In [1]:
```
import os
os.getcwd()
```

Out[1]:
```
'C:\\Users\\urkha\\ML_Assignments'
```

In [2]:
```
# It is used to work on structured data that is for example this Heart.csv data which is in Table
import pandas as pd
```

In [5]:
```
# import the Dataset
df= pd.read_csv('Heart.csv')
# Here df is the Data Frame which is nothing but Data Table
```

In [7]:
```
# To get first 5 lines
df.head()
```

Out[7]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No

In [8]:
```
# To find the Shape of the Data
df.shape
```

Out[8]:
```
(303, 15)
```

In [9]:
```
# To find Missing Values
df.isnull()
# It shows True for the values that are NULL
```

Out[9]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
299	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
300	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
301	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
302	False	False	False	False	False	False	False	False	False	False	False	False	True	False	False

303 rows × 15 columns

In [10]:
```
# To find Missing Values
df.isnull().sum()
# Here False=0 and True=1
```

Out[10]:
```
Unnamed: 0    0
Age           0
Sex           0
ChestPain     0
RestBP        0
Chol          0
Fbs           0
RestECG       0
MaxHR         0
ExAng         0
Oldpeak       0
Slope         0
Ca            4
Thal          2
AHD           0
dtype: int64
```

In [11]:
```
# To find Missing Values. Here count() gives the Not NULL values
df.count()
```

Out[11]:
```
Unnamed: 0    303
Age           303
Sex           303
ChestPain     303
RestBP        303
Chol          303
Fbs           303
RestECG       303
MaxHR         303
ExAng         303
Oldpeak       303
Slope         303
Ca            299
Thal          301
AHD           303
dtype: int64
```

In [12]:
```
# To find Datatype of each column
df.dtypes
```

Out[12]:
```
Unnamed: 0      int64
Age             int64
Sex             int64
ChestPain      object
RestBP          int64
Chol            int64
Fbs             int64
RestECG         int64
MaxHR           int64
ExAng           int64
Oldpeak       float64
Slope           int64
Ca            float64
Thal           object
AHD            object
dtype: object
```

In [13]:
```
# To find out Zero's. Here we use Boolean filtering
df==0
```

Out[13]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	False	False	False	False	False	False	False	False	False	True	False	False	True	False	False
1	False	False	False	False	False	False	True	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	True	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	True	True	False	True	False	False	True	False	False
4	False	False	True	False	False	False	True	False	False	True	False	False	True	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	False	False	False	False	False	False	True	True	False	True	False	False	True	False	False
299	False	False	False	False	False	False	False	True	False	True	False	False	False	False	False
300	False	False	False	False	False	False	True	True	False	False	False	False	False	False	False
301	False	False	True	False	False	False	True	False	False	True	True	False	False	False	False
302	False	False	False	False	False	False	True	True	False	True	True	False	False	False	False

303 rows × 15 columns

In [14]:
```
# If we want to Highlight the Zero's then use
df[df==0]
```

Out[14]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	NaN	0.0	NaN	NaN	0.0	NaN	NaN
4	NaN	NaN	0.0	NaN	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	NaN	0.0	NaN	NaN	0.0	NaN	NaN
299	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0	NaN	0.0	NaN	NaN	NaN	NaN	NaN
300	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
301	NaN	NaN	0.0	NaN	NaN	NaN	0.0	NaN	NaN	0.0	0.0	NaN	NaN	NaN	NaN
302	NaN	NaN	NaN	NaN	NaN	NaN	0.0	0.0	NaN	0.0	0.0	NaN	NaN	NaN	NaN

303 rows × 15 columns

In [15]:
```
# Also if we want to get the number of times Zero then use
df[df==0].count()
```

Out[15]:
```
Unnamed: 0      0
Age             0
Sex            97
ChestPain       0
RestBP          0
Chol            0
Fbs           258
RestECG       151
MaxHR           0
ExAng         204
Oldpeak        99
Slope           0
Ca            176
Thal            0
AHD             0
dtype: int64
```

In [16]:
```
# To get the column names
df.columns
```

Out[16]:
```
Index(['Unnamed: 0', 'Age', 'Sex', 'ChestPain', 'RestBP', 'Chol', 'Fbs',
       'RestECG', 'MaxHR', 'ExAng', 'Oldpeak', 'Slope', 'Ca', 'Thal', 'AHD'],
      dtype='object')
```

In [17]:
```
# To access a particular. Here it is known as Label Based Slicing
df['Age']
```

Out[17]:
```
0      63
1      67
2      67
3      37
4      41
      ..
298    45
299    68
300    57
301    57
302    38
Name: Age, Length: 303, dtype: int64
```

