Cardiovascular Disease Pediction Project.

Source:

Creators:

1. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
2. University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
3. University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
4. V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

- According to WHO , Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worlwide.
- More than four out of five CVD deaths are due to heart attacks and strokes, and one third of these deaths occur prematurely in people under 70 years of age.
- CVDs are a group of disorders of the heart and blood vessels and include coronary heart disease, cerebrovascular disease, rheumatic heart disease and other conditions.

- It can also be associated with damage to arteries in organs such as the brain, heart, kidneys and eyes. People with cardiovascular disease or who are at high cardiovascular risk (due to the existence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help.

`How can we reduce the Heart diseases death rate?`

- The early prognosis of cardiovascular diseases can aid in making decisions on lifestyle changes in high risk patients and in turn reduce the complications.
- We Aim to deploy a machine learning model that can predict whether the person may have a heart disease or not.

Our Problem

- 🩺 Predict the presence or absence of cardiovascular disease (CVD) using the patient examination results.

Import Libraries

In [1]:
```
import numpy as np              # NumPy is a Python library used for working with
import pandas as pd             # Pandas is mainly used for data analysis. Pandas
import seaborn as sns           # Seaborn is a library in Python predominantly us
import matplotlib.pyplot as plt # Matplotlib is a cross-platform, data visualizat
```

In [2]:
```
# Loading the data from csv file to a Pandas DataFrame
df = pd.read_csv('heart.csv')
```

🔍 EDA :

Let's Explore our data !!!

- Analyze by describing the data

In [3]:
```
# First 5 rows of the dataframe
df.head()
```

Out[3]:

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

In [4]:
```
# Columns of data
df.columns
```

Out[4]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

🖹 Features:

There are 2 types of input features:

- Objective: factual information;

- Examination: results of medical examination;

sex : 1 = Male , 0=Female

cp : Chest Pain

- Angina: Angina is caused when there is not enough oxygen-rich blood flowing to a certain part of the heart. The arteries of the heart become narrow due to fatty deposits in the artery walls. The narrowing of arteries means that blood supply to the heart is reduced, causing angina. Value 0: typical angina || Value 1: atypical angina || Value 2: non-anginal pain || 3: asymptomatic

threstbps :Resting blood pressure

- (Normal pressure with no exercise)

chol: serum cholestoral in mg/dl

- Cholesterol means the blockage for blood supply in the blood vessels

fbs: fasting blood sugar > 120 mg/dl

- (1 = true; 0 = false) blood sugar taken after a long gap between a meal and the test. Typically, it's taken before any meal in the morning.

restecg: resting electrocardiographic results (values 0,1,2)

- ECG values taken while person is on rest which means no exercise and normal functioning of heart is happening

thalach: maximum heart rate achieved

exang: exercise induced angina

- (1 = yes; 0 = no) is chest pain while exercising or doing any physical activity.

oldpeak = ST depression induced by exercise relative to rest

- ST Depression is the difference between value of ECG at rest and after exercise.
- An electrocardiogram records the electrical signals in your heart. It's a common and painless test used to quickly detect heart problems and monitor your heart's health. Electrocardiograms — also called ECGs or EKGs — are often done in a doctor's office, a clinic or a hospital room. ECG machines are standard equipment in operating rooms and ambulances. Some personal devices, such as smart watches,

slope: the slope of the peak exercise ST segment

- Value 0: upsloping — Value 1: flat — Value 2: downsloping

ca: number of major vessels (0-3) colored by flourosopy

- Fluoroscopy is an imaging technique that uses X-rays to obtain real-time moving images of the interior of an object. In its primary application of medical imaging, a fluoroscope (/

ˈflʊərəskoʊp/) allows a physician to see the internal structure and function of a patient, so that the pumping action of the heart or the motion of swallowing, for example, can be watched

thal:The Types of thalassemia

- (1,3 = normal; 6 = fixed defect; 7 = reversable defect)

In [5]:
```
# Getting some informations about the dataset
df.info()
```

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

In [6]:
```
# Number of rows and columns
df.shape
```

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

In [7]:
```
df.values
```

Out[7]:
```
array([[63.,  1.,  3., ...,  0.,  1.,  1.],
       [37.,  1.,  2., ...,  0.,  2.,  1.],
       [41.,  0.,  1., ...,  0.,  2.,  1.],
       ...,
       [68.,  1.,  0., ...,  2.,  3.,  0.],
       [57.,  1.,  0., ...,  1.,  3.,  0.],
       [57.,  0.,  1., ...,  1.,  2.,  0.]])
```

In [8]:
```
sex = df.sex.values
restecg = df.restecg.values
exang = df.exang.values
slope = df.slope.values
print("Sex values are : ",set(sex))
print("Restecg values are : ",set(restecg))
print("Exang values are : ",set(exang))
print("Slope values are : ",set(slope))
```

```
Sex values are :  {0, 1}
Restecg values are :  {0, 1, 2}
Exang values are :  {0, 1}
Slope values are :  {0, 1, 2}
```

In [9]:
```
df.apply(lambda x:len(x.unique()))
```

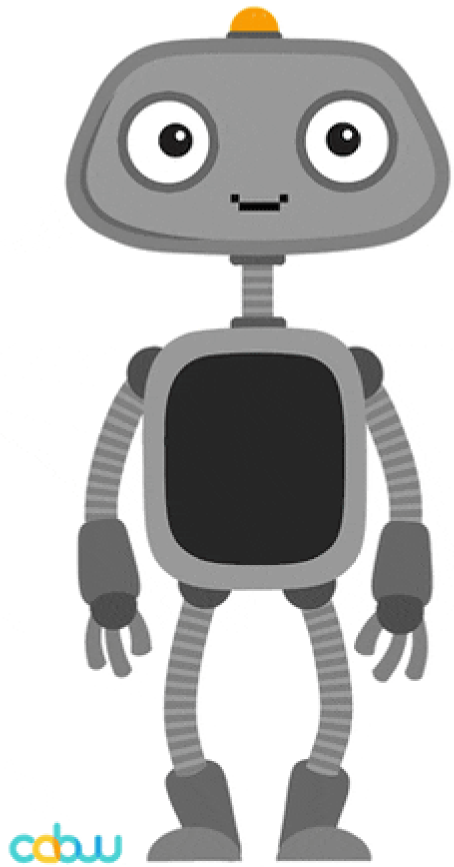
Out[9]:
```
age          41
sex           2
cp            4
trestbps     49
chol        152
fbs           2
restecg       3
thalach      91
exang         2
oldpeak      40
slope         3
ca            5
thal          4
target        2
dtype: int64
```

- Dealing with Missing Values :

In [10]:
```
# Checking for missing values
df.isna().sum()
```

Out[10]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

- No missing data, cool! :)



- Does it indicate that the data is really stable? check the outliers or Wrong Data. . .
- First we will need some statistical information...

- Statistical measuers (mean , standard deviation , min , max) :

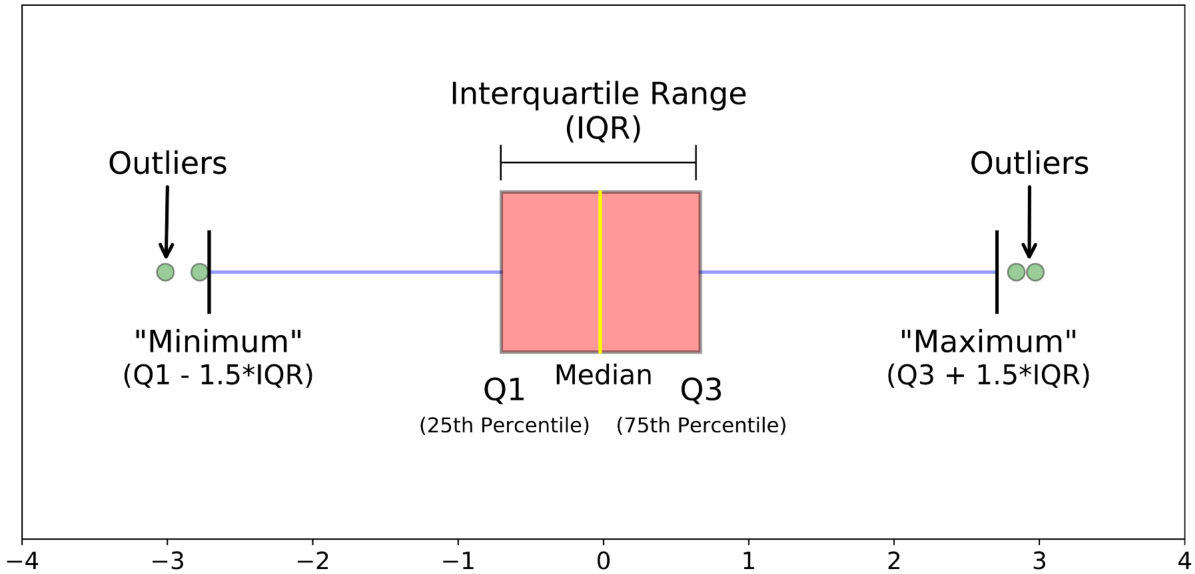
In [11]:
```
display(df.describe())
```

	age	sex	cp	trestbps	chol	fbs	restecg	tha
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.64
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.90
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.00
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.50
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.00
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.00
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.00

