In [1]:
```
import pandas as pd
df=pd.read_csv("diabetes.csv",header=None,names=["Pregnancies","Glucose","
```

In [2]:
```
df.sample(2)
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes	Age	Cl
734	2	105	75	0	0	23.3	0.560	53	
590	11	111	84	40	0	46.8	0.925	45	

In [3]:
```
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   Diabetes       768 non-null    float64
 7   Age            768 non-null    int64  
 8   Class          768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [4]:
```
x=df.iloc[:,:-1]
x
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63
764	2	122	70	27	0	36.8	0.340	27
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

768 rows × 8 columns

In [16]:
```
from sklearn.preprocessing import StandardScaler
```

In [17]:
```
sc=StandardScaler()
xscaled=sc.fit_transform(x)
```

In [18]:
```
x.shape
```

Out[18]:
```
(768, 8)
```

In [19]:
```
y=df.iloc[:,-1]
y
```

Out[19]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Class, Length: 768, dtype: int64
```

In [20]:
```
y.shape
```
Out[20]:
```
(768,)
```

In [21]:
```
from sklearn.model_selection import train_test_split
```

In [23]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,random_state=1,test_size=0.
```

In [24]:
```
xstrain,xstest,ystrain,ystest=train_test_split(x,y,random_state=1,test_siz
```

In [25]:
```
xtrain.shape
```
Out[25]:
```
(576, 8)
```

In [26]:
```
xtest.shape
```
Out[26]:
```
(192, 8)
```

In [27]:
```
ytrain.shape
```
Out[27]:
```
(576,)
```

In [28]:
```
ytest.shape
```
Out[28]:
```
(192,)
```

In [29]:
```
from sklearn.linear_model import LogisticRegression
```

In [30]:
```
model=LogisticRegression()
```

In [31]:
```
model2=LogisticRegression()
```

In [32]:
```
model.fit(xtrain,ytrain)
```

```
C:\Users\SAM\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.p
y:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://s
cikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression (https://scikit-learn.org/stable/modules/linear_model.html#logis
tic-regression)
  n_iter_i = _check_optimize_result(
```

Out[32]:
```
▾ LogisticRegression
LogisticRegression()
```

In [33]:
```
model2.fit(xtrain,ytrain)
```

```
C:\Users\SAM\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.p
y:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://s
cikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression (https://scikit-learn.org/stable/modules/linear_model.html#logis
tic-regression)
  n_iter_i = _check_optimize_result(
```

Out[33]:
```
▾ LogisticRegression
LogisticRegression()
```

In [35]:
```
df["Class"].value_counts()
```

Out[35]:
```
Class
0    500
1    268
Name: count, dtype: int64
```

In [36]:
```
predictions=model.predict(xtest)
```

In [37]:
```
predictions2=model2.predict(xtest)
```

In [38]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification
```

In [39]:
```
accuracy_score(ytest,predictions)
```

Out[39]:
```
0.7760416666666666
```

In [40]:
```
accuracy_score(ystest,predictions2)
```

Out[40]:
```
0.7760416666666666
```

In [41]:
```
confusion_matrix(ytest,predictions)
```

Out[41]:
```
array([[109,  14],
       [ 29,  40]], dtype=int64)
```

In [42]:
```
confusion_matrix(ystest,predictions2)
```

Out[42]:
```
array([[109,  14],
       [ 29,  40]], dtype=int64)
```

In [44]:
```
ytest.value_counts()
```

Out[44]:
```
Class
0    123
1     69
Name: count, dtype: int64
```

In [45]:
```
print(classification_report(ytest,predictions))
```

```
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       123
           1       0.74      0.58      0.65        69

    accuracy                           0.78       192
   macro avg       0.77      0.73      0.74       192
weighted avg       0.77      0.78      0.77       192
```

In [46]:
```
print(classification_report(ystest,predictions2))
```

```
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       123
           1       0.74      0.58      0.65        69

    accuracy                           0.78       192
   macro avg       0.77      0.73      0.74       192
weighted avg       0.77      0.78      0.77       192
```

In [ ]:
```

```