In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df=pd.read_csv("diabetes.csv")
```

In [3]:
```
df.head()
```

Out[3]:

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

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [6]:
```
df.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   BloodPressure             768 non-null    int64  
 3   SkinThickness             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 [5]:
```
df.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPe
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 [7]:
```
for i in ["Glucose","BloodPressure","SkinThickness","Insulin","BMI"]:
    df[i].replace(0,np.nan,inplace=True)
    df[i].fillna(df[i].mean(),inplace=True)
```

In [8]:
```
df.describe()
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPe
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	121.686763	72.405184	29.153420	155.548223	32.457464	
std	3.369578	30.435949	12.096346	8.790942	85.021108	6.875151	
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	
25%	1.000000	99.750000	64.000000	25.000000	121.500000	27.500000	
50%	3.000000	117.000000	72.202592	29.153420	155.548223	32.400000	
75%	6.000000	140.250000	80.000000	32.000000	155.548223	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [9]:
```
df["Outcome"].value_counts()
```

Out[9]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [10]:
```
#seperating X and Y
```

In [11]:
```
df.head()
```

Out[11]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148.0	72.0	35.00000	155.548223	33.6	0.62
1	1	85.0	66.0	29.00000	155.548223	26.6	0.35
2	8	183.0	64.0	29.15342	155.548223	23.3	0.67
3	1	89.0	66.0	23.00000	94.000000	28.1	0.16
4	0	137.0	40.0	35.00000	168.000000	43.1	2.28

In [12]:
```
x=df.iloc[:,1:-1]
x
```

. . .

In [13]:
```
y=df["Outcome"]
y
```

. . .

In [14]:
```
#split a data into training and testing
```

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=1)
```

In [17]:
```
#build model
```

In [18]:
```
from sklearn.linear_model import LogisticRegression
```

In [19]:
```
logreg=LogisticRegression()
logreg.fit(xtrain,ytrain)
ypred=logreg.predict(xtest)
```

In [20]:
```
#evaluate a mdodel
```

In [22]:
```
from sklearn.metrics import classification_report
```

In [23]:
```
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.87      0.84       123
           1       0.73      0.64      0.68        69

    accuracy                           0.79       192
   macro avg       0.77      0.75      0.76       192
weighted avg       0.78      0.79      0.78       192
```

In [24]:
```
#hypertuning using solver parameter
```

In [26]:
```
logreg=LogisticRegression(solver="liblinear")
logreg.fit(xtrain,ytrain)
ypred=logreg.predict(xtest)
```

In [27]:
```
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       123
           1       0.75      0.58      0.66        69

    accuracy                           0.78       192
   macro avg       0.77      0.74      0.75       192
weighted avg       0.78      0.78      0.77       192
```

In [28]:
```
logreg=LogisticRegression(solver="saga")
logreg.fit(xtrain,ytrain)
ypred=logreg.predict(xtest)
```

In [29]:
```
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.72      0.90      0.80       123
           1       0.68      0.38      0.49        69

    accuracy                           0.71       192
   macro avg       0.70      0.64      0.64       192
weighted avg       0.71      0.71      0.69       192
```

In [30]:
```
#feature scaling
```

In [31]:
```
from sklearn.preprocessing import StandardScaler
```

In [32]:
```
sc=StandardScaler()
xtrain=sc.fit_transform(xtrain)
```

In [34]:
```
xtest=sc.transform(xtest)
```

In [37]:
```
logreg=LogisticRegression(solver="saga")
logreg.fit(xtrain,ytrain)
ypred=logreg.predict(xtest)
```

In [38]:
```
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.87      0.84       123
           1       0.73      0.64      0.68        69

    accuracy                           0.79       192
   macro avg       0.77      0.75      0.76       192
weighted avg       0.78      0.79      0.78       192
```

In [ ]:
```

```