Central Tendency Measures		
Measure	**Formula**	**Description**
Mean	$\sum x/n$	Balance Point
Median	n+1/2 Position	Middle Value when ordered
Mode	None	Most frequent

- Detecting outliers 😼 :

- A box and whisker plot (also called boxplot) shows the five numbers summary of a set of data : minimum , lower quartile , meduium, upper quartile and maximum .

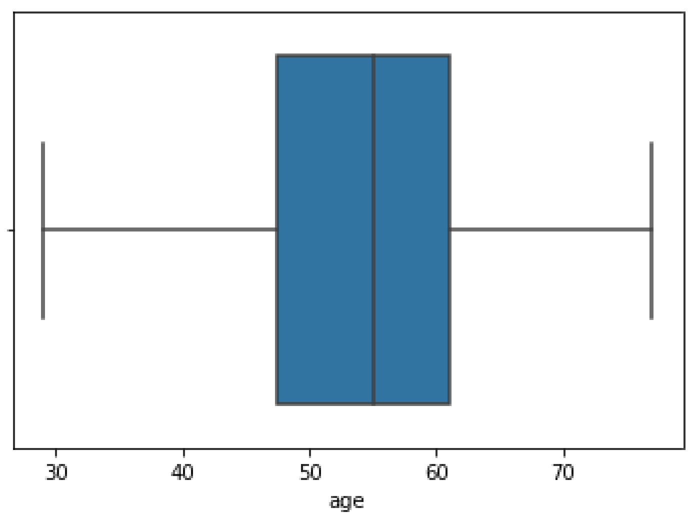


Age outliers :

In [12]:
```
sns.boxplot(df['age'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[12]: <AxesSubplot:xlabel='age'>

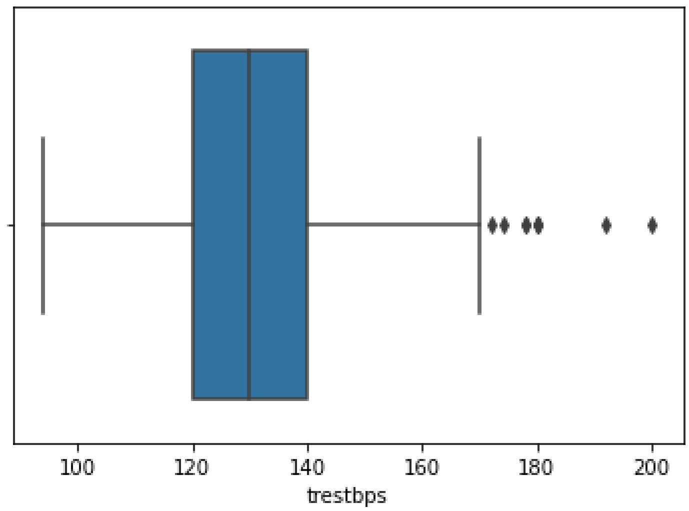


Trestbps outliers :

In [13]:
```
sns.boxplot(df['trestbps'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[13]: <AxesSubplot:xlabel='trestbps'>



In [14]:
```
# Trestbps
Q1 = df.trestbps.quantile(0.25)
Q3 = df.trestbps.quantile(0.75)
Q1, Q3
```

Out[14]: (120.0, 140.0)

In [15]:
```
IQR = Q3 - Q1
IQR
```

Out[15]: 20.0

In [16]:
```
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit
```

Out[16]: (90.0, 170.0)

In [17]:
```
df[df['trestbps'] > upper_limit]
```

Out[17]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
101	59	1	3	178	270	0	0	145	0	4.2	0	0	3	1
110	64	0	0	180	325	0	1	154	1	0.0	2	0	2	1
203	68	1	2	180	274	1	0	150	1	1.6	1	0	3	0
223	56	0	0	200	288	1	0	133	1	4.0	0	2	3	0
241	59	0	0	174	249	0	1	143	1	0.0	1	0	2	0
248	54	1	1	192	283	0	0	195	0	0.0	2	1	3	0
260	66	0	0	178	228	1	1	165	1	1.0	1	2	3	0
266	55	0	0	180	327	0	2	117	1	3.4	1	0	2	0

In [18]:
```
df = df[df['trestbps'] <= upper_limit]
```

In [19]:
```
df.shape
```

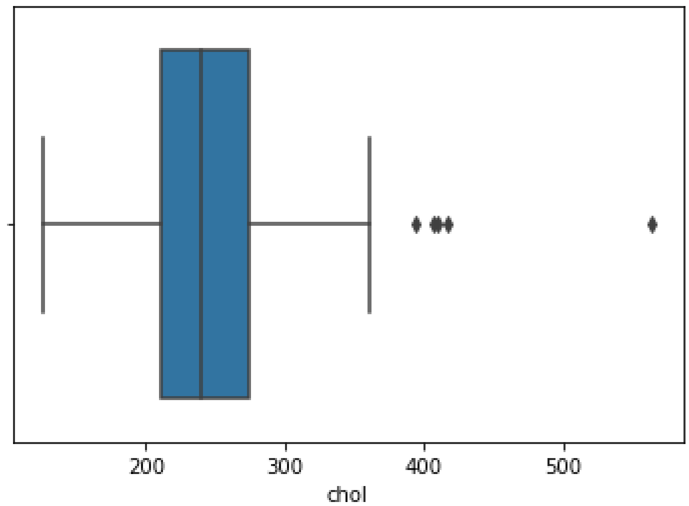
Out[19]:
```
(294, 14)
```

Chol outliers :

In [20]:
```
sns.boxplot(df['chol'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[20]: <AxesSubplot:xlabel='chol'>



In [21]:
```
# Chol
Q1 = df.chol.quantile(0.25)
Q3 = df.chol.quantile(0.75)
Q1, Q3
```

Out[21]: (211.0, 273.75)

In [22]:
```
IQR = Q3 - Q1
IQR
```

Out[22]: 62.75

In [23]:
```
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit
```

Out[23]: (116.875, 367.875)

In [24]:
```
df[df['chol'] > upper_limit]
```

Out[24]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
28	65	0	2	140	417	1	0	157	0	0.8	2	1	2	1
85	67	0	2	115	564	0	0	160	0	1.6	1	0	3	1
96	62	0	0	140	394	0	0	157	0	1.2	1	0	2	1
220	63	0	0	150	407	0	0	154	0	4.0	1	3	3	0
246	56	0	0	134	409	0	0	150	1	1.9	1	2	3	0

In [25]:
```
df = df[df['chol'] < upper_limit]
```

In [26]:
```
df.shape
```

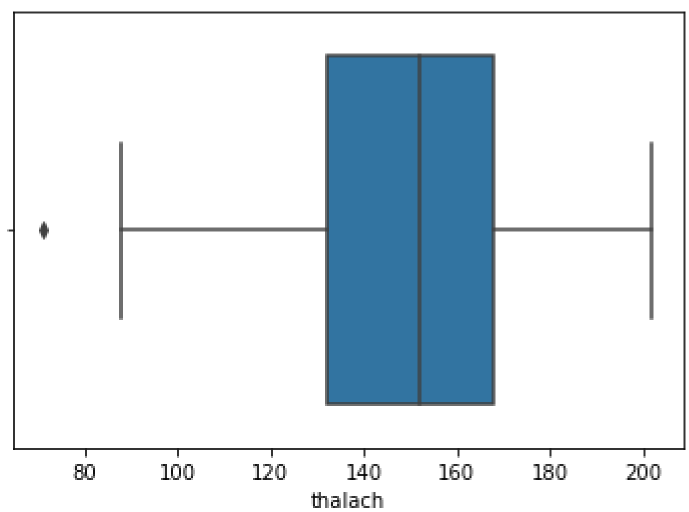
Out[26]:
```
(289, 14)
```

Thalach outliers :

In [27]:
```
sns.boxplot(df['thalach'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[27]:
```
<AxesSubplot:xlabel='thalach'>
```



In [28]:
```
# Thalach
Q1 = df.thalach.quantile(0.25)
Q3 = df.thalach.quantile(0.75)
Q1, Q3
```

Out[28]:
```
(132.0, 168.0)
```

In [29]:
```
IQR = Q3 - Q1
IQR
```

Out[29]: 36.0

In [30]:
```
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit
```

Out[30]: (78.0, 222.0)

In [31]:
```
df[df['thalach'] < lower_limit]
```

Out[31]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
272	67	1	0	120	237	0	1	71	0	1.0	1	0	2	0

In [32]:
```
df = df[df['thalach'] > lower_limit]
```

In [33]:
```
df.shape
```

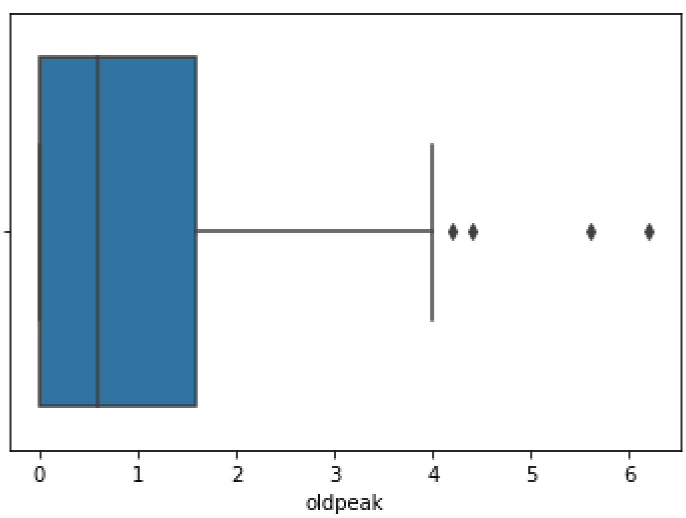
Out[33]: (288, 14)

Oldpeak outliers :

In [34]:
```
sns.boxplot(df['oldpeak'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[34]: <AxesSubplot:xlabel='oldpeak'>



In [35]:
```
# Oldpeak
Q1 = df.oldpeak.quantile(0.25)
Q3 = df.oldpeak.quantile(0.75)
Q1, Q3
```

Out[35]: (0.0, 1.6)

In [36]:
```
IQR = Q3 - Q1
IQR
```

Out[36]: 1.6

In [37]:
```
lower_limit = Q1 - 1.5*IQR
upper_limit = Q3 + 1.5*IQR
lower_limit, upper_limit
```

Out[37]: (-2.4000000000000004, 4.0)

In [38]:
```
df[df['oldpeak'] > upper_limit]
```

Out[38]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
204	62	0	0	160	164	0	0	145	0	6.2	0	3	3	0
221	55	1	0	140	217	0	1	111	1	5.6	0	0	3	0
250	51	1	0	140	298	0	1	122	1	4.2	1	3	3	0
291	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0

In [39]:
```
df = df[df['oldpeak'] < upper_limit]
```

Visulization

Distribution of data :

1- Dealing with discreate features using Probability Mass Function :

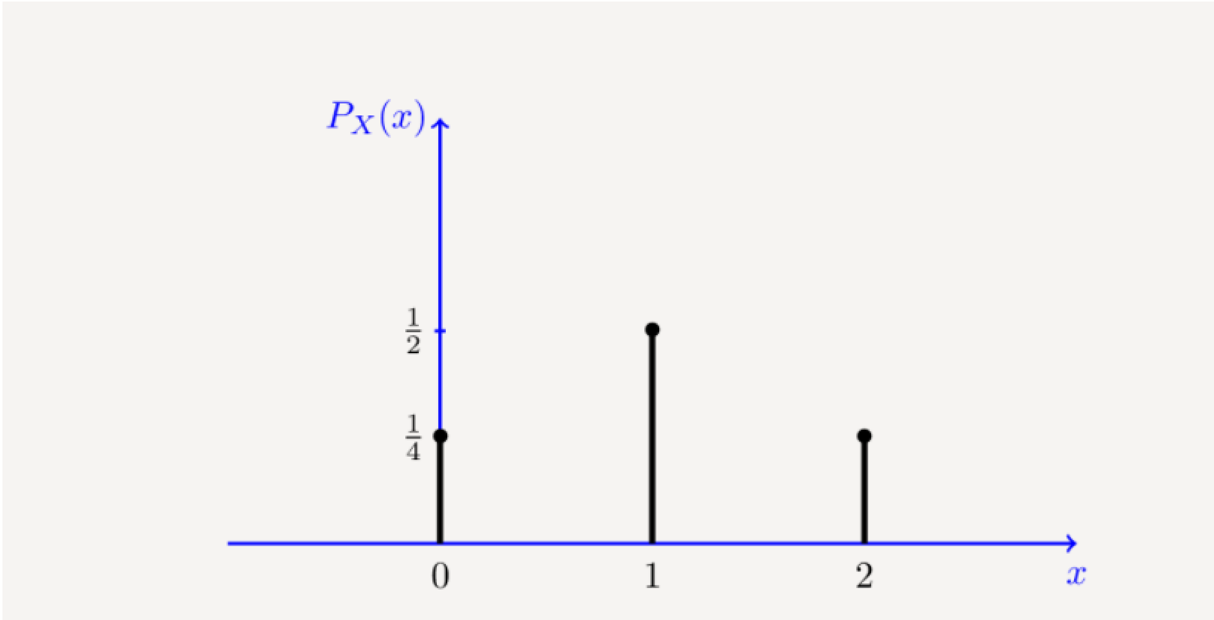
- **A discrete variable is a variable that can only take on a "countable" number of values. If you can count a set of items, then it’s a discrete variable.**
- **In statistics we represent a distribution of discrete variables with PMF's (Probability Mass Functions) and CDF's (Cumulative Distribution Functions).**
- **A probability mass function (pmf) is a function over the sample space of a discrete random variable X which gives the probability that X is equal to a certain value.**
- We have some discrete fetures such as sex, cp , fps , target.

Let X be a discrete random variable on a sample space S. Then the *probability mass function* $f(x)$ is defined as

$$f(x) = \mathrm{P}[X = x].$$

Each probability mass function satisfies the following two conditions:

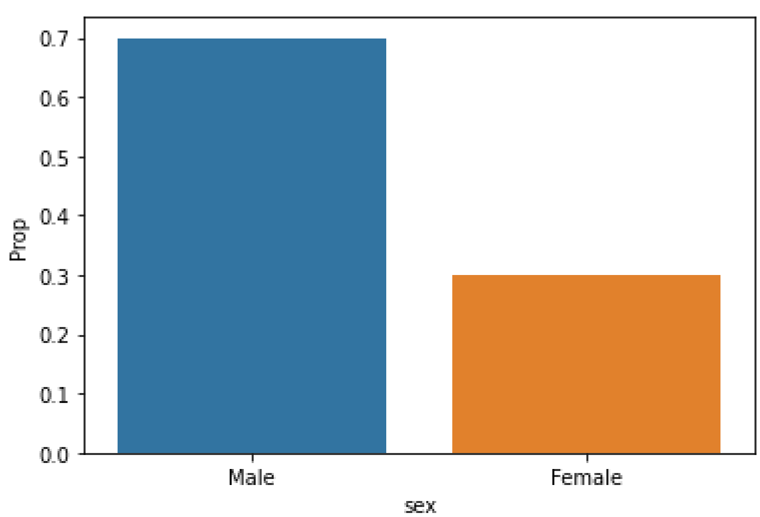
$$\begin{array}{ll} \text{(i)} & f(x) \geq 0 \text{ for all } x \in S, \\ \text{(ii)} & \sum_{x \in S} f(x) = 1. \end{array}$$



In [40]:
```
x = df['sex']
x = pd.DataFrame(x.value_counts())      # Make a new Data Frame for ( gender , val
length = len(df['sex'])                  # Total numbers of people (sex gender)
data = pd.DataFrame(x)
data.columns = ["Counts"]                # Rename the coulmn from Sex to Counts
data["Prop"] = data["Counts"]/ length  # Make a new column for probability for ec
sex = ["Male","Female"]
data['sex'] = sex
sns.barplot(data["sex"],data["Prop"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

