In [1]:

```
import pandas as pd
```

In [2]:

```
import numpy as np
```

In [3]:

```
df = pd.read_csv(r'https://github.com/YBI-Foundation/Dataset/raw/main/Diabetes.csv')
```

In [4]:

```
df.head()
```

Out[4]:

| | pregnancies | glucose | diastolic | triceps | insulin | bmi | dpf | age | diabetes |
|---|---|---|---|---|---|---|---|---|---|
| **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 [5]:

```
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   diastolic    768 non-null    int64  
 3   triceps      768 non-null    int64  
 4   insulin      768 non-null    int64  
 5   bmi          768 non-null    float64
 6   dpf          768 non-null    float64
 7   age          768 non-null    int64  
 8   diabetes     768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [6]:

```
df.describe()
```

Out[6]:

| | pregnancies | glucose | diastolic | triceps | insulin | bmi | dpf | 7 |
|---|---|---|---|---|---|---|---|---|
| **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 [7]:

```
df.columns
```

Out[7]:

```
Index(['pregnancies', 'glucose', 'diastolic', 'triceps', 'insulin', 'bmi',
       'dpf', 'age', 'diabetes'],
      dtype='object')
```

In [8]:

```
df.shape
```

Out[8]:

```
(768, 9)
```

In [9]:

```
df['diabetes'].value_counts()
```

Out[9]:

```
0    500
1    268
Name: diabetes, dtype: int64
```

In [10]:

```
df.groupby('diabetes').mean()
```

Out[10]:

| diabetes | pregnancies | glucose | diastolic | triceps | insulin | bmi | dpf | |
|---|---|---|---|---|---|---|---|---|
| 0 | 3.298000 | 109.980000 | 68.184000 | 19.664000 | 68.792000 | 30.304200 | 0.429734 | 31.1 |
| 1 | 4.865672 | 141.257463 | 70.824627 | 22.164179 | 100.335821 | 35.142537 | 0.550500 | 37.0 |

In [11]:

```
y = df['diabetes']
```

In [12]:

```
y.shape
```

Out[12]:

```
(768,)
```

In [13]:

```
y
```

Out[13]:

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: diabetes, Length: 768, dtype: int64
```

In [14]:

```
x = df[['pregnancies', 'glucose', 'diastolic', 'triceps', 'insulin', 'bmi',
       'dpf', 'age']]
```

In [15]:

```
x = df.drop('diabetes',axis=1)
```

In [16]:

```
x.shape
```

Out[16]:

```
(768, 8)
```

In [17]:

```
x
```

Out[17]:

| | pregnancies | glucose | diastolic | triceps | insulin | bmi | dpf | 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 [18]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [19]:

```
mn = MinMaxScaler()
```

In [20]:

```
x = mn.fit_transform(x)
```

In [21]:

```
x
```

Out[21]:

```
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]])
```

In [22]:

```
from sklearn.model_selection import train_test_split
```

In [23]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3, stratify=y, rando
```

In [24]:

```
x_train.shape, x_test.shape, y_train.shape,y_test.shape
```

Out[24]:

```
((537, 8), (231, 8), (537,), (231,))
```

In [25]:

```
from sklearn.linear_model import LogisticRegression
```

In [26]:

```
lr = LogisticRegression()
```

In [27]:

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

Out[27]:

```
LogisticRegression()
```

In [28]:

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

In [29]:

```
y_pred
```

Out[29]:

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

In [30]:

```
lr.predict_proba(x_test)
```

Out[30]:

```
array([[0.71101198, 0.28898802],
       [0.80246044, 0.19753956],
       [0.50085081, 0.49914919],
       [0.8745601 , 0.1254399 ],
       [0.84313967, 0.15686033],
       [0.72965238, 0.27034762],
       [0.32611128, 0.67388872],
       [0.82905388, 0.17094612],
       [0.57764733, 0.42235267],
       [0.5794767 , 0.4205233 ],
       [0.90475455, 0.09524545],
       [0.42428281, 0.57571719],
       [0.81659611, 0.18340389],
       [0.86057018, 0.13942982],
       [0.55629153, 0.44370847],
       [0.83208198, 0.16791802],
       [0.40636481, 0.59363519],
       [0.8430081 , 0.1569919 ],
```

In [31]:

```
from sklearn.metrics import confusion_matrix, classification_report
```

In [32]:

```
print(confusion_matrix(y_test, y_pred))
```

```
[[136  14]
 [ 37  44]]
```

In [33]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.91      0.84       150
           1       0.76      0.54      0.63        81

    accuracy                           0.78       231
   macro avg       0.77      0.72      0.74       231
weighted avg       0.78      0.78      0.77       231
```

In [34]:

```
x_new = df.sample(1)
```

In [35]:

```
x_new
```

Out[35]:

| | pregnancies | glucose | diastolic | triceps | insulin | bmi | dpf | age | diabetes |
|---|---|---|---|---|---|---|---|---|---|
| **416** | 1 | 97 | 68 | 21 | 0 | 27.2 | 1.095 | 22 | 0 |

In [36]:

```
x_new.shape
```

Out[36]:

```
(1, 9)
```

In [37]:

```
x_new = x_new.drop('diabetes', axis=1)
```

In [38]:

```
x_new
```

Out[38]:

| | pregnancies | glucose | diastolic | triceps | insulin | bmi | dpf | age |
|---|---|---|---|---|---|---|---|---|
| **416** | 1 | 97 | 68 | 21 | 0 | 27.2 | 1.095 | 22 |

In [39]:

```
x_new.shape
```

Out[39]:

```
(1, 8)
```

In [40]:

```
x_new = mn.fit_transform(x_new)
```

In [41]:

```
y_pred_new = lr.predict(x_new)
```

In [42]:

```
y_pred_new
```

Out[42]:

```
array([0], dtype=int64)
```

In [43]:

```
lr.predict_proba(x_new)
```

Out[43]:

```
array([[0.99508059, 0.00491941]])
```

In [ ]:

```

```