```
import pandas as pd
import numpy as np


df=pd.read_csv("/content/Heart.csv")


df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  303 non-null    int64  
 1   Age         303 non-null    int64  
 2   Sex         303 non-null    int64  
 3   ChestPain   303 non-null    object 
 4   RestBP      303 non-null    int64  
 5   Chol        303 non-null    int64  
 6   Fbs         303 non-null    int64  
 7   RestECG     303 non-null    int64  
 8   MaxHR       303 non-null    int64  
 9   ExAng       303 non-null    int64  
 10  Oldpeak     303 non-null    float64
 11  Slope       303 non-null    int64  
 12  Ca          299 non-null    float64
 13  Thal        301 non-null    object 
 14  AHD         303 non-null    object 
dtypes: float64(2), int64(10), object(3)
memory usage: 35.6+ KB
```

```
shape=df.shape
print("The Shape of the data is:",shape)
```

```
The Shape of the data is: (303, 15)
```

```
missing_value=df.isnull()
print("The missing values are:",missing_value)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-ae3dccd35724> in <cell line: 1>()
----> 1 missing_value=df.isnull()
      2 print("The missing values are:",missing_value)

NameError: name 'df' is not defined
```

SEARCH STACK OVERFLOW

```
sum=df.isnull().sum()
print("The sum of miising value in each column is\n:",sum)
```

```
The sum of miising value in each column is
: 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
```

```
type=df.dtypes
print("The Data Type is\n:",type)
```

```
The Data Type is
: 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
```

```
zero=(df==0)
print("The zeros are:\n",zero)
```

```
The zeros are:
      Unnamed: 0    Age    Sex  ChestPain  RestBP   Chol    Fbs  RestECG  \
0         False  False  False      False   False  False  False    False
1         False  False  False      False   False  False   True    False
2         False  False  False      False   False  False   True    False
3         False  False  False      False   False  False   True     True
4         False  False   True      False   False  False   True    False
..          ...    ...    ...        ...     ...    ...    ...      ...
298       False  False  False      False   False  False   True     True
299       False  False  False      False   False  False  False     True
300       False  False  False      False   False  False   True     True
301       False  False   True      False   False  False   True    False
302       False  False  False      False   False  False   True     True

     MaxHR  ExAng  Oldpeak  Slope     Ca   Thal    AHD
0    False   True    False  False   True  False  False
1    False  False    False  False  False  False  False
2    False  False    False  False  False  False  False
3    False   True    False  False   True  False  False
4    False   True    False  False   True  False  False
..     ...    ...      ...    ...    ...    ...    ...
298  False   True    False  False   True  False  False
299  False   True    False  False  False  False  False
300  False  False    False  False  False  False  False
301  False   True     True  False  False  False  False
302  False   True     True  False  False  False  False

[303 rows x 15 columns]
```

```
df.columns
```

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

```
age=df['Age'].mean()
print("The mean is:\n",age)
```

```
The mean is:
 54.43894389438944
```

```
newdf=df[['Age', 'Sex','ChestPain','RestBP','Chol']]
print("The extracted columns are:\n",newdf)
```

```
The extracted columns are:
      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 x 5 columns]
```

```
from sklearn.model_selection import train_test_split
train,test=train_test_split(df,random_state=0,test_size=0.25)
```

```
train_shape=train.shape
print("The shape of training is:",train_shape)
```

```
The shape of training is: (227, 15)
```

```
test_shape=test.shape
print("The shape of testing is:",test_shape)
```

```
The shape of testing is: (76, 15)
```

```
#numpay library is used for array.
print("The actual Data is:\n")
actual=list(np.ones(45))+list(np.zeros(55))
np.array(actual)
```

```
The actual Data is:

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.])
```

```
print("The predicted Data is:\n")
predicted=list(np.ones(40))+list(np.zeros(52))+list(np.ones(8))
np.array(predicted)
```

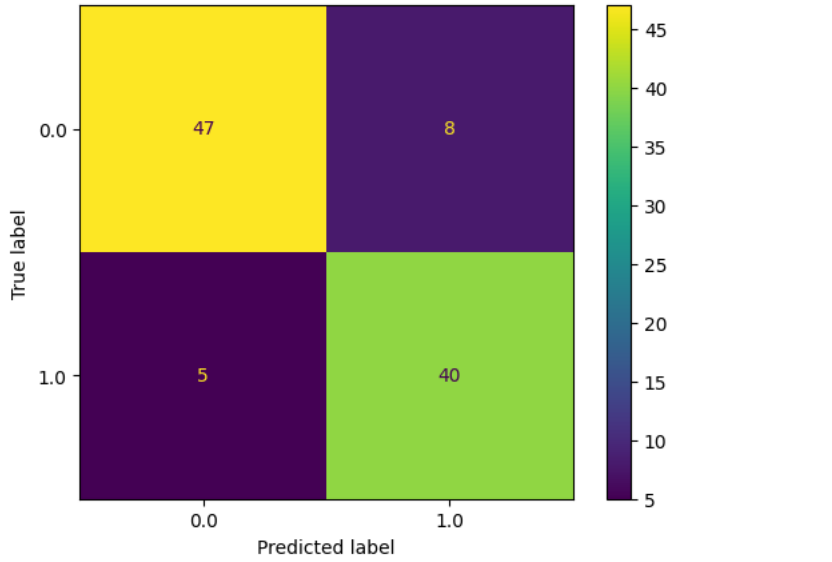
```
The predicted Data is:

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., 1., 1., 1., 1., 1., 1., 1.])
```

```
from sklearn.metrics import ConfusionMatrixDisplay
```

```
ConfusionMatrixDisplay.from_predictions(actual,predicted)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7c0d36ddbf10>
```




```
from sklearn.metrics import classification_report
```

```
print("All Classification report is:\n")
print(classification_report(actual,predicted))
```

```
All Classification report is:

              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
```