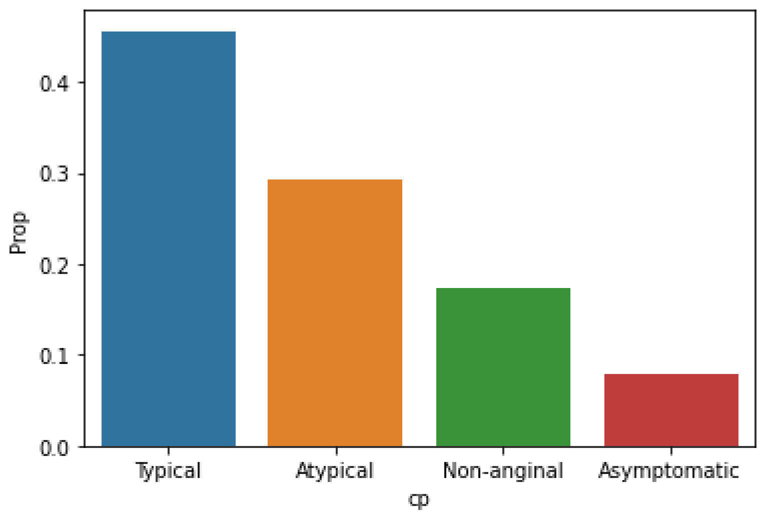
Out[40]: <AxesSubplot:xlabel='sex', ylabel='Prop'>



In [41]:
```
x = df['cp']
x = pd.DataFrame(x.value_counts()) #Make a new Data Frame for ( gender , value co
length = len(df['cp'])              # Total numbers of people (sex gender)
data = pd.DataFrame(x)
data.columns = ["Counts"]           # Rename the coulmn from Sex to Counts
data["Prop"] = data["Counts"]/ length  # Make a new column for probability for ea
cp = ["Typical","Atypical","Non-anginal","Asymptomatic"]
data['cp'] = cp
sns.barplot(data["cp"],data["Prop"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

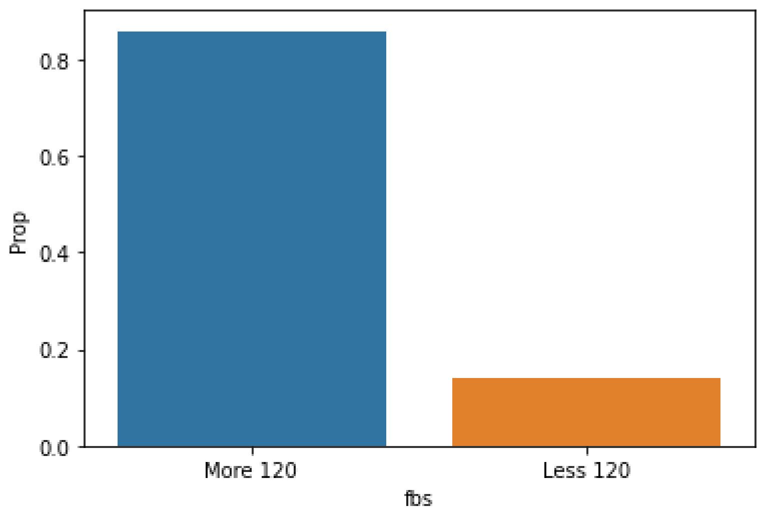
Out[41]: <AxesSubplot:xlabel='cp', ylabel='Prop'>



In [42]:
```
x = df['fbs']
x = pd.DataFrame(x.value_counts())
length = len(df['fbs'])
data = pd.DataFrame(x)
data.columns = ["Counts"]
data["Prop"] = data["Counts"]/ length
fbs = ["More 120","Less 120"]
data['fbs'] = fbs
sns.barplot(data["fbs"],data["Prop"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

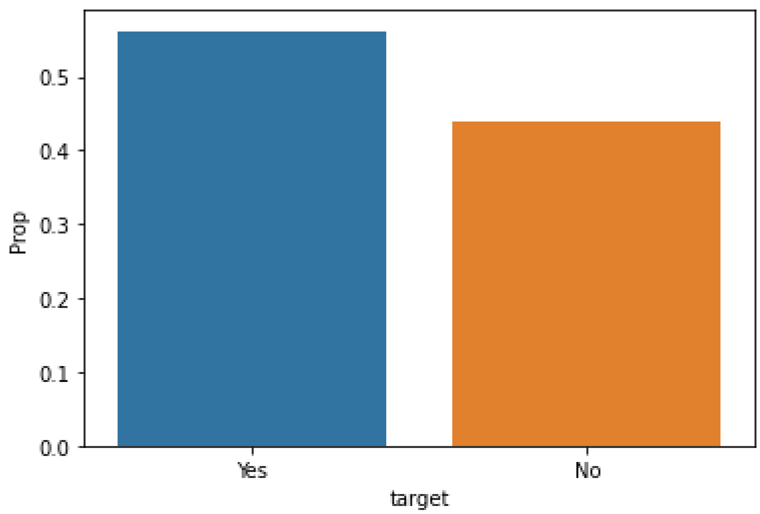
Out[42]:
```
<AxesSubplot:xlabel='fbs', ylabel='Prop'>
```



