DIABETICS PREDICTION

In [18]:
```
import numpy as np
import pandas as pd
```

IMPORTING DATASET

In [19]:
```
dataset=pd.read_csv("Diabetics.csv")
dataset.head(5)
```

Out[19]:

	Pregnancies	Glucose	blood pressure	skin thickness	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

ANALYZING DATA

In [20]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   blood pressure            768 non-null    int64  
 3   skin thickness            768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [21]:
```
dataset.isnull().sum()
```

Out[21]:
```
Pregnancies                 0
Glucose                     0
blood pressure              0
skin thickness              0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [22]:
```
x=dataset.iloc[:,:-1]
y=dataset.iloc[:,-1]
```

In [23]:
```
print(x)
```

```
     Pregnancies  Glucose  blood pressure  skin thickness  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
```

SPLITTING DATA

In [26]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=25,random_state=0)
```

APPLYING CLASSIFIERS AND EVALUATION

RANDOM FOREST

In [32]:
```
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators=8,criterion='entropy',random_state=0)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
```

In [37]:
```
from sklearn.metrics import accuracy_score
acc_logreg2 = round(accuracy_score(y_pred,y_test),2)*100
print("Accuracy:",acc_logreg2)
```

```
Accuracy: 88.0
```

LOGISTIC REGRESSION

In [39]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, r2_score, classification_report
logreg=LogisticRegression(solver='lbfgs',max_iter=1000)
logreg.fit(x_train,y_train)
y_pred=logreg.predict(x_test)
acc_logreg1=round(accuracy_score(y_pred,y_test),2)*100
print("Accuracy:",acc_logreg1)
```

```
Accuracy: 96.0
```

K NEIGHBOR CLASSIFIER

In [41]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
acc_knn=round(accuracy_score(y_pred,y_test),2)*100
print("Accuracy:",acc_knn)
```

```
Accuracy: 88.0
```