age'],bins=20)
plt.show()
# the average of age in dataset is between 40 to 68.
```

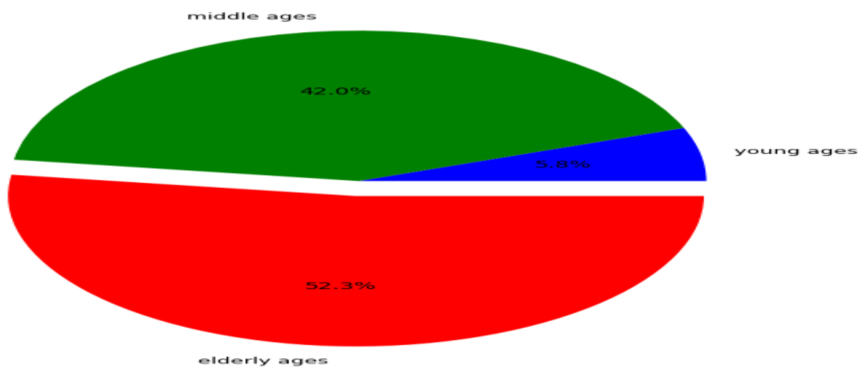




```
plt.pie(df['gender'].value_counts(), labels=['Male','Female'], colors=['blue','pink'], autopct='%1.1f%%', radius=1.5)
#We can see that 30% of people were female and 70% were male.
```

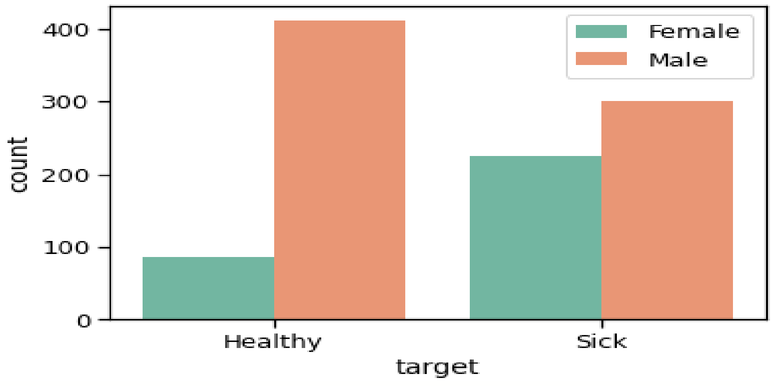
```
([<matplotlib.patches.Wedge at 0x7f79418f15d0>,
  <matplotlib.patches.Wedge at 0x7f7941903090>],
 [Text(-0.9517028041603282, 1.3478730550587354, 'Male'),
  Text(0.9517029303573062, -1.3478729659538826, 'Female')],
 [Text(-0.519110620451088, 0.7352034845774919, '69.6%'),
  Text(0.519110689285803З, -0.7352034359748449, '30.4%')])
```

```
minAge=min(df.age)
maxAge=max(df.age)
meanAge=df.age.mean()
print('Min Age :',minAge)
print('Max Age :',maxAge)
print('Mean Age :',meanAge)


Young = df[(df.age>=29)&(df.age<40)]
Middle = df[(df.age>=40)&(df.age<55)]
Elder = df[(df.age>55)]

colors = ['blue','black','Red']
explode = [0,0,0.1]
plt.figure(figsize=(10,10))
sns.set_context('notebook',font_scale = 1.2)
plt.pie([len(Young),len(Middle),len(Elder)],labels=['young ages','middle a
ges','elderly ages'],explode=explode,colors=colors, autopct='%1.1f%%')
plt.tight_layout()
```



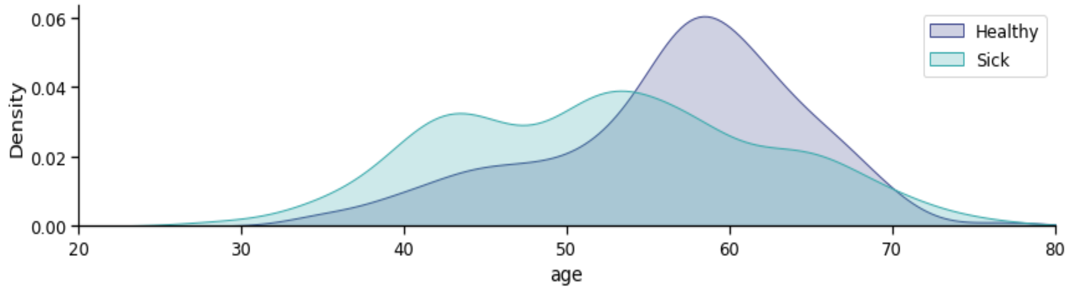
```
fig = sns.countplot(x = 'target', data = df, hue = 'gender', palette='Set2
')
fig.set_xticklabels(labels=['Healthy', 'Sick'])
plt.legend(['Female', 'Male'])
```