In [43]:
```
x = df['target']
x = pd.DataFrame(x.value_counts())      #Make a new Data Frame for ( gender , val
length = len(df['target'])                # Total numbers of people (sex gender)
data = pd.DataFrame(x)
data.columns = ["Counts"]                 # Rename the coulmn from Sex to Counts
data["Prop"] = data["Counts"]/ length    # Make a new column for probability for e
target = ["Yes","No"]
data['target'] = target
sns.barplot(data["target"],data["Prop"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[43]: <AxesSubplot:xlabel='target', ylabel='Prop'>

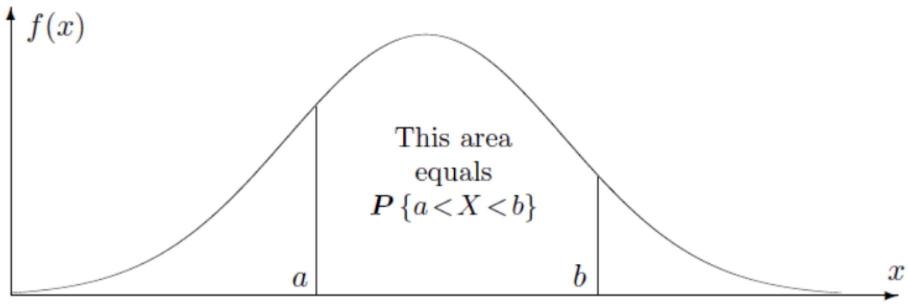


- Ohhh ! Our Problem looks balanced !!

2- Dealing with Continious features using Probability Denisty Function :

- **A continuous variable takes on an "uncountable" number of values. An example of a continuous variable is length. Length can be measured to an arbitrary degree and is therefore continuous.**
- **In statistics We represent distributions of continuous variables with PDF's (Probability Density Functions) and CDF's.**
- **Probability density function (PDF) is a statistical expression that defines a probability distribution (the likelihood of an outcome) for a discrete random variable (e.g., a stock or ETF) as opposed to a continuous random variable.**
- We have some continious features such as age, trestbps , chol , oldpeak.

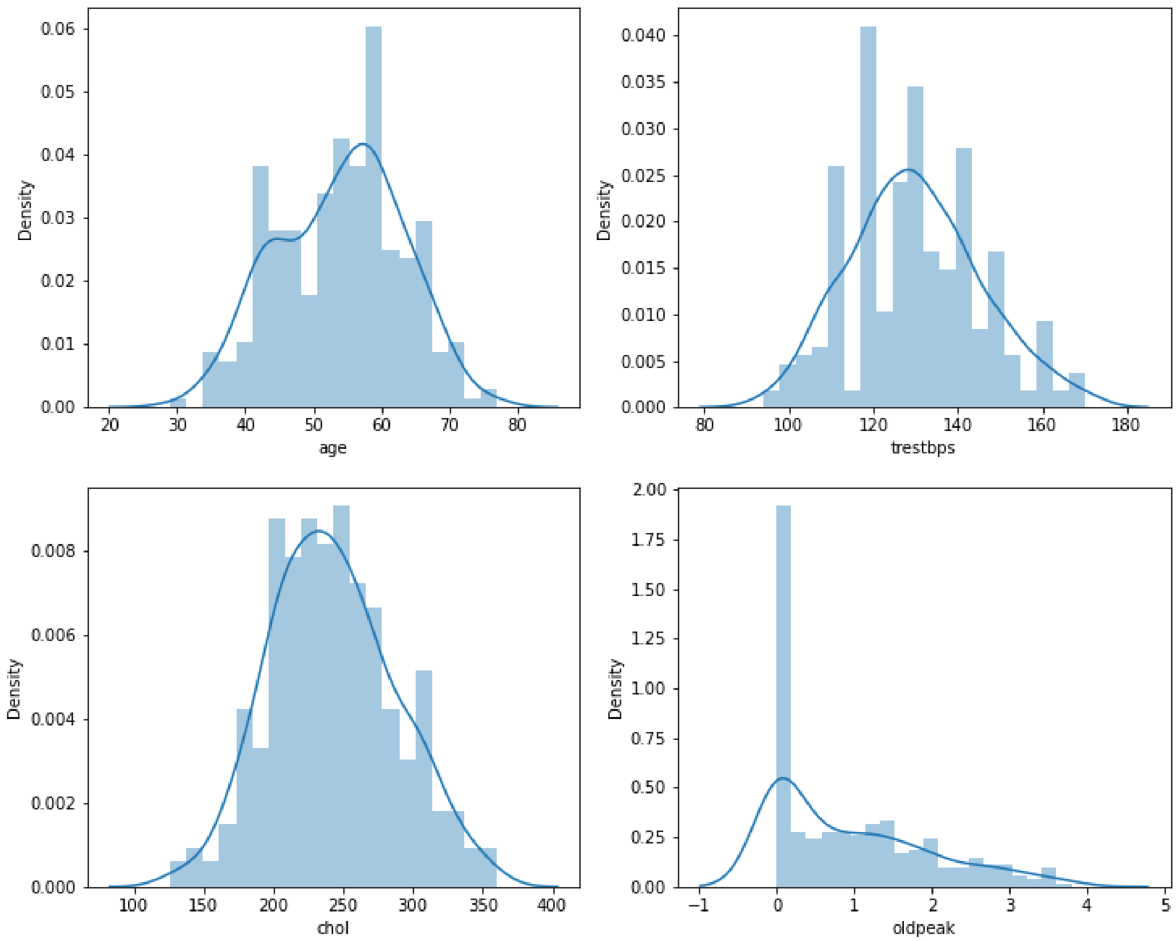
$$\int_a^b f(x)dx = F(b) - F(a) = P\{a < X < b\}$$



In [44]:
```
fig, ax = plt.subplots(2,2, figsize=(12,10))
sns.distplot(df.age, bins = 20, ax=ax[0,0])
sns.distplot(df.trestbps, bins = 20, ax=ax[0,1])
sns.distplot(df.chol, bins = 20, ax=ax[1,0])
sns.distplot(df.oldpeak, bins = 20, ax=ax[1,1])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[44]:
```
<AxesSubplot:xlabel='oldpeak', ylabel='Density'>
```



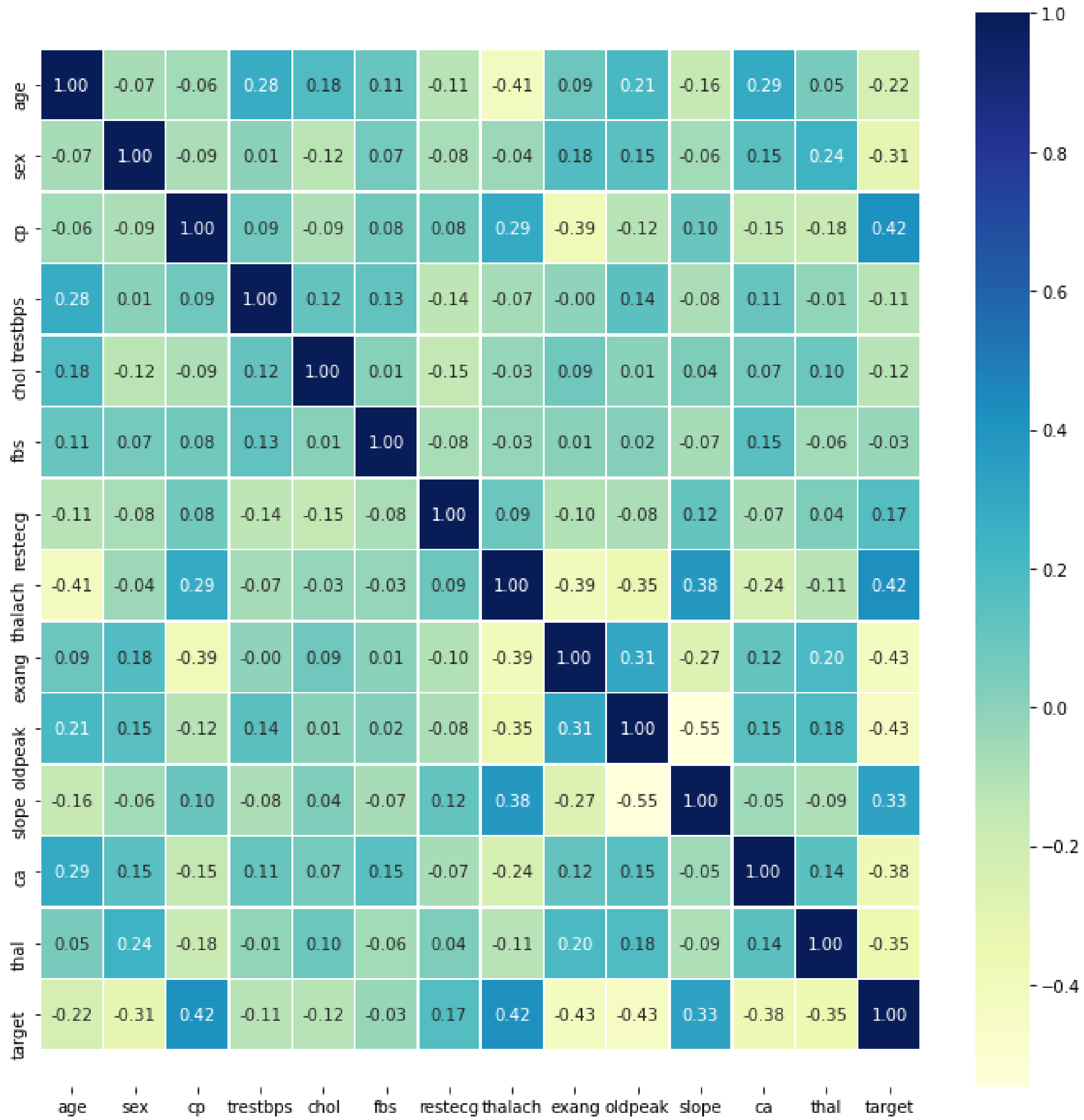
Other Visulizations :

Correlation Matrix :

- In probability theory and statistics, a covariance matrix is a square matrix giving the covariance between each pair of elements of a given random vector.

