In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df=pd.read_csv("diabetes.csv")
df
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [3]:

```
df.shape
```

Out[3]:

```
(768, 9)
```

In [4]:

```
df.describe()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree
count	768.000000	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	0.471876
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000

In [5]:

```
df.head()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	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 [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   Pedigree       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
```

Spliting of data

In [7]:

```
x=df.drop("Outcome", axis= 1)
x
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	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 [8]:

```
y=df["Outcome"]
y
```

Out[8]:

```
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
```

Scaling the data

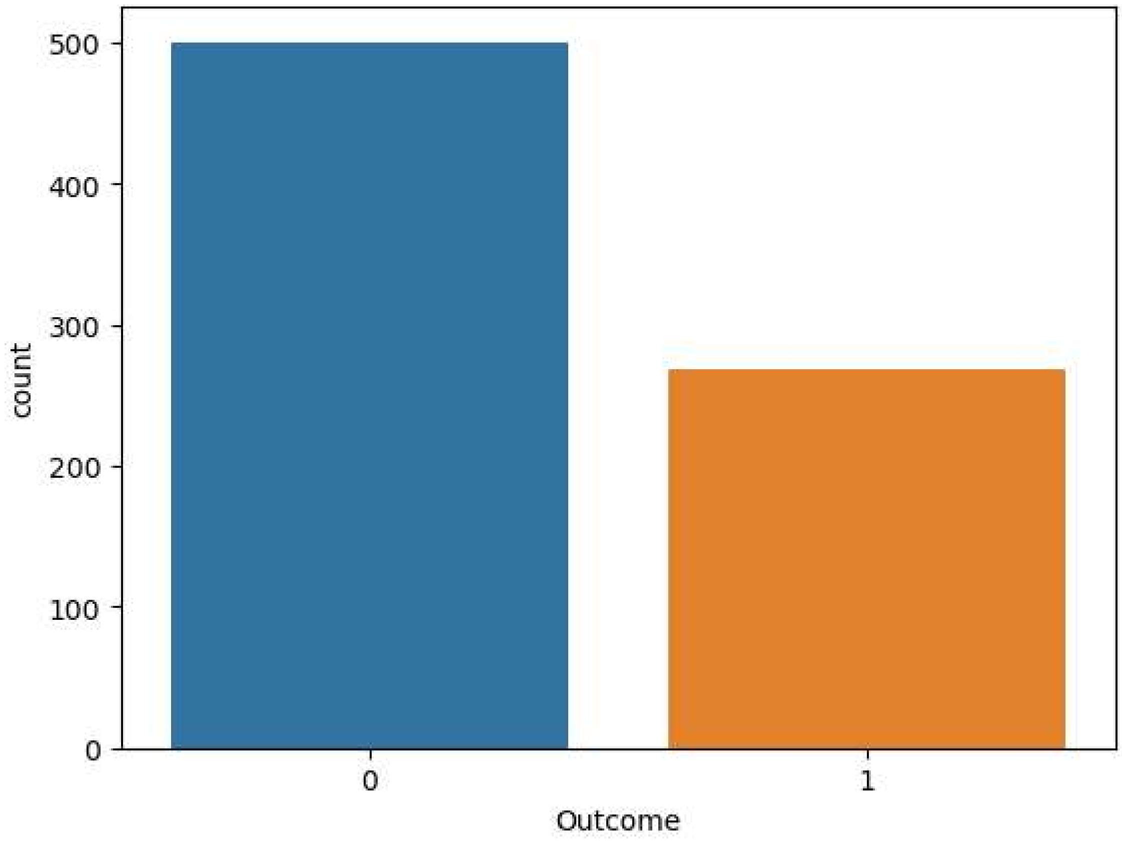
```
In [9]: from sklearn.preprocessing import MinMaxScaler
        scaler= MinMaxScaler()
        x_scale= scaler.fit_transform(x)
        x_scale
```

```
Out[9]: array([[0.35294118, 0.74371859, 0.59016393, ..., 0.50074516, 0.23441503,
                0.48333333],
               [0.05882353, 0.42713568, 0.54098361, ..., 0.39642325, 0.11656704,
                0.16666667],
               [0.47058824, 0.91959799, 0.52459016, ..., 0.34724292, 0.25362938,
                0.18333333],
               ...,
               [0.29411765, 0.6080402 , 0.59016393, ..., 0.390462  , 0.07130658,
                0.15      ],
               [0.05882353, 0.63316583, 0.49180328, ..., 0.4485842 , 0.11571307,
                0.43333333],
               [0.05882353, 0.46733668, 0.57377049, ..., 0.45305514, 0.10119556,
                0.03333333]])
```

Visualizing the data

```
In [10]: sns.countplot(x=y)
```

```
Out[10]: <Axes: xlabel='Outcome', ylabel='count'>
```



Cross-Validation

In [11]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test= train_test_split(x_scale,y, test_size=0.34, randon
```

In [12]:

```
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier()
```

In [13]:

```
knn.fit(x_train,y_train)
```

Out[13]:

```
KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [14]:

```
y_pred= knn.predict(x_test)
```

In [15]:

```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score, classificatic
```

In [16]:

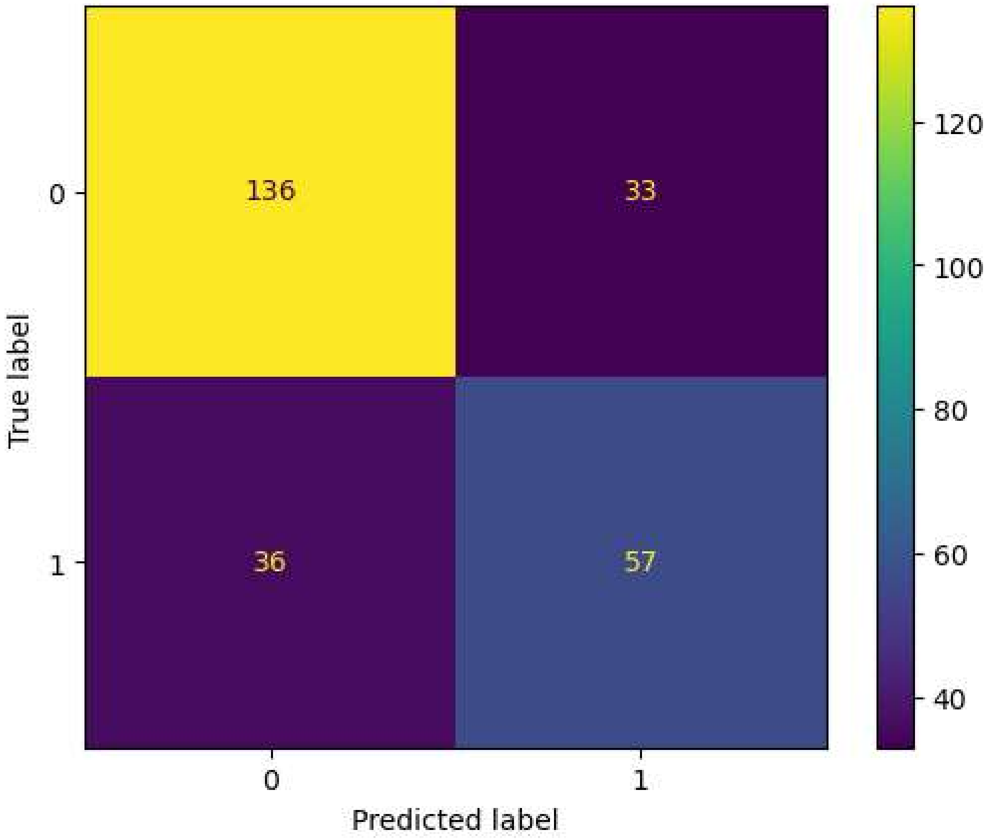
```
accuracy_score(y_test, y_pred)
```

Out[16]:

```
0.7366412213740458
```

In [17]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

Out[17]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x283525ebe10
>
```



In [18]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.80      0.80       169
           1       0.63      0.61      0.62        93

    accuracy                           0.74       262
   macro avg       0.71      0.71      0.71       262
weighted avg       0.73      0.74      0.74       262
```