```
<matplotlib.legend.Legend at 0x7f7941baf810>
```



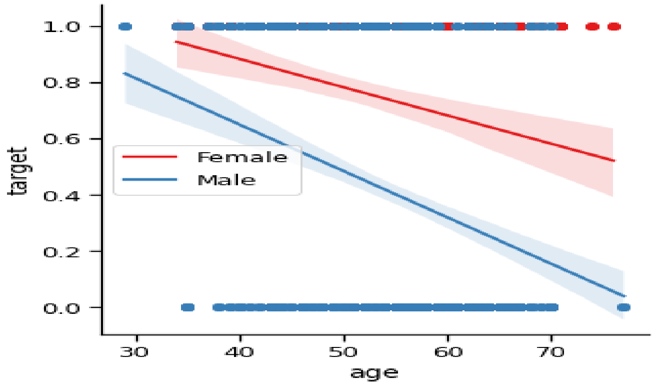
```
fig = sns.FacetGrid(df, hue="target",aspect=4, palette='mako')
fig.map(sns.kdeplot,'age',shade= True)
fig.set(xlim=(20,80))
plt.legend(labels=['Healthy' , 'Sick'])
#see a peak of healthy people at 60. Let's see if age is a factor in heart
 disease:
```

```
<matplotlib.legend.Legend at 0x7f7941f10390>
```

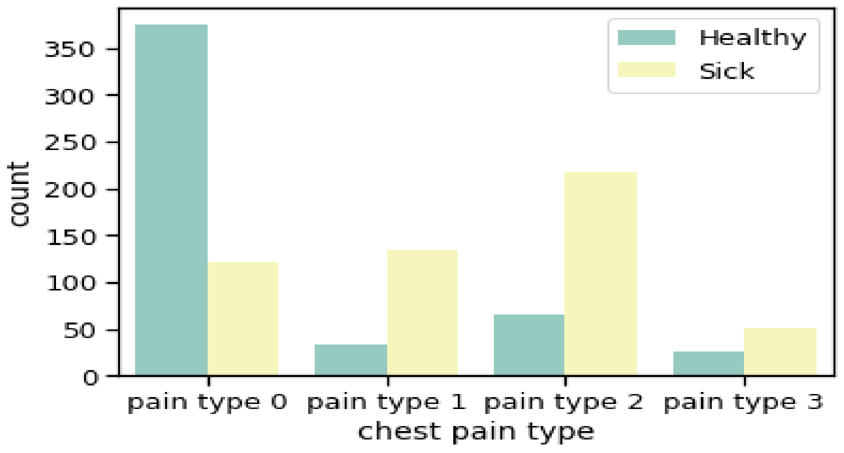


```
sns.lmplot(x='age',y='target',data=df, hue='gender', legend=False,  palett
e='Set1')
plt.legend(['Female', 'Male'])
```

```
<matplotlib.legend.Legend at 0x7f794a3ff390>
```

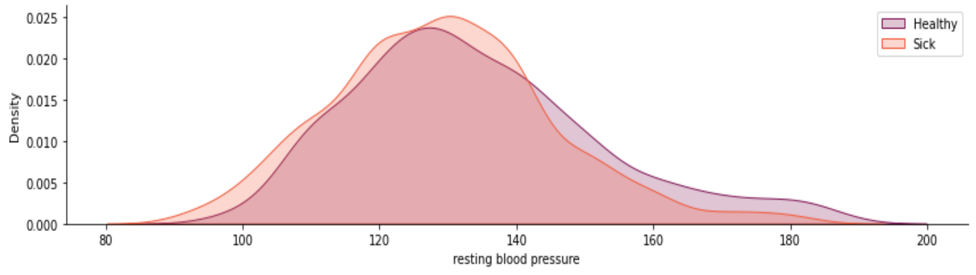


```
fig = sns.countplot(x = 'chest pain type', data = df, hue = 'target', pale
tte='Set3')
plt.legend(['Healthy', 'Sick'])
fig.set_xticklabels(labels=['pain type 0', 'pain type 1', 'pain type 2', '
pain type 3'])
# the people who have heart diseas is that have chest pain type (atypical
angina / non angina pectoris / asymtomatic) more that the others that don,
t have.
```

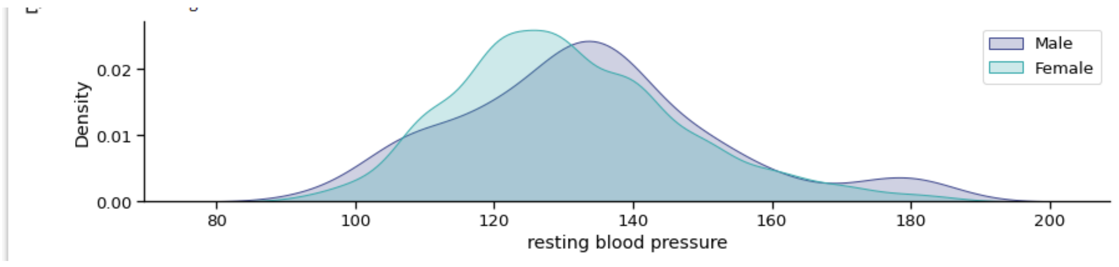


