In [1]:

```
#reading the dataset
import pandas as pd
dataset=pd.read_excel('diabetes.xlsx')
dataset.head(5)
```

Out[1]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
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 [2]:

```
#divide the dataset into input and output
array=dataset.values
X=array[:,0:8]
Y=array[:,8]
```

In [5]:

```
#divide the dataset into training and testing
from sklearn.model_selection import train_test_split
seed=10
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=seed
```

In [7]:

```
#build the model
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100)
#training the model
rf.fit(X_train,Y_train)
```

Out[7]:

```
RandomForestClassifier()
```

In [8]:

```
#testing the model
pred=rf.predict(X_test)
```

In [9]:

```
#performance Evaluation
from sklearn.metrics import accuracy_score
Acc=accuracy_score(Y_test,pred)
Acc=Acc*100
print(Acc)
```

```
74.02597402597402
```

In [10]:

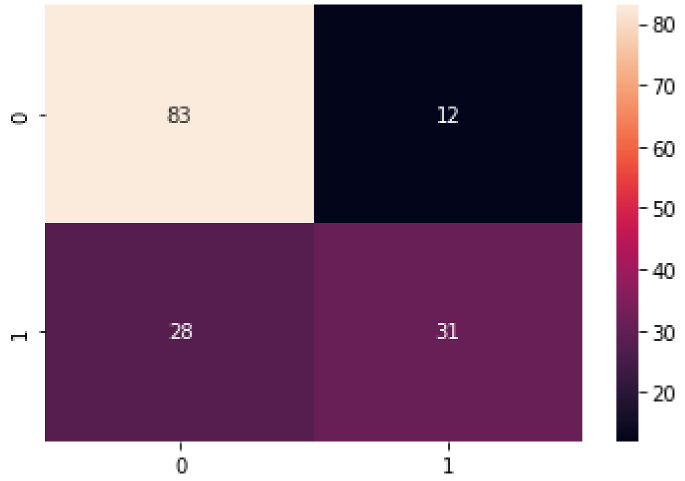
```
from sklearn.metrics import confusion_matrix
CM=confusion_matrix(Y_test,pred)
CM
```

Out[10]:

```
array([[83, 12],
       [28, 31]], dtype=int64)
```

In [12]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(CM,annot=True)
plt.show()
```



In [13]:

```
from sklearn.metrics import classification_report
CR=classification_report(Y_test,pred)
print(CR)
```

```
              precision    recall  f1-score   support

         0.0       0.75      0.87      0.81        95
         1.0       0.72      0.53      0.61        59

    accuracy                           0.74       154
   macro avg       0.73      0.70      0.71       154
weighted avg       0.74      0.74      0.73       154
```

In [ ]:

In [1]:

```
#reading the dataset
import pandas as pd
dataset=pd.read_excel('diabetes.xlsx')
dataset.head(5)
```

Out[1]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [4]:

```
dataset.shape
```

Out[4]:

```
(768, 9)
```

In [6]:

```
dataset.describe()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabet
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [7]:

```
dataset['Outcome'].value_counts()
```

Out[7]:

```
0    500
1    268
Name: Outcome, dtype: int64
```

In [8]:

```
dataset.groupby('Outcome').mean()
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
Outcome							
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	

In [9]:

```
X=dataset.drop(columns='Outcome',axis=1)
Y=dataset['Outcome']
```

In [10]:

```
print(X)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
```

In [12]:

```
print(Y)
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [19]:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X)
```

Out[19]:

```
StandardScaler()
```

In [20]:

```
standardized_data=scaler.transform(X)
```

In [21]:

```
print(standardized_data)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
```

In [22]:

```
X=standardized_data
Y=dataset['Outcome']
```

In [58]:

```
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,stratify=Y,random_state=2)
```

In [59]:

```
print(X.shape,X_train.shape,X_test.shape)
```

```
(768, 8) (614, 8) (154, 8)
```

In [60]:

```
#training the model
from sklearn import svm
classifier=svm.SVC(kernel='linear')
```

In [61]:

```
#training the support vector
classifier.fit(X_train, Y_train)
```

Out[61]:

```
SVC(kernel='linear')
```

In [62]:

```
#evaluating module
#accuracy score
from sklearn.metrics import accuracy_score
X_train_prediction=classifier.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
print('Accuracy of training data:',training_data_accuracy)
```

```
Accuracy of training data: 0.7866449511400652
```

In [63]:

```
X_test_prediction=classifier.predict(X_test)
testing_data_accuracy=accuracy_score(X_test_prediction,Y_test)
print('Accuracy of testing data:',testing_data_accuracy)
```

```
Accuracy of testing data: 0.7727272727272727
```

In [39]:

```
#making a predictive system
import numpy as np
input_data=(8,183,64,0,0,23.3,0.672,32)
input_array=np.asarray(input_data)
#reshaping the data as we are predicting for one instance
input_reshaped=input_array.reshape(1,-1)
#standardize the input data
std_data=scaler.transform(input_reshaped)
print(std_data)
prediction=classifier.predict(std_data)
print(prediction)
if(prediction[0]==0):
    print('the person is not diabetic')
else:
    print('the person is diabetic')
```

```
[[ 1.23388019  1.94372388 -0.26394125 -1.28821221 -0.69289057 -1.10325546
   0.60439732 -0.10558415]]
[1]
the person is diabetic

C:\Users\keert\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning:
X does not have valid feature names, but StandardScaler was fitted with feat
ure names
  warnings.warn(
```