Predict whether a patient is diabetic or not

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, f1_score, accuracy_score
```

In [2]:

```
filename = 'diabetes'
path = 'E:/desktop/ML/KNN Algorithm/{}.csv'.format(filename)
data = pd.read_csv(path)
data.head()
```

Out[2]:

	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 [3]:

```
main_columns = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin','BMI']
```

In [4]:

```
for i in main_columns:
    data[i] = data[i].replace(0,np.NaN)
    mean = int(data[i].mean(skipna=True))
    data[i] = data[i].replace(np.NaN,mean)
```

In [5]:

```
data.head()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	155.0	33.6	0.627	50	1
1	1	85.0	66.0	29.0	155.0	26.6	0.351	31	0
2	8	183.0	64.0	29.0	155.0	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

Spilt the Data

In [6]:

```
X = data.iloc[:,0:8]
Y = data.iloc[:,8]
```

In [7]:

```
X.head()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148.0	72.0	35.0	155.0	33.6	0.627	50
1	1	85.0	66.0	29.0	155.0	26.6	0.351	31
2	8	183.0	64.0	29.0	155.0	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33

In [8]:

```
Y.head()
```

Out[8]:

```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

In [9]:

```
X_train,X_test,Y_train,Y_test  = train_test_split(X,Y,random_state=0,test_size=0.2)
```

In [10]:

```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```

In [11]:

```
import math
n = math.sqrt(len(Y_test))
n
```

Out[11]:

```
12.409673645990857
```

Define the model : Init the KNN where n_neighbors=n-1

In [12]:

```
classifier = KNeighborsClassifier(n_neighbors = 11,p=2,metric='euclidean')
classifier.fit(X_train,Y_train)
```

Out[12]:

```
KNeighborsClassifier(metric='euclidean', n_neighbors=11)
```

```
In [13]: Y_pred  = classifier.predict(X_test)
         Y_pred
```

```
Out[13]: array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
                1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1,
                1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
                0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
               dtype=int64)
```

```
In [14]: cm = confusion_matrix(Y_test,Y_pred)
         cm
```

```
Out[14]: array([[95, 12],
                [18, 29]], dtype=int64)
```

```
In [15]: f1_score(Y_test,Y_pred)
```

```
Out[15]: 0.6590909090909092
```

```
In [16]: accuracy_score(Y_test,Y_pred)
```

```
Out[16]: 0.8051948051948052
```

Accuracy of the fitted model is 80 %