```
fig = sns.FacetGrid(df, hue="target",aspect=4, palette='rocket')
fig.map(sns.kdeplot,'blood pressure',shade= True)
plt.legend(labels=['Healthy' , 'Sick'])
fig.set(xlabel = 'resting blood pressure')
# normal blood pressure is 120.
```

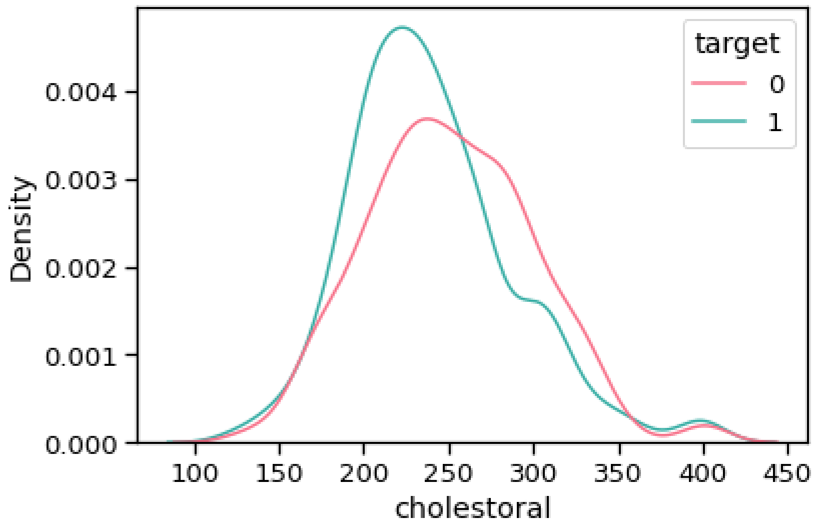
```
<seaborn.axisgrid.FacetGrid at 0x7f7947b1c190>
```



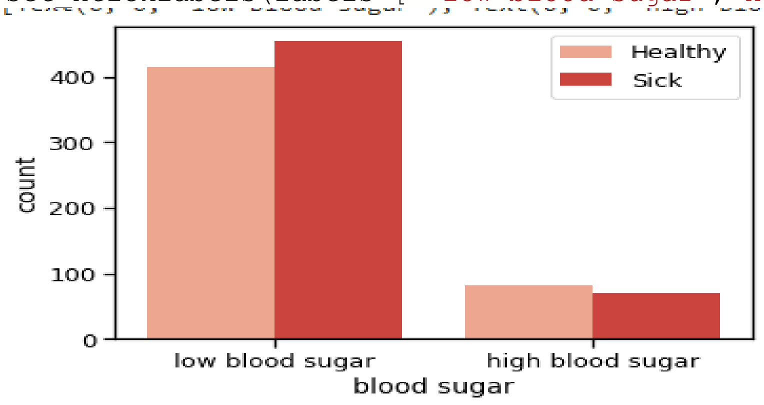
```
fig = sns.FacetGrid(df, hue="gender",aspect=4, palette='mako')
fig.map(sns.kdeplot,'blood pressure',shade= True)
plt.legend(labels=['Male' , 'Female'])
fig.set(xlabel = 'resting blood pressure')
```



```
sns.kdeplot(data=df, x='cholestoral', hue='target',palette="husl")
```

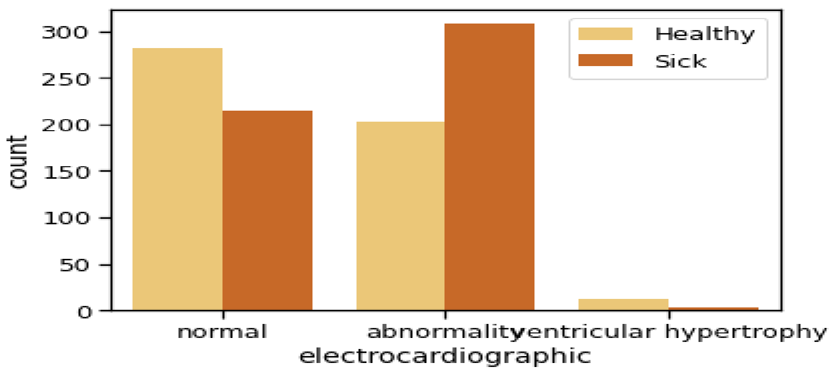


```
fig = sns.countplot(x = 'blood sugar', data = df, hue = 'target', palette=
'Reds')
plt.legend(['Healthy', 'Sick'])
fig.set_xticklabels(labels=[ 'low blood sugar','high blood sugar'])
```



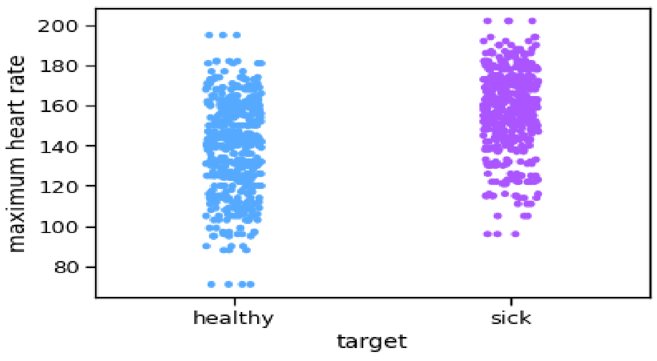
```
fig = sns.countplot(x = 'electrocardiographic', data = df, hue = 'target',
 palette='YlOrBr')
plt.legend(['Healthy', 'Sick'])
```

```
fig.set_xticklabels(labels=[ 'normal','abnormality','ventricular hypertrop
hy'])
```



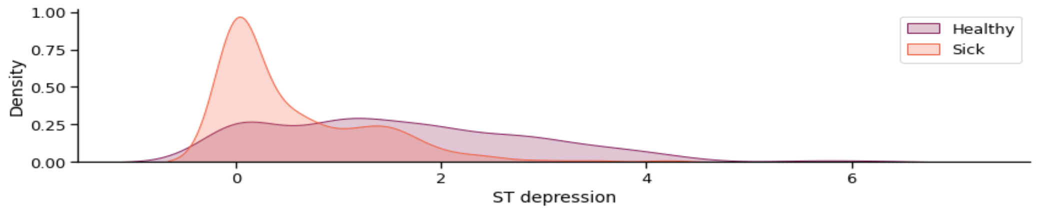
```
[233] fig = sns.stripplot(data=df, x='target', y='maximum heart rate', palette='cool')
      fig.set_xticklabels(labels=['healthy','sick'])
```

```
[Text(0, 0, 'healthy'), Text(0, 0, 'sick')]
```



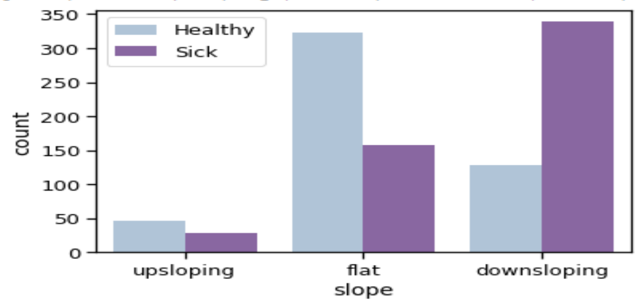
```
fig = sns.FacetGrid(df, hue="target",aspect=4, palette='rocket')
fig.map(sns.kdeplot,'ST depression',shade= True)
plt.legend(labels=['Healthy' , 'Sick'])
```

```
<matplotlib.legend.Legend at 0x7f7940db4f50>
```



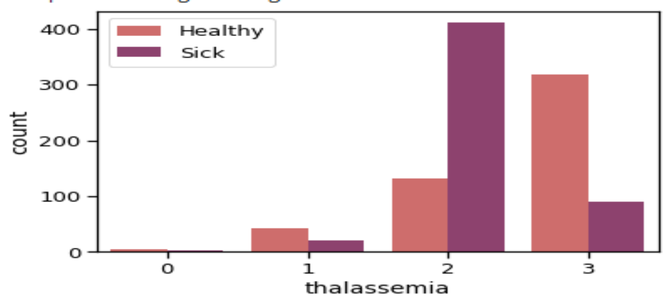
```python
fig = sns.countplot(x = 'slope', data = df, hue = 'target', palette='BuPu')
plt.legend(['Healthy', 'Sick'])
fig.set_xticklabels(labels=[ 'upsloping','flat', 'downsloping'])
```

```
[Text(0, 0, 'upsloping'), Text(0, 0, 'flat'), Text(0, 0, 'downsloping')]
```



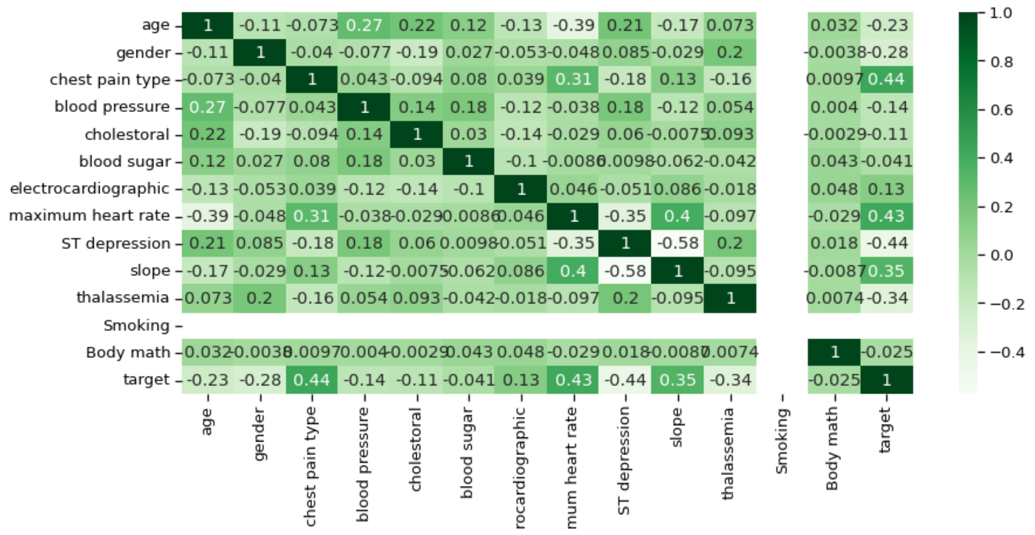
```python
fig = sns.countplot(x = 'thalassemia', data = df, hue = 'target',palette='flare')
plt.legend(['Healthy', 'Sick'])
```