In [18]:
```
# To find the Mean of the Age column
df['Age'].mean()
```

Out[18]:
```
np.float64(54.43894389438944)
```

In [20]:
```
# To extract only Age, Sex, ChestPain, RestBP, Chol.  But for that we have to write it in "Double Square Bracke
newdf=df[['Age', 'Sex', 'ChestPain', 'RestBP', 'Chol']]
```

In [21]:
```
newdf
```

Out[21]:

	Age	Sex	ChestPain	RestBP	Chol
0	63	1	typical	145	233
1	67	1	asymptomatic	160	286
2	67	1	asymptomatic	120	229
3	37	1	nonanginal	130	250
4	41	0	nontypical	130	204
...	...	...	...	...	...
298	45	1	typical	110	264
299	68	1	asymptomatic	144	193
300	57	1	asymptomatic	130	131
301	57	0	nontypical	130	236
302	38	1	nonanginal	138	175

303 rows × 5 columns

In [22]:
```
# To randomly divide the dataset into 75% Training Dataset and 25% Testing Dtaset. This method is known as "Cros
from sklearn.model_selection import train_test_split
```

In [23]:
```
train, test= train_test_split(df, random_state=0, test_size=0.25)
```

In [24]:
```
train.shape
```

Out[24]:
```
(227, 15)
```

In [25]:
```
test.shape
```

Out[25]:
```
(76, 15)
```

In [26]:
```
# Now from here second part starts
```

In [27]:
```
import numpy as np
# Here we use numpy in order to create the Array
```

In [28]:
```
actual= list(np.ones(45))+list(np.zeros(55))
```

In [29]:
```
np.array(actual)
```

Out[29]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

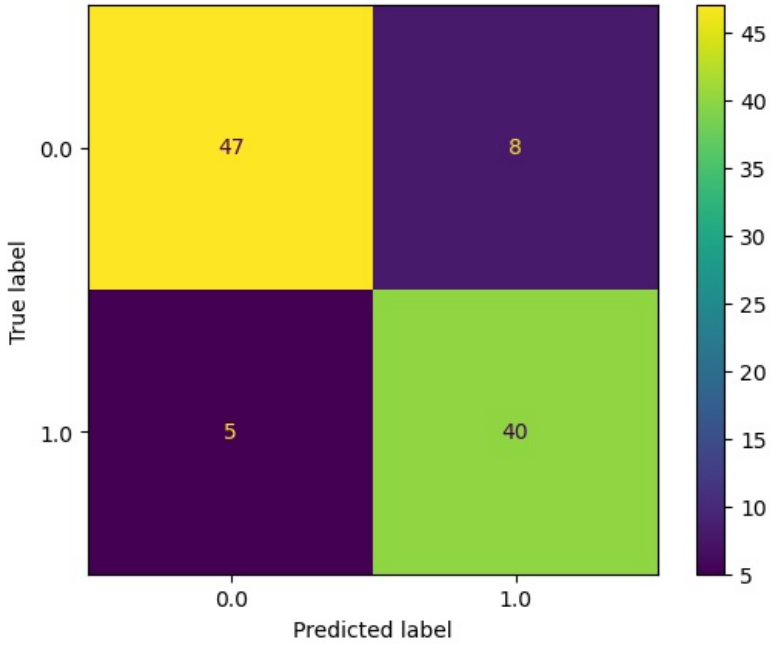
In [31]:
```
predicted= list(np.ones(40))+list(np.zeros(52))+list(np.ones(8))
np.array(predicted)
```

Out[31]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [33]:
```
from sklearn.metrics import ConfusionMatrixDisplay
# Here metrics is a package of the sklearn and ConfusionMatrixDisplay is a Class
```

In [34]:
```
ConfusionMatrixDisplay.from_predictions(actual,predicted)
# Here from_predictions is the method of the ConfusionMatrixDisplay Class
```

Out[34]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29b6819e7b0>
```



In [35]:
```
from sklearn.metrics import classification_report
# Here classification_report is a Function in the metrics package
```

In [36]:
```
print(classification_report(actual,predicted))
```

```
              precision    recall  f1-score   support

         0.0       0.90      0.85      0.88        55
         1.0       0.83      0.89      0.86        45

    accuracy                           0.87       100
   macro avg       0.87      0.87      0.87       100
weighted avg       0.87      0.87      0.87       100
```

In [39]:
```
from sklearn.metrics import accuracy_score
accuracy_score(actual,predicted)
# It is Optional
```

Out[39]: 0.87

In [ ]: