DIABETES PREDICTION

Objective:

To chcek whether a patient is suffering from diabetes or not.

Importing the dependencies

In [6]:
```
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
```

The Dataset

In [7]:
```
diabetes_dataset=pd.read_csv(r'C:\Users\HP\Downloads\diabetes.csv')
```

In [8]:
```
diabetes_dataset.head()
```

Out[8]:

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

In [9]:
```
diabetes_dataset.tail()
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

In [11]:
```
diabetes_dataset.shape
```

Out[11]:
```
(768, 9)
```

The given dataset has 768 rows and 9 columns.

In [13]:
```
diabetes_dataset.isnull().sum()
```

Out[13]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

This dataset has no null values.

Statistical insights of the dataset

In [17]:
```
diabetes_dataset.describe()
```

Out[17]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	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	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [18]:
```
diabetes_dataset['Outcome'].value_counts()
```

Out[18]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

0 --> Non-diabetic patients 1 --> Diabetic patients

Hence we can conclude that according to the given dataset 500 patients are non-diabetic patients where as 268 patients are diabetic patients.

In [20]:
```
diabetes_dataset.groupby('Outcome').mean()
```

Out[20]:

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

So observing the mean value it can be said that glucose level,blood pressure,insulin and BMI are higher in case of diabetic patients than those of non-diabetic patients.

Seperating the data and labels

In [22]:
```
X=diabetes_dataset.drop(columns=['Outcome'],axis=1)
Y=diabetes_dataset['Outcome']
```

In [23]:
```
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 [24]:
```
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 [25]:
```
scaler=StandardScaler()
```

In [26]:
```
scaler.fit(X)
```

Out[26]:
```
StandardScaler()
```

In [27]:
```
standardised_data=scaler.transform(X)
```

In [28]:
```
print(standardised_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 [29]:
```
X=standardised_data
Y=diabetes_dataset['Outcome']
```

In [30]:
```
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 [31]:
```
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
```

Train Test Split

In [32]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
```

In [33]:
```
print(X.shape,X_train.shape,X_test.shape)
```
```
(768, 8) (614, 8) (154, 8)
```

Training the Model

In [34]:
```
classifier=svm.SVC(kernel='linear')
```

Training the Support Vector Machine Classifier

In [35]:
```
classifier.fit(X_train,Y_train)
```

Out[35]:
```
SVC(kernel='linear')
```

Model Evaluation

Accuracy Score

Accuracy Score on the Training Data

In [36]:
```
X_train_prediction = classifier.predict(X_train)
```

In [37]:
```
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
```

In [40]:
```
print('Accuracy score of the training data:',training_data_accuracy)
```
```
Accuracy score of the training data: 0.7866449511400652
```

Accuracy score on the test data

In [41]:
```
X_test_prediction = classifier.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
print('Accuracy score of the test data:',test_data_accuracy)
```
```
Accuracy score of the test data: 0.7727272727272727
```

From this it can be concluded that the model is not over-trained.

Making a predictive system

In [42]:
```
input_data=(10,168,74,0,0,38,0.537,34)

#changing the input data to numpy array
input_data_as_numpy_array=np.asarray(input_data)

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

#standardise the input data
std_data=scaler.transform(input_data_reshaped)

print(std_data)

prediction = classifier.predict(std_data)
print(prediction)
```
```
[[ 1.82781311  1.4742667   0.25303625 -1.28821221 -0.69289057  0.76245745
   0.1966813   0.06459135]]
[1]
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

In [45]:
```
if (prediction[0] == 0):
    print('The patient is not diabetic.')
else:
    print('The patient is diabetic.')
```
```
The patient is diabetic.
```

In [ ]:
```

```