importing the dependencies

In [4]:
```
import pandas as pd
import numpy as np
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 Data Processing

PIMA Diabetes Dataset

In [18]:
```
#loading the dataset to pandas Dataframe
diabetes_data = pd.read_csv('diabetes.csv')
```

In [14]:
```
pd.read_csv?
```

In [19]:
```
diabetes_data
```

Out[19]:

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

768 rows × 9 columns

In [21]:
```
#printing the frist 5 rows
diabetes_data.head()
```

Out[21]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctic
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 [20]:
```
#number of rows and columns in this dataset
diabetes_data.shape
```

Out[20]: (768, 9)

In [22]:
```
#getting statistcal measures of the dataset
diabetes_data.describe()
```

Out[22]:

	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 [23]:
```
diabetes_data['Outcome'].value_counts()
```

Out[23]:
```
Outcome
0    500
1    268
Name: count, dtype: int64
```

```
0 --> Non-Diabetic
1 --> Diabetic
```

In [25]:
```
diabetes_data.groupby('Outcome').mean()
```

Out[25]:

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
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 [27]:
```
#separeting data andlabels
x = diabetes_data.drop(columns='Outcome', axis=1)
y = diabetes_data['Outcome']
```

In [28]:
```
print(x)
print(y)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BM
I  \
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 Standardizition

In [29]:
```
scaler = StandardScaler()
```

In [30]:
```
scaler.fit(x)
```

Out[30]:
```
▼ StandardScaler
StandardScaler()
```

In [31]:
```
standardized_data = scaler.transform(x)
```

In [32]:
```
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 [33]:
```
x= standardized_data
y = diabetes_data['Outcome']
```

In [34]:
```
print(x)
```

```
[[ 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 [35]:
```
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
```

Training and Test split

In [42]:
```
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size = 0.2,st
```

In [43]:
```
print(x.shape,x_train.shape,x_test.shape)
```

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

In [44]:
```
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
```

Model training --> SVM Model

In [46]:
```
classifier = svm.SVC(kernel='linear')
```

In [48]:
```
#traning the support vector Machine Classifier
classifier.fit(x_train, y_train)
```

Out[48]:
```
SVC(kernel='linear')
```

Model Evaluation

In [50]:
```
# accuracy on training data
x_train_prediction = classifier.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction,y_train)
```

In [51]:
```
print("Accuracy on training data : " ,training_data_accuracy)
```

```
Accuracy on training data :  0.7866449511400652
```

In [52]:
```
# accuracy on test data
x_test_prediction = classifier.predict(x_test)
test_data_accuracy = accuracy_score(x_test_prediction,y_test)
```

In [53]:
```
print("Accuracy on test data : " ,test_data_accuracy)
```

```
Accuracy on test data :  0.7727272727272727
```

Making a Predictive System

In [84]:
```
input_data = (5,121,72,23,112,26.2,0.245,30)

# changing the input_data to numpy arrary
input_data_as_numpy_array  = np.asarray(input_data)

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

#to print the standardiziation data
print(std_data)

#to predict the input data
prediction  = classifier.predict(std_data)

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

```
[[ 1.53084665 -0.99820778 -0.36733675 -1.28821221 -0.69289057 -1.2047
9085
  -0.99626558 -0.0204964 ]]
The person is not disabetic
```