```
Assignment no 5
```

Aim:

1. Logistic Regression
2. Differentiate between Linear and Logistic Regression
3. Sigmoid Function
4. Types of LogisticRegression
5. Confusion Matrix Evaluation Metrics

In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [7]:
```
df=pd.read_csv("diabetes.csv");
df
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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	

768 rows × 9 columns

In [8]:
```
print(df.isnull().sum())
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [6]:

```
cov_matrix = df.cov()
print(cov_matrix)
```

```
                          Pregnancies      Glucose  BloodPressure  \
Pregnancies                 11.354056    13.947131       9.214538   
Glucose                     13.947131  1022.248314      94.430956   
BloodPressure                9.214538    94.430956     374.647271   
SkinThickness               -4.390041    29.239183      64.029396   
Insulin                    -28.555231  1220.935799     198.378412   
BMI                          0.469774    55.726987      43.004695   
DiabetesPedigreeFunction    -0.037426     1.454875       0.264638   
Age                         21.570620    99.082805      54.523453   
Outcome                      0.356618     7.115079       0.600697   

                          SkinThickness       Insulin         BMI  \
Pregnancies                   -4.390041    -28.555231    0.469774   
Glucose                       29.239183   1220.935799   55.726987   
BloodPressure                 64.029396    198.378412   43.004695   
SkinThickness                254.473245    802.979941   49.373869   
Insulin                      802.979941  13281.180078  179.775172   
BMI                           49.373869    179.775172   62.159984   
DiabetesPedigreeFunction       0.972136      7.066681    0.367405   
Age                          -21.381023    -57.143290    3.360330   
Outcome                        0.568747      7.175671    1.100638   

                          DiabetesPedigreeFunction         Age   Outcome  
Pregnancies                              -0.037426   21.570620  0.356618  
Glucose                                   1.454875   99.082805  7.115079  
BloodPressure                             0.264638   54.523453  0.600697  
SkinThickness                             0.972136  -21.381023  0.568747  
Insulin                                   7.066681  -57.143290  7.175671  
BMI                                       0.367405    3.360330  1.100638  
DiabetesPedigreeFunction                  0.109779    0.130772  0.027472  
Age                                       0.130772  138.303046  1.336953  
Outcome                                   0.027472    1.336953  0.227483  
```

In [9]:

```
X = df.drop('Outcome', axis=1)
y = df['Outcome']
```

In [13]:

```
X
```

Out[13]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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	

768 rows × 8 columns

In [11]:

```
y
```

Out[11]:

```
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
```

In [12]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

In [13]:

```
# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Print the shape of the splits to verify
print(f"Training data shape: {X_train.shape}")
print(f"Testing data shape: {X_test.shape}")
```

```
Training data shape: (614, 8)
Testing data shape: (154, 8)
```

In [14]:

```
X_train
```

Out[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
60	2	84	0	0	0	0.0	
618	9	112	82	24	0	28.2	
346	1	139	46	19	83	28.7	
294	0	161	50	0	0	21.9	
231	6	134	80	37	370	46.2	
...	...	...	...	...	...	...	
71	5	139	64	35	140	28.6	
106	1	96	122	0	0	22.4	
270	10	101	86	37	0	45.6	
435	0	141	0	0	0	42.4	
102	0	125	96	0	0	22.5	

614 rows × 8 columns

In [15]:

```
X_test
```

Out[15]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
668	6	98	58	33	190	34.0	
324	2	112	75	32	0	35.7	
624	2	108	64	0	0	30.8	
690	8	107	80	0	0	24.6	
473	7	136	90	0	0	29.9	
...	...	...	...	...	...	...	
355	9	165	88	0	0	30.4	
534	1	77	56	30	56	33.3	
344	8	95	72	0	0	36.8	
296	2	146	70	38	360	28.0	
462	8	74	70	40	49	35.3	

154 rows × 8 columns

In [16]:
```
y_train
```

Out[16]:
```
60     0
618    1
346    0
294    0
231    1
      ..
71     0
106    0
270    1
435    1
102    0
Name: Outcome, Length: 614, dtype: int64
```

In [17]:
```
y_test
```

Out[17]:
```
668    0
324    0
624    0
690    0
473    0
      ..
355    1
534    0
344    0
296    1
462    0
Name: Outcome, Length: 154, dtype: int64
```