In [45]:
```
# Let's make our correlation matrix a little prettier
corr_matrix = df.corr()
fig, ax = plt.subplots(figsize=(12, 12))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Out[45]: (14.5, -0.5)



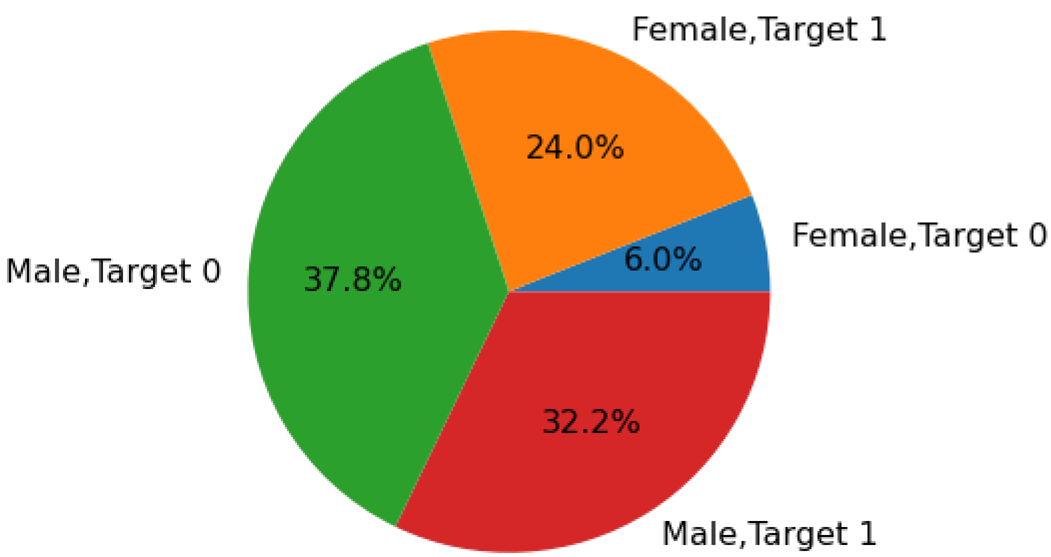
Conclusion:

- fbs and chol are the lowest correlated with the target variable.
- All other variables have a significant correlation with the target variable.

Gender vs Target :

💡

In [46]:
```
df_sex = df.groupby(["sex","target"]).size()
plt.pie(df_sex.values, labels = ["Female,Target 0", "Female,Target 1", "Male,Targ
plt.show()
```



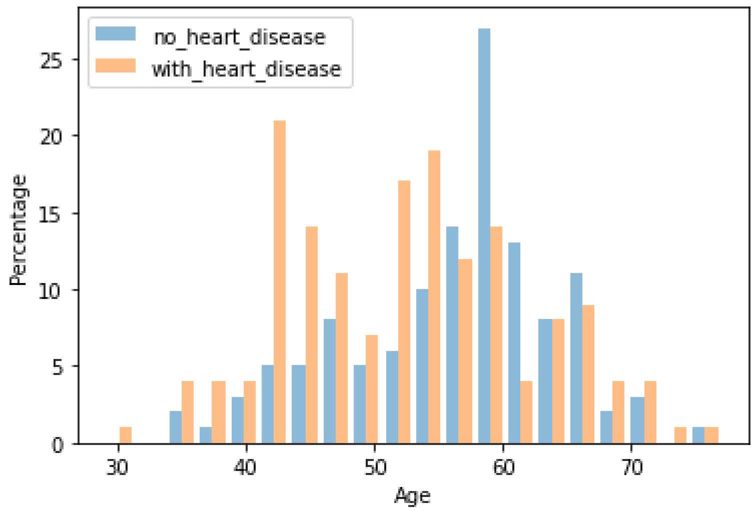
Conclusion:

- 🕵 The ratio of male has heart disease is 30.7%, a little bit higher than female.

Age vs Target :

💡

In [47]:
```
plt.hist([df[df.target==0].age, df[df.target==1].age], bins = 20, alpha = 0.5, la
plt.xlabel("Age")
plt.ylabel("Percentage")
plt.legend()
plt.show()
```



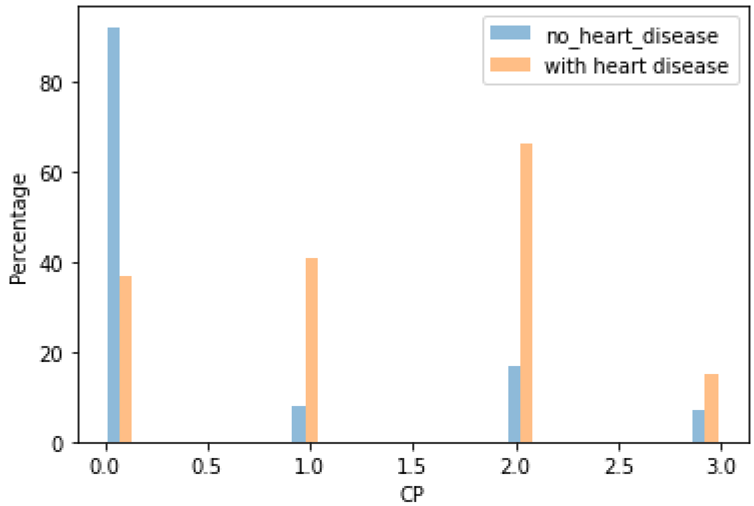
Conclusion:

- 🕵 The ratio get higher over the age of forty. That is, people who is over forty is under high risk of heart disease.

CP vs Target :

💡

In [48]:
```
plt.hist([df[df.target==0].cp, df[df.target==1].cp], bins = 20, alpha = 0.5, labe
plt.xlabel("CP")
plt.ylabel("Percentage")
plt.legend()
plt.show()
```



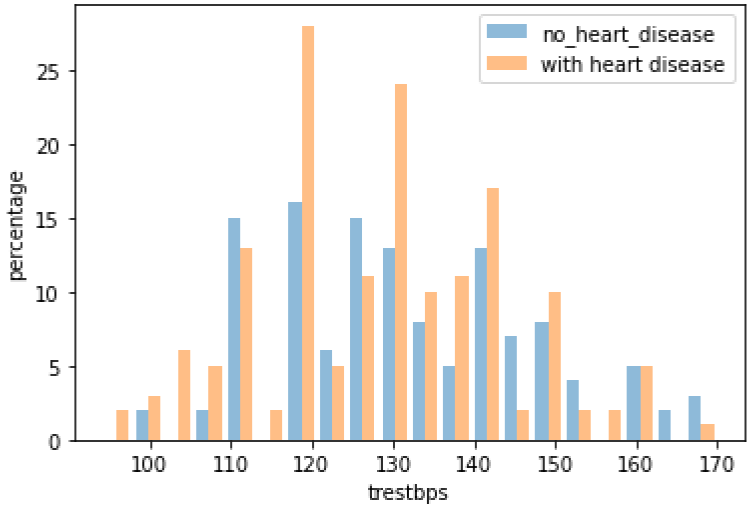
Conclusion:

- 🚨 cp {Chest Pain} : People with cp equl to 1, 2, 3 are more likely to have heart disease than people with cp equal to 0.

Trestbps vs Target :

💡

In [49]:
```
plt.hist([df[df.target==0].trestbps, df[df.target==1].trestbps], bins = 20, alpha
plt.xlabel("trestbps")
plt.ylabel("percentage")
plt.legend()
plt.show()
```



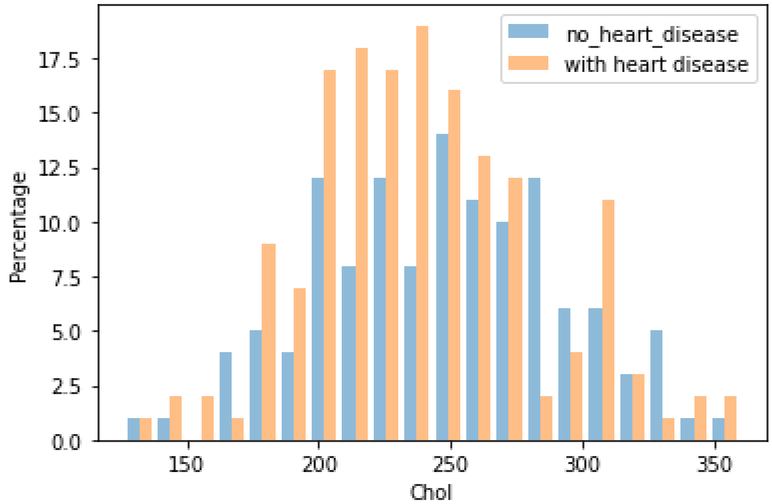
Conclusion:

- 🚨 The ideal blood pressure should be lower than 120 mmHg. Whether the patients have heart disease or not , over 50% patients have higher blood pressure.

Chol vs Target :

💡

In [50]:
```
plt.hist([df[df.target==0].chol, df[df.target==1].chol], bins = 20, alpha = 0.5,
plt.xlabel("Chol")
plt.ylabel("Percentage")
plt.legend()
plt.show()
```



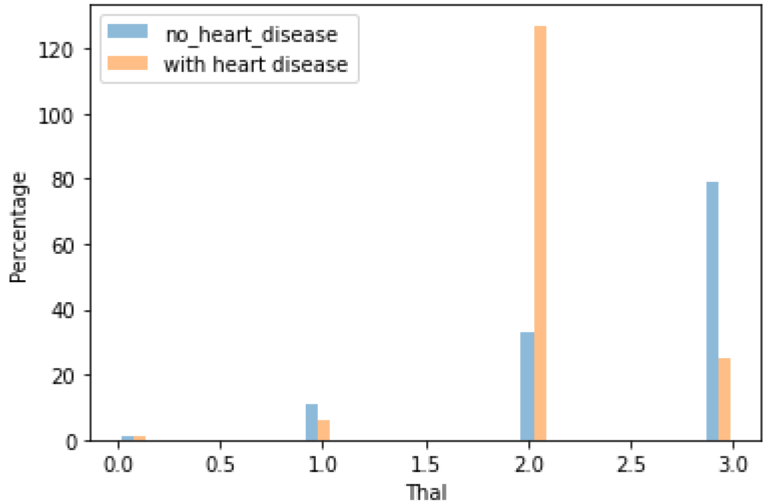
Conclusion:

- 🚑 Also, amounts of people having heart disease are over 200mg/dl of chol. According to the research, the normal value of chol should be lower than 200mg/dl.

Thal vs Target :

💡

In [51]:
```
plt.hist([df[df.target==0].thal, df[df.target==1].thal], bins = 20, alpha = 0.5,
plt.xlabel("Thal")
plt.ylabel("Percentage")
plt.legend()
plt.show()
```



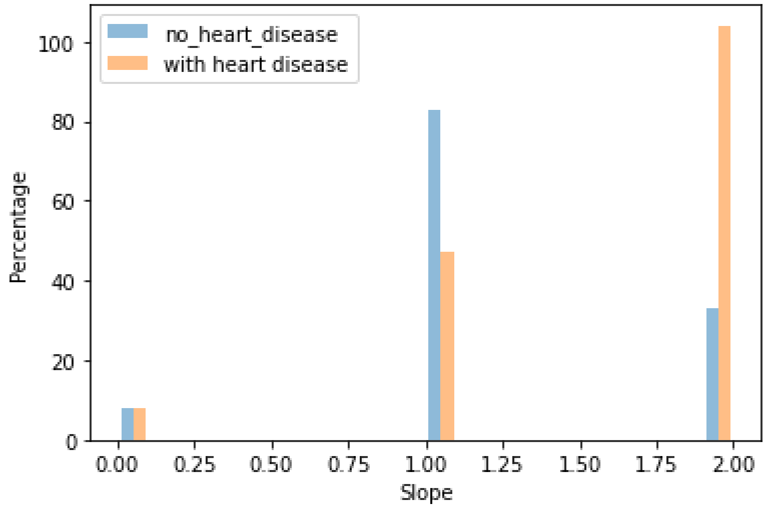
Conclusion:

- 🕵 People with thal value equal to 2 (fixed defect: used to be defect but ok now) are more likely to have heart disease.

Slope vs Target :

💡

In [52]:
```
plt.hist([df[df.target==0].slope, df[df.target==1].slope], bins = 20, alpha = 0.5
plt.xlabel("Slope")
plt.ylabel("Percentage")
plt.legend()
plt.show()
```

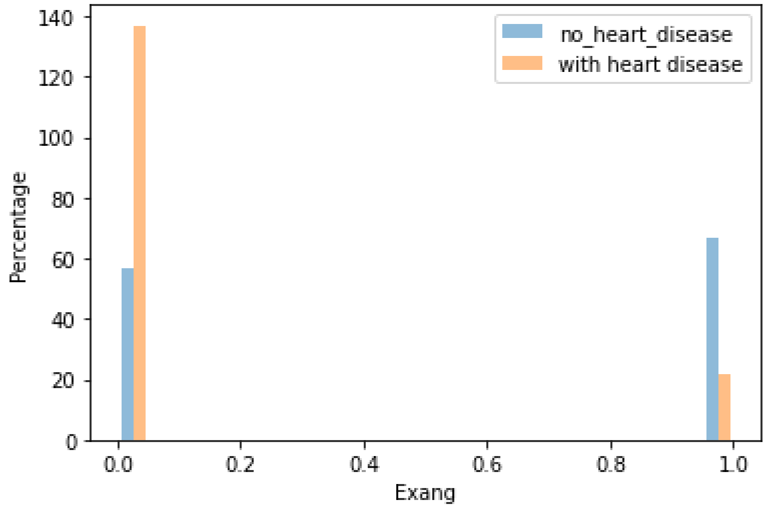


Conclusion:

- People with slope value equal to 2 (Downslopins: signs of unhealthy heart) are more likely to have heart disease than people with slope value equal to 0 (Upsloping: better heart rate with excercise) or 1 (Flatsloping: minimal change (typical healthy heart)).

Exang vs Target :

In [53]:
```
plt.hist([df[df.target==0].exang, df[df.target==1].exang], bins = 20, alpha = 0.5
plt.xlabel("Exang")
plt.ylabel("Percentage")
plt.legend()
plt.show()
```



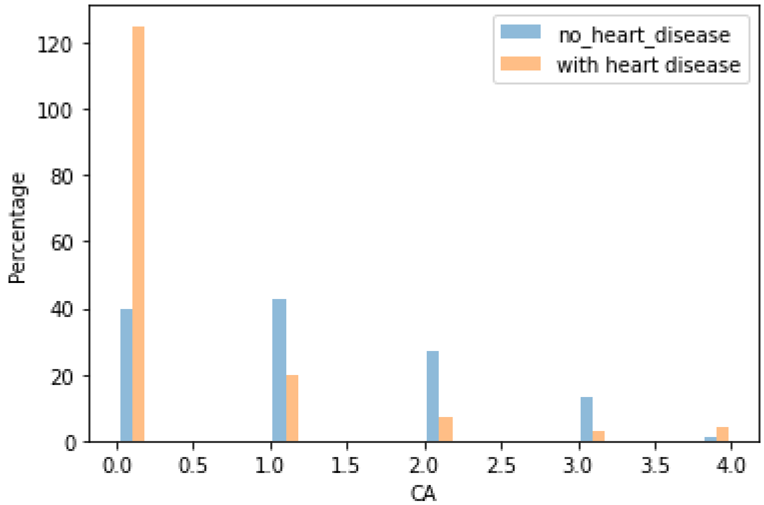
Conclusion:

- 🕮 People with value 0 (No ==> exercice induced angina) have heart disease more than people with value 1 (Yes ==> exercice induced angina)

CA vs Target :

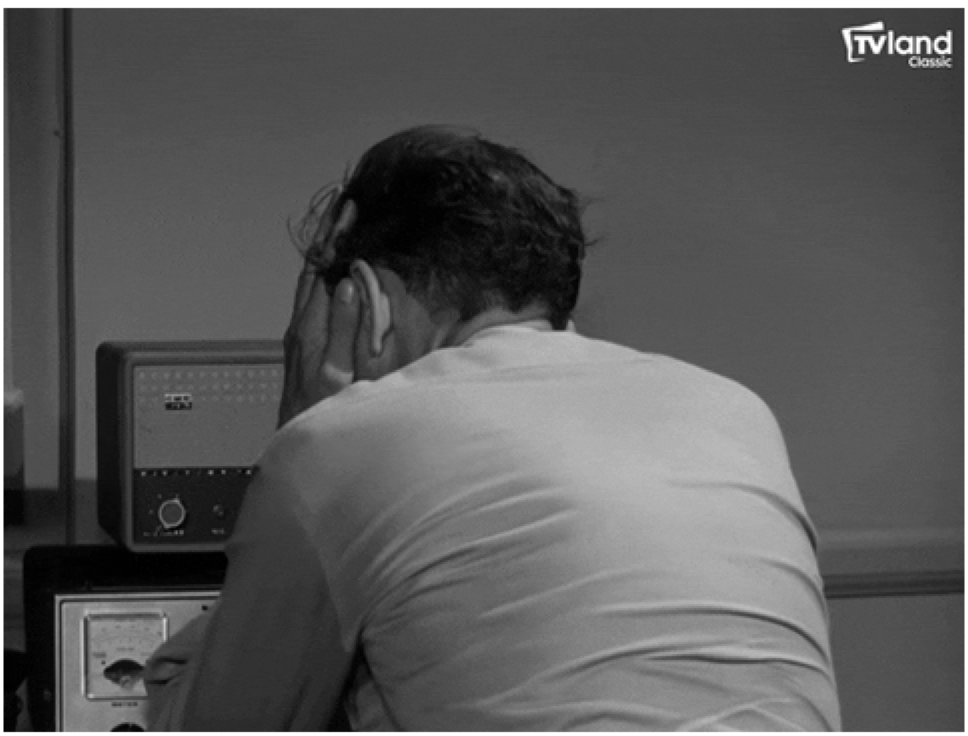
💡

In [54]:
```
plt.hist([df[df.target==0].ca, df[df.target==1].ca], bins = 20, alpha = 0.5, labe
plt.xlabel("CA")
plt.ylabel("Percentage")
plt.legend()
plt.show()
```



Conclusion:

- The more blood movement the better so people with ca equal to 0 are more likely to have heart disease.



Modeling

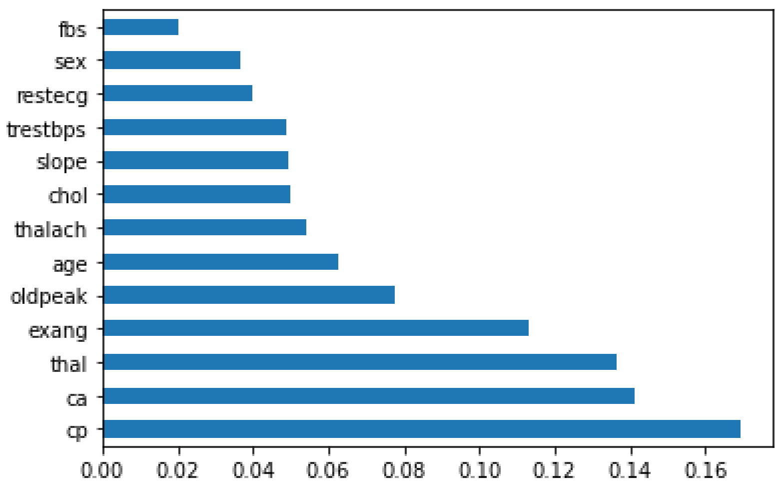
feature selection

In [300]:
```
from sklearn.ensemble import ExtraTreesRegressor
```

In [301]:
```
x = df.iloc[:, :-1]
y = df.iloc[:,-1]
```

In [302]:
```
model = ExtraTreesRegressor()
feat_imp = model.fit(x, y)
feat_imp.feature_importances_
imp = pd.Series(feat_imp.feature_importances_, index = x.columns)
imp.nlargest(14).plot(kind = 'barh')
```

Out[302]: <AxesSubplot:>



In [303]:
```
x = np.array(df[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thala
       'exang', 'oldpeak', 'slope', 'ca', 'thal']])
y = np.array(df['target'])
```

In [304]:
```
model = ExtraTreesClassifier()
model.fit(x, y)
print(model.feature_importances_)
```

