Practical No. 11

Aim: To perform and Data analysis with Confusion matrix

In [3]:
```
#Name : Taufiq Rafik Nagori
#Roll no. : 77 (BDA-B77)
#Section : B
#Subject : PE-II
```

In [5]:
```
import os
import pandas as pd
import numpy as np
```

In [7]:
```
os.getcwd()
```

Out[7]:
```
'C:\\Users\\USER'
```

In [9]:
```
os.chdir("C:\\Users\\USER\\Desktop")
```

In [11]:
```
data=pd.read_csv("heart.csv")
```

In [13]:
```
data.head()
```

Out[13]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

In [15]:
```
data.tail()
```

Out[15]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

In [17]:

```
data.info
```

Out[17]:

```
<bound method DataFrame.info of       age  sex  cp  trestbps  chol  fbs  restecg  thalac
h  exang  oldpeak  \
0      52    1   0       125   212    0        1      168      0      1.0
1      53    1   0       140   203    1        0      155      1      3.1
2      70    1   0       145   174    0        1      125      1      2.6
3      61    1   0       148   203    0        1      161      0      0.0
4      62    0   0       138   294    1        1      106      0      1.9
...   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
1020   59    1   1       140   221    0        1      164      1      0.0
1021   60    1   0       125   258    0        0      141      1      2.8
1022   47    1   0       110   275    0        0      118      1      1.0
1023   50    0   0       110   254    0        0      159      0      0.0
1024   54    1   0       120   188    0        1      113      0      1.4

      slope  ca  thal  target
0         2   2     3       0
1         0   0     3       0
2         0   0     3       0
3         2   1     3       0
4         1   3     2       0
...     ...  ..   ...     ...
1020      2   0     2       1
1021      1   1     3       0
1022      1   1     2       0
1023      2   0     2       1
1024      1   1     3       0

[1025 rows x 14 columns]>
```

In [19]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [21]:

```
data.describe()
```

Out[21]:

	age	sex	cp	trestbps	chol	fbs	restecg	
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	102
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	14
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	2
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	7
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	13
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	15
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	16
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	20

In [23]:

```
data.shape
```

Out[23]:

```
(1025, 14)
```

In [25]:

```
data.size
```

Out[25]:

```
14350
```

In [27]:

```
data.ndim
```

Out[27]:

```
2
```

In [29]:

```
data.columns
```

Out[29]:

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

Data pre-processing, data-cleaning, mising value treatment

In [32]:

```
data.isna()
```

Out[32]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	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
2	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

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
4	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	False	False	False	False	False	False	False	False	False	False	False	False	False
1021	False	False	False	False	False	False	False	False	False	False	False	False	False
1022	False	False	False	False	False	False	False	False	False	False	False	False	False
1023	False	False	False	False	False	False	False	False	False	False	False	False	False
1024	False	False	False	False	False	False	False	False	False	False	False	False	False

1025 rows × 14 columns

In [34]:

```
data.isna().any()
```

Out[34]:

```
age         False
sex         False
cp          False
trestbps    False
chol        False
fbs         False
restecg     False
thalach     False
exang       False
oldpeak     False
slope       False
ca          False
thal        False
target      False
dtype: bool
```

In [36]:

```
data.isna().sum()
```

Out[36]:

```
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
```

Remove Duplicates

In [39]:

```
data_dup =data.duplicated().any()
```

```
In [41]:
data_dup
Out[41]:
True
```

```
In [43]:
data=data.drop_duplicates()
```

```
In [45]:
data_dup =data.duplicated().any()
```

```
In [47]:
data_dup
Out[47]:
False
```

Splitting dataset into training and testing

```
In [50]:
x = data.drop('target', axis=1)
y = data['target']
```

```
In [52]:
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2 ,random_state=42)
```

```
In [54]:
x_train
Out[54]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
163	48	1	0	124	274	0	0	166	0	0.5	1	0	3
291	58	1	0	128	259	0	0	130	1	3.0	1	2	3
280	45	0	1	130	234	0	0	175	0	0.6	1	0	2
85	44	1	1	120	220	0	1	170	0	0.0	2	0	2
239	62	0	0	150	244	0	1	154	1	1.4	1	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
267	67	1	0	120	237	0	1	71	0	1.0	1	0	2
77	63	1	0	140	187	0	0	144	1	4.0	2	2	3
125	60	0	3	150	240	0	1	171	0	0.9	2	0	2
522	67	0	2	152	277	0	1	172	0	0.0	2	1	2
119	42	1	1	120	295	0	1	162	0	0.0	2	0	2

241 rows × 13 columns