```
<matplotlib.legend.Legend at 0x7f7941bd9f90>
```

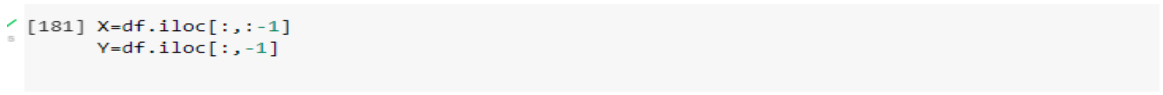


```python
df.corr().style.background_gradient()
plt.figure(figsize=(14,6))
sns.heatmap(df.corr(),annot=True,cmap='Greens')
#RdYlGn
```

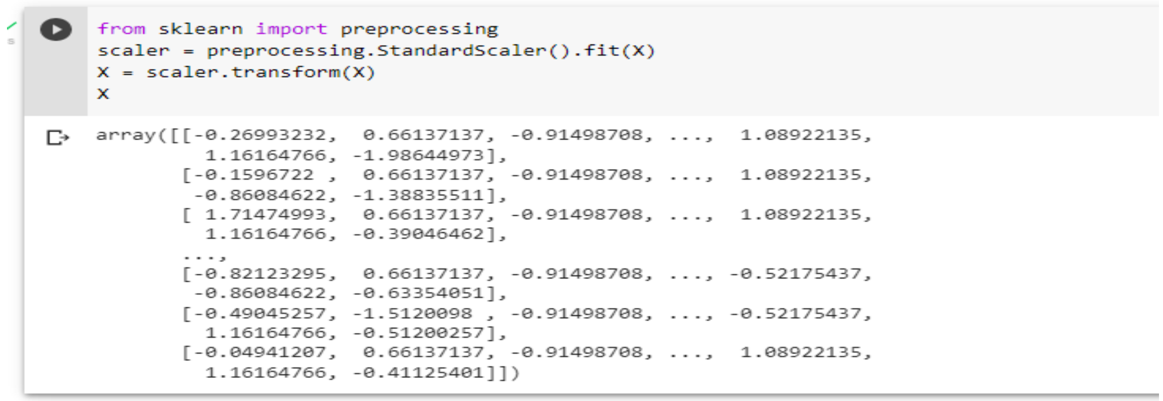
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7941ec6ad0>
```



## 5.model machine learning(logistic regression):

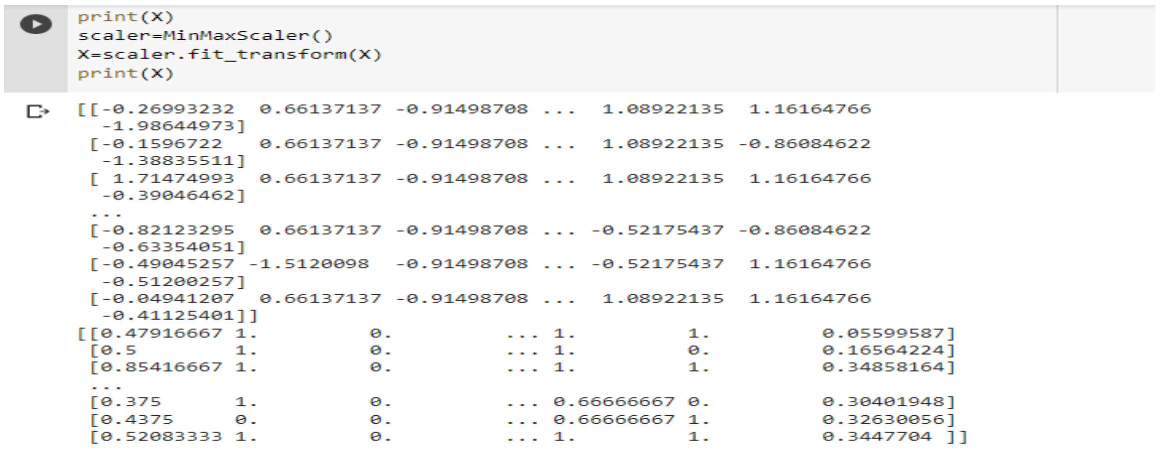


```
[181] X=df.iloc[:,:-1]
      Y=df.iloc[:,-1]
```