```
[0.07019788 0.06127766 0.12780186 0.06368515 0.06120136 0.02112841
 0.03363597 0.08791796 0.09404421 0.09198112 0.06144244 0.11963547
 0.1060505 ]
```

knn

In [305]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x,y)
```

Out[305]:
```
KNeighborsClassifier()
```

In [306]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.4,random_state=5)
```

In [307]:
```
confusion_matrix(y_test,knn.predict(x_test))
```

Out[307]:
```
array([[34, 19],
       [ 9, 52]], dtype=int64)
```

In [308]:
```
df.columns
```

Out[308]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [309]:
```
print(knn.score(x_train,y_train))
print(knn.score(x_test,y_test))
```

```
0.7514792899408284
0.7543859649122807
```

In [310]:
```
prediction_lr=knn.predict(x_test)
print('__________________________')
print('\n clasification report:\n', classification_report(y_test,prediction_lr))
print('__________________________')
```

```
__________________________

 clasification report:
               precision    recall  f1-score   support

           0       0.79      0.64      0.71        53
           1       0.73      0.85      0.79        61

    accuracy                           0.75       114
   macro avg       0.76      0.75      0.75       114
weighted avg       0.76      0.75      0.75       114

__________________________
```

logestic regression

In [311]:
```
classifier = LogisticRegression(solver='lbfgs',random_state=0)
```

In [312]:
```
classifier.fit(x_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
3: 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 (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-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

Out[312]:
```
LogisticRegression(random_state=0)
```

In [313]:
```
print(classifier.score(x_train,y_train))
print(classifier.score(x_test,y_test))
```

```
0.8698224852071006
0.8508771929824561
```

In [314]:
```
prediction_lr=classifier.predict(x_test)
print('__________________________')
print('\n clasification report:\n', classification_report(y_test,prediction_lr))
print('__________________________')
```

```
__________________________

 clasification report:
               precision    recall  f1-score   support

           0       0.93      0.74      0.82        53
           1       0.81      0.95      0.87        61

    accuracy                           0.85       114
   macro avg       0.87      0.84      0.85       114
weighted avg       0.86      0.85      0.85       114

__________________________
```

Random Forest classifier

In [315]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
rf_tuned = RandomForestClassifier()
```

In [316]:
```
rf_tuned = rf_tuned.fit(x,y)
```

In [323]:
```
cross_val_score(rf_tuned, x, y, cv = 10).mean()
```

Out[323]:
```
0.8442118226600985
```

In [324]:
```
print(rf_tuned.score(x_train,y_train))
print(rf_tuned.score(x_test,y_test))
```

```
1.0
1.0
```

In [325]:
```
prediction_lr=rf_tuned.predict(x_test)
print('__________________________')
print('\n clasification report:\n', classification_report(y_test,prediction_lr))
print('__________________________')
```

```
__________________________

 clasification report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        53
           1       1.00      1.00      1.00        61

    accuracy                           1.00       114
   macro avg       1.00      1.00      1.00       114
weighted avg       1.00      1.00      1.00       114

__________________________
```

svm

In [99]:
```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
```

In [109]:
```
from sklearn import svm
clf = svm.SVC(kernel='rbf')
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
```

In [110]:
```
y_pred = clf.predict(x_test)
y_pred
```

Out[110]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 0, 1, 1], dtype=int64)
```

In [117]:
```
cm = confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[23 12]
 [ 2 34]]
```

In [113]:
```
print(clf.score(x_train,y_train))
print(clf.score(x_test,y_test))
```

```
0.9339622641509434
0.8028169014084507
```