```
In [56]:
x_test
Out[56]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
245	44	1	1	130	219	0	0	188	0	0.0	2	0	2
349	62	0	2	130	263	0	1	97	0	1.2	1	1	3
135	58	0	0	170	225	1	0	146	1	2.8	1	2	1
389	63	1	3	145	233	1	0	150	0	2.3	0	0	1
66	53	1	2	130	197	1	0	152	0	1.2	0	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
402	70	1	1	156	245	0	0	143	0	0.0	2	0	2
123	65	0	2	140	417	1	0	157	0	0.8	2	1	2
739	52	1	0	128	255	0	1	161	1	0.0	2	1	3
274	66	1	0	160	228	0	0	138	0	2.3	2	0	1
256	35	0	0	138	183	0	1	182	0	1.4	2	0	2

61 rows × 13 columns

```
In [58]:
y_train
Out[58]:
163    0
291    0
280    1
85     1
239    0
      ..
267    0
77     0
125    1
522    1
119    1
Name: target, Length: 241, dtype: int64
In [60]:
y_test
Out[60]:
245    1
349    0
135    0
389    1
66     1
      ..
402    1
123    1
739    0
274    1
```

```
256    1
Name: target, Length: 61, dtype: int64
```

Logistic Regression

In [63]:

```
from sklearn.linear_model import LogisticRegression
```

In [65]:

```
log = LogisticRegression()
log.fit(x_train, y_train)
```

```
C:\Users\USER\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:469: Converg
enceWarning: 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[65]:

```
▾ LogisticRegression
LogisticRegression()
```

In [67]:

```
y_predict=log.predict(x_test)
```

In [69]:

```
from sklearn.metrics import accuracy_score
accuracy_score (y_test,y_predict)
```

Out[69]:

```
0.8032786885245902
```

Confusion Matrix

In [72]:

```
from sklearn.naive_bayes import GaussianNB
```

In [74]:

```
from sklearn.metrics import confusion_matrix, classification_report
```

In [76]:

```
model = GaussianNB()
model.fit(x_train, y_train)
y_predict = model.predict(x_test)
```

In [78]:

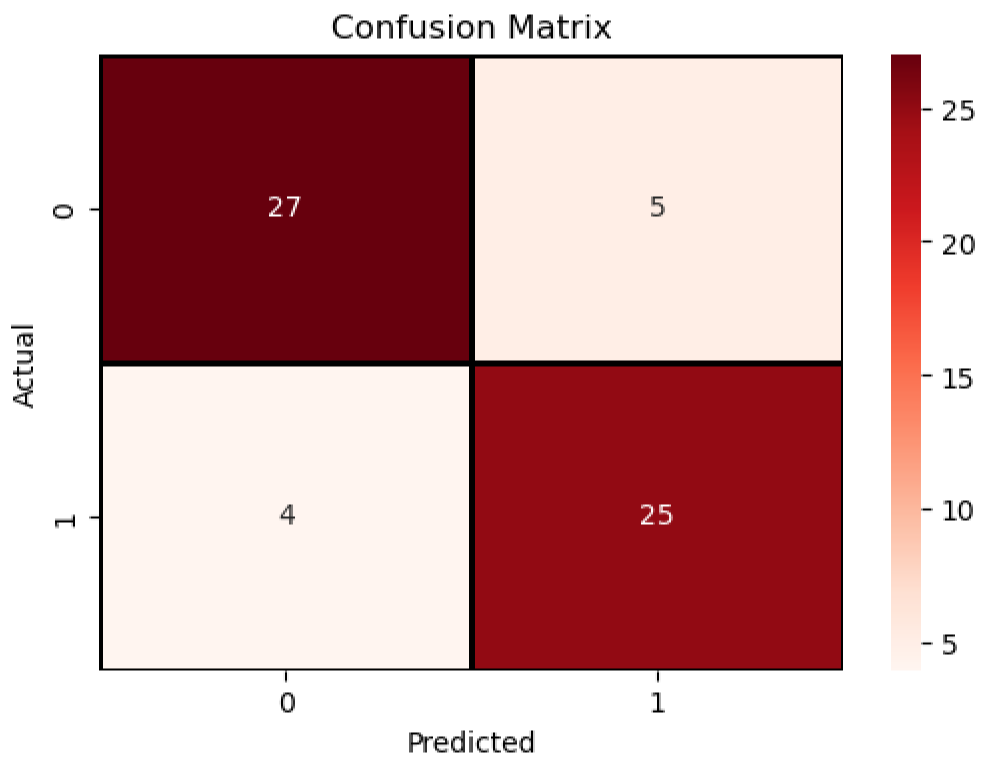
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
```

In [80]:

```
confusionMatrix = confusion_matrix(y_test, y_predict)
```

```
plt.figure(figsize=(6,4))

sns.heatmap(confusionMatrix, annot=True, fmt ='d',cmap="Reds",linewidths=1, linecolor='b
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



In [82]:

```
print("Classification Report: \n")
print(classification_report(y_test,y_predict))
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.87      0.84      0.86        32
           1       0.83      0.86      0.85        29

    accuracy                           0.85        61
   macro avg       0.85      0.85      0.85        61
weighted avg       0.85      0.85      0.85        61
```