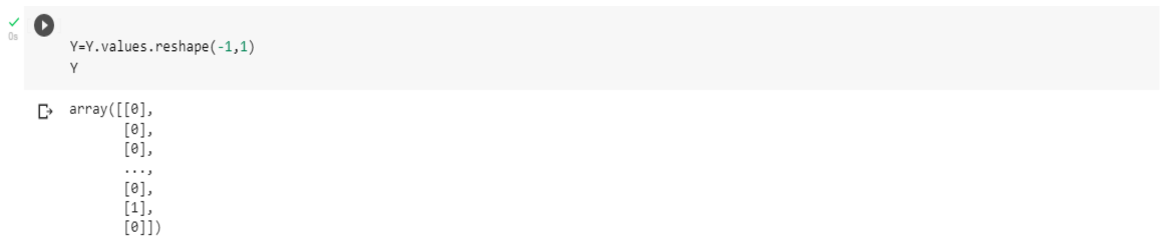
```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler().fit(X)
X = scaler.transform(X)
X
```

```
array([[-0.26993232,  0.66137137, -0.91498708, ...,  1.08922135,
         1.16164766, -1.98644973],
       [-0.1596722 ,  0.66137137, -0.91498708, ...,  1.08922135,
        -0.86084622, -1.38835511],
       [ 1.71474993,  0.66137137, -0.91498708, ...,  1.08922135,
         1.16164766, -0.39046462],
       ...,
       [-0.82123295,  0.66137137, -0.91498708, ..., -0.52175437,
        -0.86084622, -0.63354051],
       [-0.49045257, -1.5120098 , -0.91498708, ..., -0.52175437,
         1.16164766, -0.51200257],
       [-0.04941207,  0.66137137, -0.91498708, ...,  1.08922135,
         1.16164766, -0.41125401]])
```



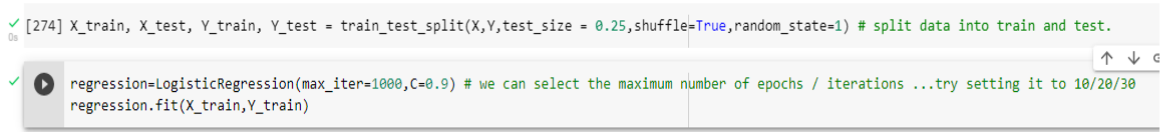
```
print(X)
scaler=MinMaxScaler()
X=scaler.fit_transform(X)
print(X)
```

```
[[-0.26993232  0.66137137 -0.91498708 ...  1.08922135  1.16164766
  -1.98644973]
 [-0.1596722   0.66137137 -0.91498708 ...  1.08922135 -0.86084622
  -1.38835511]
 [ 1.71474993  0.66137137 -0.91498708 ...  1.08922135  1.16164766
  -0.39046462]
 ...
 [-0.82123295  0.66137137 -0.91498708 ... -0.52175437 -0.86084622
  -0.63354051]
 [-0.49045257 -1.5120098  -0.91498708 ... -0.52175437  1.16164766
  -0.51200257]
 [-0.04941207  0.66137137 -0.91498708 ...  1.08922135  1.16164766
  -0.41125401]]
[[0.47916667 1.         0.         ... 1.         1.         0.05599587]
 [0.5        1.         0.         ... 1.         0.         0.16564224]
 [0.85416667 1.         0.         ... 1.         1.         0.34858164]
 ...
 [0.375      1.         0.         ... 0.66666667 0.         0.30401948]
 [0.4375     0.         0.         ... 0.66666667 1.         0.32630056]
 [0.52083333 1.         0.         ... 1.         1.         0.3447704 ]]
```



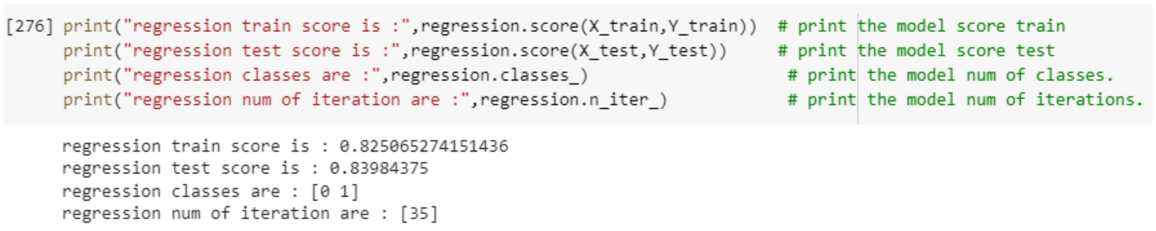
```
Y=Y.values.reshape(-1,1)
Y
```

```
array([[0],
       [0],
       [0],
       ...,
       [0],
       [1],
       [0]])
```

