Diabetes Prediction with Logistics Regression

Import Library

In [2]:

```
import pandas as pd
import numpy as np
```

Import CSV as DataFrame

In [3]:

```
df = pd.read_csv("Dataset-main/Dataset-main/Diabetes.csv")
```

Get the First Five Rows of Dataframe

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

Get Information of DataFrame

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
```

Get Missing Values Drop

In [6]:

```
df = df.dropna()
```

Get the summary Statistics

In [7]:

```
df.describe()
```

Out[7]:

	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	7
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	

Get Unique Values(class or label)in y variable

In [56]:

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

Out[56]:

```
diabetes
0           500
1           268
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

Get Column Names

In [11]:

```
df.columns
```

Out[11]:

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

Get Shape of Dataframe

In [12]:

```
df.shape
```

Out[12]:

```
(768, 9)
```

Define y(dependent or label or target variable) and X(independent or features or attribute Variable

In [13]:

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

In [14]:

```
y.shape
```

Out[14]:

```
(768,)
```

In [15]:

```
y
```

Out[15]:

```
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 [16]:

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

or use drop function to define X

In [17]:

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

In [18]:

```
X.shape
```

Out[18]:

```
(768, 8)
```

In [19]:

```
X
```

Out[19]:

	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

Get X Variables Standardized

In [24]:

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

In [25]:

```
mm= MinMaxScaler()
```

In [26]:

```
X = mm.fit_transform(X)
```

In [27]:

```
X
```

Out[27]:

```
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]])
```

Get Train Test Split

In [28]:

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

In [29]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3, random_state=2529)
```

In [30]:

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[30]:

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

Get Model Train

In [38]:

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

In [39]:

```
lr = LogisticRegression()
```

In [40]:

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

Out[40]:

```
LogisticRegression()
```

Get Model Prediction

In [41]:

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

In [42]:

```
y_pred.shape
```

Out[42]:

```
(231,)
```

In [43]:

```
y_pred
```

Out[43]:

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

Get Probability of each predicted class

In [44]:

```
lr.predict_proba
```

Out[44]:

```
array([[0.87946998, 0.12053002],
       [0.59958843, 0.40041157],
       [0.33869015, 0.66130985],
       [0.70625215, 0.29374785],
       [0.55893242, 0.44106758],
       [0.36724463, 0.63275537],
       [0.16360933, 0.83639067],
       [0.30066834, 0.69933166],
       [0.91142377, 0.08857623],
       [0.49981736, 0.50018264],
       [0.92067479, 0.07932521],
       [0.27488402, 0.72511598],
       [0.75356548, 0.24643452],
       [0.92015134, 0.07984866],
       [0.81541256, 0.18458744],
       [0.47341943, 0.52658057],
       [0.37791835, 0.62208165],
       [0.55553959, 0.44446041],
```

Get Future Prediction

In [46]:

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

In [47]:

```
X_new
```

Out[47]:

	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age	diabetes
388	5	144	82	26	285	32.0	0.452	58	1

In [48]:

```
X_new.shape
```

Out[48]:

```
(1, 9)
```

In [49]:

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

In [50]:

```
X_new
```

Out[50]:

	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age
388	5	144	82	26	285	32.0	0.452	58

In [51]:

```
X_new.shape
```

Out[51]:

```
(1, 8)
```

In [52]:

```
X_new = mm.fit_transform(X_new)
```

In [53]:

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

In [54]:

```
y_pred_new
```

Out[54]:

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

In [55]:

```
lr.predict_proba(X_new)
```

Out[55]:

```
array([[0.9928188, 0.0071812]])
```

Predicted and actual class is zero that is non diabetes