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
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
```

In [5]:
```
dataset = pd.read_csv('diabetes.csv')
print(len(dataset))
print(dataset.head())
```

```
768
   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

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

Replace Zeros with Mean

In [7]:
```
zero_not_acc = ['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
```

In [12]:
```
for column in zero_not_acc:
    dataset[column] = dataset[column].replace(0,np.NaN)
    mean=int(dataset[column].mean(skipna=True))# skipna will ignore the Nan value and return the mean
    dataset[column] = dataset[column].replace(np.NaN,mean)
```

Split Dataset in train and test dataset

In [14]:
```
dataset
```

Out[14]:

	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
...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.0	32.9	0.171	63	0
764	2	122.0	70.0	27.0	155.0	36.8	0.340	27	0
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0
766	1	126.0	60.0	29.0	155.0	30.1	0.349	47	1
767	1	93.0	70.0	31.0	155.0	30.4	0.315	23	0

768 rows × 9 columns

In [15]:
```
X = dataset.iloc[:,0:8]# this is our whole data except output
y = dataset.iloc[:,8]# this is our output
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0,test_size=0.2)
#test size 20% we make it a  side so that we can train it later
```

Setting Standard Scale

In [16]:
```
Sc_X = StandardScaler()
X_train = Sc_X.fit_transform(X_train)
X_test = Sc_X.fit_transform(X_test)
```

Defining Model

In [18]:
```
#n_neighbors = odd (number of neighbours around test data point)
# p= 2 (We need to find is that a diabatic or not)
#metric = euclidean (it is method of finding distance between points)
classifier = KNeighborsClassifier(n_neighbors=11 , p = 2 , metric='euclidean')
```

Evaluate the model through confusion metrix

In [22]:
```
classifier.fit(X_train,y_train)
```

Out[22]:
```
KNeighborsClassifier(metric='euclidean', n_neighbors=11)
```

In [23]:
```
y_pred = classifier.predict(X_test)
y_pred
```

```
C:\Users\mosai\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Out[23]:
```
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 [24]:
```
cm = confusion_matrix(y_test,y_pred)
print(cm)
print(f1_score(y_test,y_pred))
```

```
[[95 12]
 [18 29]]
0.6590909090909092
```

In [25]:
```
print(accuracy_score(y_test,y_pred))
```

```
0.8051948051948052
```

In [ ]:
```

```