In [18]:
```
# Initialize the StandardScaler
scaler = StandardScaler()

# Fit on training data and transform both training and testing data
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [19]:
```
X_train
```

Out[19]:
```
array([[-0.52639686, -1.15139792, -3.75268255, ..., -4.13525578,
        -0.49073479, -1.03594038],
       [ 1.58804586, -0.27664283,  0.68034485, ..., -0.48916881,
         2.41502991,  1.48710085],
       [-0.82846011,  0.56687102, -1.2658623 , ..., -0.42452187,
         0.54916055, -0.94893896],
       ...,
       [ 1.8901091 , -0.62029661,  0.89659009, ...,  1.76054443,
         1.981245  ,  0.44308379],
       [-1.13052335,  0.62935353, -3.75268255, ...,  1.34680407,
        -0.78487662, -0.33992901],
       [-1.13052335,  0.12949347,  1.43720319, ..., -1.22614383,
        -0.61552223, -1.03594038]])
```

In [20]:
```
X_test
```

```
Out[20]: array([[ 0.68185612, -0.71402038, -0.61712658, ...,  0.26073561,
                 -0.11637247,  0.87809089],
                [-0.52639686, -0.27664283,  0.30191569, ...,  0.48053518,
                 -0.954231  , -1.03594038],
                [-0.52639686, -0.40160784, -0.29275872, ..., -0.15300476,
                 -0.9245197 , -1.03594038],
                ...,
                [ 1.28598261, -0.80774414,  0.13973176, ...,  0.62275843,
                  0.04703966,  2.0961108 ],
                [-0.52639686,  0.78555979,  0.03160914, ..., -0.51502758,
                 -0.39268751, -0.33992901],
                [ 1.28598261, -1.46381046,  0.03160914, ...,  0.42881763,
                  0.70068816,  0.53008521]])
```

```
In [21]: # Initialize the Logistic Regression model
         logreg = LogisticRegression(max_iter=1000)

         # Fit the model to the training data
         logreg.fit(X_train, y_train)
```

```
Out[21]: ▾ LogisticRegression
         LogisticRegression(max_iter=1000)
```

```
In [24]: # Predict on the training data
         y_train_pred = logreg.predict(X_train)

         # Predict on the testing data
         y_test_pred = logreg.predict(X_test)
```

```
In [25]: y_train_pred
```

```
Out[25]: array([0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0,
                0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
                1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
                1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0,
                1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
                0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
                1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
                1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1,
                1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
                1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
                0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
                1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0,
                0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
                0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
                0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
                0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
                0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
                0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0],
               dtype=int64)
```

```
In [27]: y_test_pred
```

```
Out[27]: array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
                1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0,
                0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
                0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
                0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1,
                0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
                0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
               dtype=int64)
```

```
In [29]: train_accuracy = accuracy_score(y_train, y_train_pred)
         train_accuracy
```

```
Out[29]: 0.7703583061889251
```

```
In [30]: train_precision = precision_score(y_train, y_train_pred)
         train_precision
```

```
Out[30]: 0.7168674698795181
```

```
In [31]: train_recall = recall_score(y_train, y_train_pred)
         train_recall
```

```
Out[31]: 0.5586854460093896
```

In [32]:
```
train_f1 = f1_score(y_train, y_train_pred)
train_f1
```

Out[32]:
```
0.6279683377308708
```

In [33]:
```
train_cm = confusion_matrix(y_train, y_train_pred)
train_cm
```

Out[33]:
```
array([[354,  47],
       [ 94, 119]], dtype=int64)
```

In [34]:
```
test_accuracy = accuracy_score(y_test, y_test_pred)
test_accuracy
```

Out[34]:
```
0.7532467532467533
```

In [35]:
```
test_precision = precision_score(y_test, y_test_pred)
test_precision
```

Out[35]:
```
0.6491228070175439
```

In [36]:
```
test_recall = recall_score(y_test, y_test_pred)
test_recall
```

Out[36]:
```
0.6727272727272727
```

In [37]:
```
test_f1 = f1_score(y_test, y_test_pred)
test_f1
```

Out[37]:
```
0.6607142857142858
```

In [38]:
```
test_cm = confusion_matrix(y_test, y_test_pred)
test_cm
```

Out[38]:
```
array([[79, 20],
       [18, 37]], dtype=int64)
```

In [ ]:
```
Name:Kadhane Pratiksha
Rollno:13213
Batch:B1
```