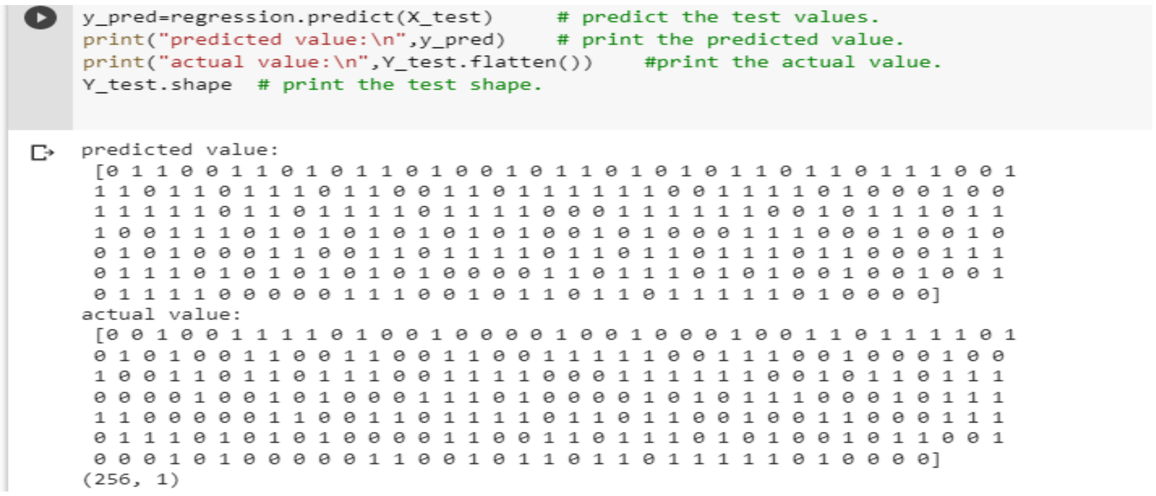


```
[274] X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.25,shuffle=True,random_state=1) # split data into train and test.
```

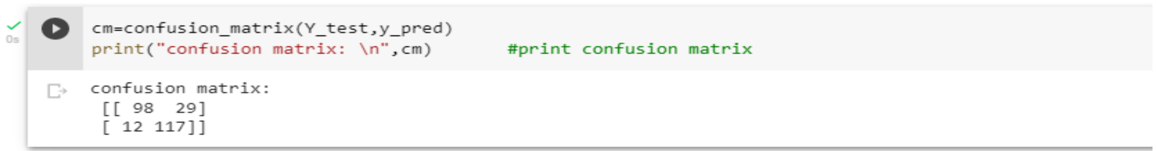
```
regression=LogisticRegression(max_iter=1000,C=0.9) # we can select the maximum number of epochs / iterations ...try setting it to 10/20/30
regression.fit(X_train,Y_train)
```



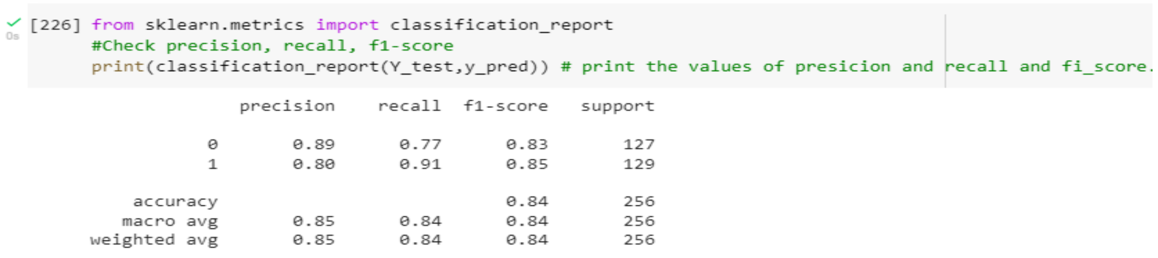
```
[276] print("regression train score is :",regression.score(X_train,Y_train))  # print the model score train
      print("regression test score is :",regression.score(X_test,Y_test))     # print the model score test
      print("regression classes are :",regression.classes_)                    # print the model num of classes.
      print("regression num of iteration are :",regression.n_iter_)            # print the model num of iterations.

      regression train score is : 0.825065274151436
      regression test score is : 0.83984375
      regression classes are : [0 1]
      regression num of iteration are : [35]
```



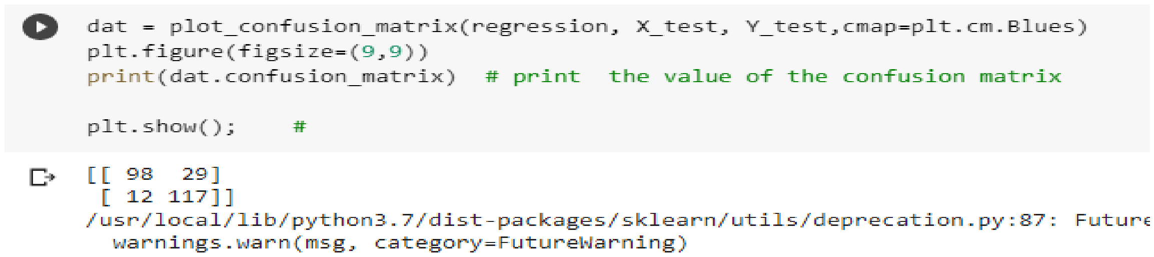
```
y_pred=regression.predict(X_test)     # predict the test values.
print("predicted value:\n",y_pred)    # print the predicted value.
print("actual value:\n",Y_test.flatten())    #print the actual value.
Y_test.shape  # print the test shape.

predicted value:
 [0 1 1 0 0 1 1 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1 0 1 0 1 1 0 1 1 0 1 1 1 0 0 1
 1 1 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 0 1 0 0 0 1 0 0
 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1
 1 0 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 0 0 1 1 1 0 0 0 1 0 0 1 0
 0 1 0 1 0 0 0 1 1 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 0 0 1 1 1
 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 1 0 0 1 0 0 1
 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0]
actual value:
 [0 0 1 0 0 1 1 1 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1
 0 1 0 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 0 0 1 0 0 0 1 0 0
 1 0 0 1 1 0 1 1 0 1 1 1 0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 1 0 1 1 0 1 1 1
 0 0 0 0 1 0 0 1 0 1 0 0 0 1 1 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0 0 0 1 0 1 1 1
 1 1 0 0 0 0 0 1 1 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0 1 1 0 0 0 1 1 1
 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 1 0 0 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 1 0 0 1
 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0]
(256, 1)
```



```
cm=confusion_matrix(Y_test,y_pred)
print("confusion matrix: \n",cm)       #print confusion matrix

confusion matrix:
 [[ 98  29]
 [ 12 117]]
```

