Importing the dependencies

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import adjusted_rand_score
```

Data Collection and Analysis Diabetes dataset

In [2]:

```
#load the Dataset to a pandas dataframe
diabetes_data = pd.read_csv(r"C:\Users\user\Desktop\Machine learning
```

In [3]:

```
# printing the first 10 rowa of the data
diabetes_data.head(10)
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
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	
5	5	116	74	0	0	25.6	
6	3	78	50	32	88	31.0	
7	10	115	0	0	0	35.3	
8	2	197	70	45	543	30.5	
9	8	125	96	0	0	0.0	

In [4]:

```
# lets look at the number of rows and colums in the dataset
diabetes_data.shape
```

Out[4]:

```
(768, 9)
```

In [5]:

```python
# statistical measures of the data
diabetes_data.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Di
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 [6]:

```python
# checking counts of output
diabetes_data['Outcome'].value_counts()
```

Out[6]:

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

0 --> Non-diabetic

1 --> diabetic

In [7]:

```python
# mean value for both ccaes
diabetes_data.groupby('Outcome').mean()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	I
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 [8]:

```python
# separating data and labels
x = diabetes_data.drop(columns= 'Outcome', axis=1)
y = diabetes_data['Outcome']
```

In [9]:

```
print(x)
print(y)
```

```
     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]
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
```

Data Standardization

In [10]:

```
# Performing standardization
scaler = StandardScaler()
```

In [11]:

```
scaler.fit(x)
```

Out[11]:

```
StandardScaler()
```

In [12]:

```
Sd_data = scaler.transform(x)
```

In [13]:

```
print(Sd_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 [14]:

```
x = Sd_data
y = diabetes_data['Outcome']
```

In [15]:

```
print(x)
print(y)
```

```
[[ 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]]
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
```

Train Test Split

Spliting dataset into train and test set

In [16]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0
```

In [17]:

```
# let's see the shapes
print(x.shape,x_train.shape, x_test.shape)
```

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

Training a model

In [18]:

```
classifier = svm.SVC(kernel='linear')
```

In [19]:

```
#training the suport vector Machine Classifier
classifier.fit(x_train, y_train)
```

Out[19]:

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

Model Evaluation

Lets check classification report,confusion marix and accuracy score of above model

In [20]:

```
#importing tools
from sklearn.metrics import classification_report, accuracy_score, co
```

In [21]:

```
# prediction classes
y_pred = classifier.predict(x_test)
```

In [22]:

```
print("CLASSIFICATION REPORT: ")
print(classification_report(y_test, y_pred))
```

```
CLASSIFICATION REPORT: 
              precision    recall  f1-score   support

           0       0.78      0.91      0.84       100
           1       0.76      0.52      0.62        54

    accuracy                           0.77       154
   macro avg       0.77      0.71      0.73       154
weighted avg       0.77      0.77      0.76       154
```

In [23]:

```
#Accuracy score of training data
x_train_prediction = classifier.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction,y_train )*
```

In [24]:

```
print("ACCURACY OF TRAINING DATA: ",training_data_accuracy.round(2))
```

```
ACCURACY OF TRAINING DATA:  78.66
```

In [25]:

```
#Accuracy score of test data
x_test_prediction = classifier.predict(x_test)
test_data_accuracy = accuracy_score(x_test_prediction,y_test )*100
```

In [26]:

```
print("ACCURACY OF Test DATA: ",test_data_accuracy.round(2))
```

```
ACCURACY OF Test DATA:  77.27
```

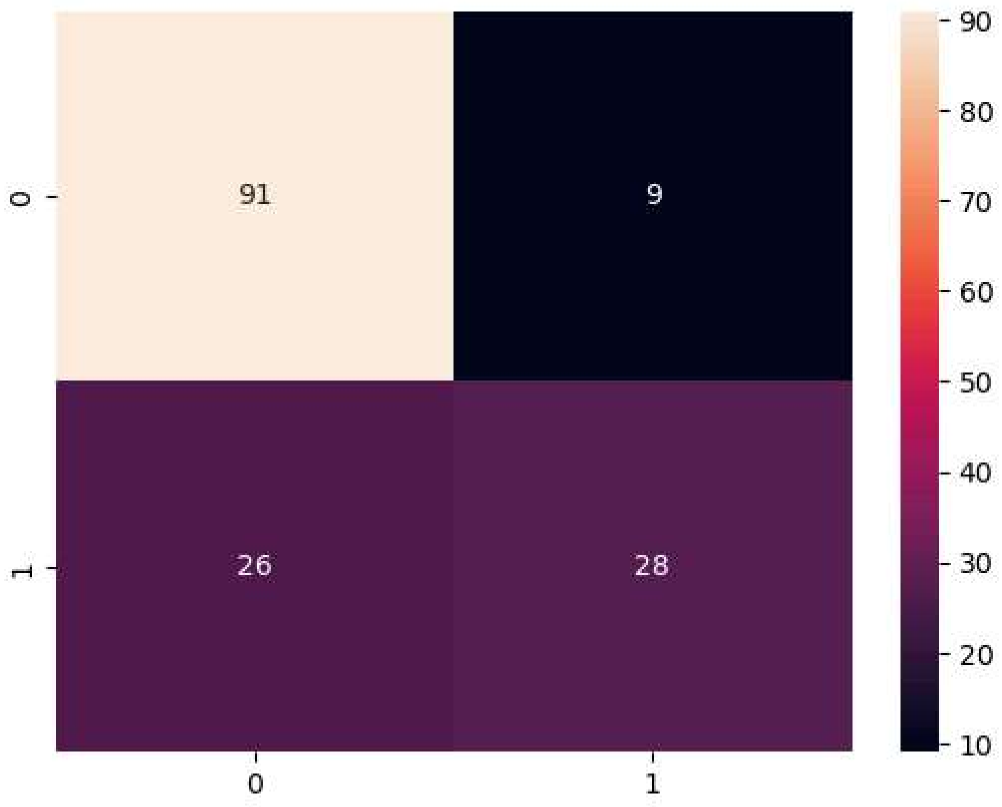
Confusion Matrix

In [27]:

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
#Visualizing the Confusion Matrix
import seaborn as sns
sns.heatmap(cm,annot=True)
```

Out[27]:

```
<AxesSubplot:>
```



Making a predictive System

In [28]:

```python
# input_data = (4,110,92,0,0,37.6,0.191,30)  not diabetic
input_data = (5,166,72,19,175,25.8,0.587,51)  # diabetic

# input data into numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the inputdata
std_data = scaler.transform(input_data_reshaped)
# print(std_data)

prediction = classifier.predict(std_data) # imp line
# print(prediction)

if (prediction[0]==0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

```
The person is diabetic
```

Saving the trained model

In [29]:

```python
import pickle
```

In [30]:

```python
filename = "trained_model.sav"
pickle.dump(classifier, open(filename, 'wb'))
```

In [31]:

```python
#loading the save model
loaded_model = pickle.load(open('trained_model.sav', 'rb'))
```

In [32]:

```python
# input_data = (4,110,92,0,0,37.6,0.191,30)  not diabetic
input_data = (5,166,72,19,175,25.8,0.587,51)  # diabetic

# input data into numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the inputdata
std_data = scaler.transform(input_data_reshaped)
# print(std_data)

prediction = loaded_model.predict(std_data) # imp line
# print(prediction)

if (prediction[0]==0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

```
The person is diabetic
```

In [ ]:

```

```