In [1]:
```
#Step1:importing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_r
```

```
c:\users\vamsi2001\appdata\local\programs\python\python39\lib\site-packages\num
py\_distributor_init.py:30: UserWarning: loaded more than 1 DLL from .libs:
c:\users\vamsi2001\appdata\local\programs\python\python39\lib\site-packages\num
py\.libs\libopenblas.EL2C6PLE4ZYW3ECEVIV3OXXGRN2NRFM2.gfortran-win_amd64.dll
c:\users\vamsi2001\appdata\local\programs\python\python39\lib\site-packages\num
py\.libs\libopenblas.XWYDX2IKJW2NMTWSFYNGFUWKQU3LYTCZ.gfortran-win_amd64.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
```

In [2]:
```
df = pd.read_csv('diabetes.csv')
df.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

In [3]:
```
df.shape
```

Out[3]:
```
(768, 9)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [4]:
```
df['Outcome']=df['Outcome'].astype('category')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype   
---  ------                    --------------  -----   
 0   Pregnancies               768 non-null    int64   
 1   Glucose                   768 non-null    int64   
 2   BloodPressure             768 non-null    int64   
 3   SkinThickness             768 non-null    int64   
 4   Insulin                   768 non-null    int64   
 5   BMI                       768 non-null    float64 
 6   DiabetesPedigreeFunction  768 non-null    float64 
 7   Age                       768 non-null    int64   
 8   Outcome                   768 non-null    category
dtypes: category(1), float64(2), int64(6)
memory usage: 49.0 KB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [6]:
```
x= df.drop('Outcome', axis=1)
#x = df.iloc[:,:-1]#input
y=df['Outcome']#target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_stat
```

In [7]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
```

In [8]:
```
from sklearn.linear_model import LogisticRegression
cl=LogisticRegression()
```

In [9]:
```
cl.fit(x_train,y_train)
```

Out[9]:
```
LogisticRegression()
```

In [10]:
```
y_pred=cl.predict(x_test)
y_pred
```

Out[10]:
```
array([1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0], dtype=int64)
```

In [13]:
```
#probality
probs = cl.predict_proba(x_test)
probs
```

Out[13]:
```
array([[0.16695202, 0.83304798],
       [0.65685958, 0.34314042],
       [0.38593995, 0.61406005],
       [0.94525754, 0.05474246],
       [0.93027869, 0.06972131],
       [0.38391164, 0.61608836],
       [0.09991261, 0.90008739],
       [0.53596695, 0.46403305],
       [0.41237179, 0.58762821],
       [0.70424728, 0.29575272],
       [0.15612443, 0.84387557],
       [0.82386921, 0.17613079],
       [0.8685419 , 0.1314581 ],
       [0.76649611, 0.23350389],
       [0.17896692, 0.82103308],
       [0.9564985 , 0.0435015 ],
       [0.7567262 , 0.2432738 ],
       [0.92733956, 0.07266044],
       [0.94188829, 0.05811171],
       [0.64808065, 0.35191935],
```

In [12]:
```
print("Accuracy :",accuracy_score(y_pred,y_test))
print("cls_Report :\n",classification_report(y_pred,y_test))
```

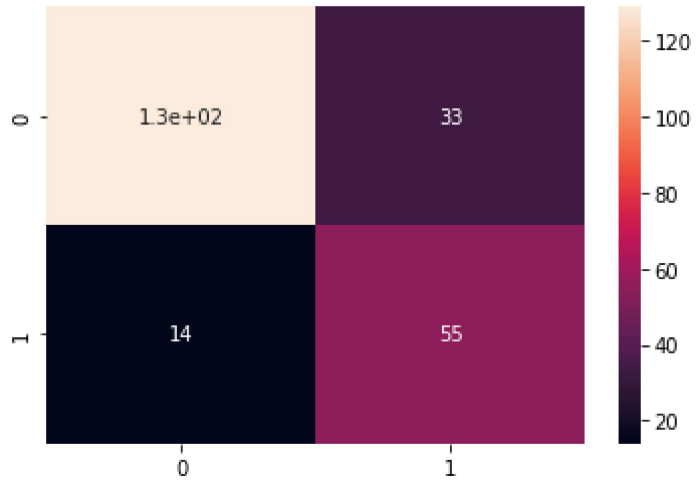
```
Accuracy : 0.7965367965367965
cls_Report :
              precision    recall  f1-score   support

           0       0.90      0.80      0.85       162
           1       0.62      0.80      0.70        69

    accuracy                           0.80       231
   macro avg       0.76      0.80      0.77       231
weighted avg       0.82      0.80      0.80       231
```

In [15]:
```
1 confusion_matrix(y_pred,y_test)
```

Out[15]:
```
array([[129,  33],
       [ 14,  55]], dtype=int64)
```

In [16]:
```
1 sns.heatmap(confusion_matrix(y_pred,y_test),annot=True)
```

Out[16]:
```
<AxesSubplot:>
```



In [17]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    category
dtypes: category(1), float64(2), int64(6)
memory usage: 49.0 KB
```

In [22]:
```
X = [[2, 120, 70, 25, 80, 28.5, 0.5, 33]]  # custom input
X_scaled = scaler.fit_transform(X)
res=cl.predict(X_scaled)[0]
if(res==0):
    print("No :) Diabetes for you...")
else:
    print("Sorry :( you have diabetes")
res
```

```
No :) Diabetes for you...
```

Out[22]:
```
0
```

In [20]:
```
X_scaled
```

Out[20]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0.]])
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [22]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

In [25]:
```
cl.fit(x_train_scaled,y_train)
```

Out[25]:
```
LogisticRegression()
```

In [26]:
```
y_pred_scaled=cl.predict(x_test_scaled)
```

In [28]:
```
print("Accuracy :",accuracy_score(y_pred,y_test))
print("cls_Report :\n",classification_report(y_pred,y_test))
```

```
Accuracy : 0.7965367965367965
cls_Report :
               precision    recall  f1-score   support

           0       0.89      0.80      0.84       158
           1       0.65      0.78      0.71        73

    accuracy                           0.80       231
   macro avg       0.77      0.79      0.78       231
weighted avg       0.81      0.80      0.80       231
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```