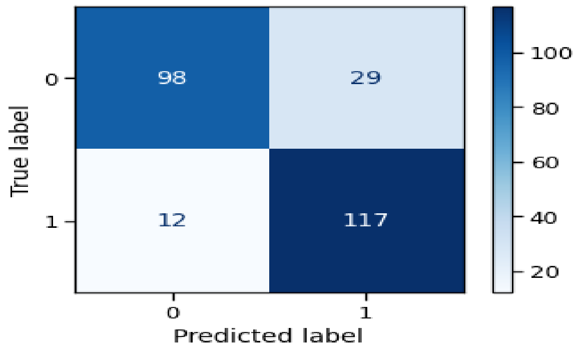


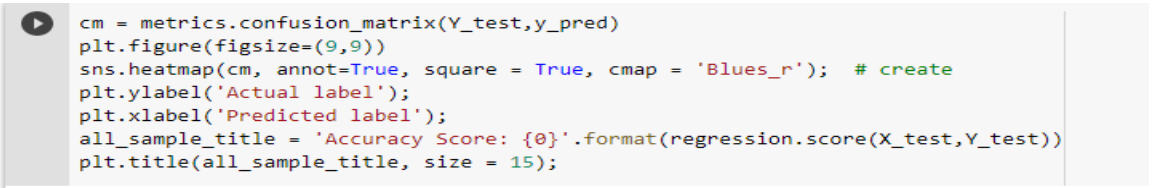
```
[226] from sklearn.metrics import classification_report
      #Check precision, recall, f1-score
      print(classification_report(Y_test,y_pred)) # print the values of presicion and recall and fi_score.

              precision    recall  f1-score   support

           0       0.89      0.77      0.83       127
           1       0.80      0.91      0.85       129

    accuracy                           0.84       256
   macro avg       0.85      0.84      0.84       256
weighted avg       0.85      0.84      0.84       256
```



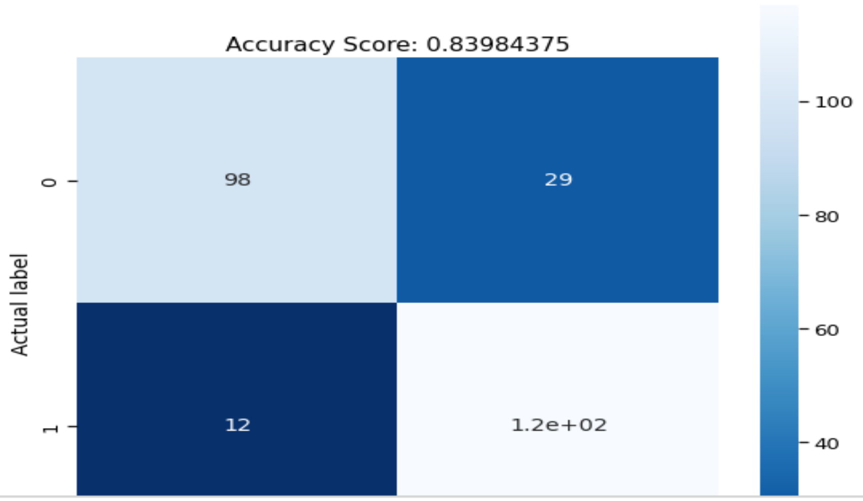
```
dat = plot_confusion_matrix(regression, X_test, Y_test,cmap=plt.cm.Blues)
plt.figure(figsize=(9,9))
print(dat.confusion_matrix)  # print  the value of the confusion matrix

plt.show();    #
```

```
[[ 98  29]
 [ 12 117]]
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: Future
  warnings.warn(msg, category=FutureWarning)
```





```
cm = metrics.confusion_matrix(Y_test,y_pred)
plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, square = True, cmap = 'Blues_r');  # create
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(regression.score(X_test,Y_test))
plt.title(all_sample_title, size = 15);
```


Accuracy Score: 0.83984375
1.2e+02
Actual label