Importing 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
```

Data Collection and Analysis

In [5]:
```
diabetes_data = pd.read_csv('diabetes.csv')
```

In [7]:
```
#prinrting the first five row of our data
```

In [6]:
```
diabetes_data.head()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	O
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	

In [8]:
```
#number of rows and columns in this data set
diabetes_data.shape
```

Out[8]:
```
(768, 9)
```

In [9]:
```
#getting the stststical measures if the data
diabetes_data.describe()
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeI
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	76
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	

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeI
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [10]:
```
diabetes_data['Outcome'].value_counts()
```

Out[10]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

0--> Non Diabetic

1--> Diabetic

In [12]:
```
diabetes_data.groupby('Outcome').mean()
```

Out[12]:

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
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 [14]:
```
#seperaring Data and Lables
X = diabetes_data.drop(columns = 'Outcome', axis = 1)
Y = diabetes_data['Outcome']
```

In [15]:
```
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 [16]:
```
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
```

Data Standardization

In [17]:
```
scaler = StandardScaler()
```

In [18]:
```
scaler.fit(X)
```

Out[18]:
```
StandardScaler()
```

In [19]:
```
standardized_data = scaler.transform(X)
```

In [20]:
```
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 [21]:
```
X = standardized_data
Y = diabetes_data['Outcome']
```

In [22]:
```
print(X , 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

In [23]:
```
X_train , X_test ,Y_train ,Y_test = train_test_split(X ,Y ,test_size = 0.2 ,stratify =
```

In [24]:
```
print(X.shape , X_train.shape , X_test.shape)
```

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

Trianing the model

In [27]:
```
clasifier = svm.SVC(kernel = 'linear')
```

In [28]:
```
#training the support vector machine Classifier
clasifier.fit(X_train,Y_train)
```

Out[28]:
```
SVC(kernel='linear')
```

Model Evaluating

Accuray score

In [31]:
```
#accuracy score on training data
X_train_prediction = clasifier.predict(X_train)
training_data_accurcy = accuracy_score(X_train_prediction , Y_train)
```

In [33]:
```
print('Accuracy Score of the training data',training_data_accurcy)
```

```
Accuracy Score of the training data 0.7866449511400652
```

In [34]:
```
X_test_prediction = clasifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction , Y_test)
```

In [35]:
```
print('Accuracy score for Test data ' , test_data_accuracy)
```

```
Accuracy score for Test data  0.7727272727272727
```

Making a prediticive System

In [50]:
```
input_data = (1,85,66,29,0,26.6,0.351,31)
#chaniging input dat tor numpy array
input_data_as_numpy = np.asarray(input_data)
#reshape the array as we are prediting for one instance
input_data_reshaped = input_data_as_numpy.reshape(1,-1)
```

In [51]:
```
#standardizing the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)
predition = clasifier.predict(std_data)
print(predition)
```

```
[[-0.84488505 -1.12339636 -0.16054575  0.53090156 -0.69289057 -0.68442195
  -0.36506078 -0.19067191]]
[0]
```

In [52]:
```
if (predition[0] == 0):
    print('The person is not diabetic')
else:
    print('The person is diabetic')
```

```
The person is not diabetic
```