```
import pandas as pd

import numpy as np

df =
pd.read_csv(r'https://raw.githubusercontent.com/Sarvesh-S-Patil/Dataset/main/Diabetes.csv')

df.head()
```

	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

```
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
```

```
df.describe()
```

```
       pregnancies     glucose   diastolic     triceps     insulin  \
count   768.000000  768.000000  768.000000  768.000000  768.000000
mean      3.845052  120.894531   69.105469   20.536458   79.799479
std       3.369578   31.972618   19.355807   15.952218  115.244002
min       0.000000    0.000000    0.000000    0.000000    0.000000
25%       1.000000   99.000000   62.000000    0.000000    0.000000
50%       3.000000  117.000000   72.000000   23.000000   30.500000
75%       6.000000  140.250000   80.000000   32.000000  127.250000
max      17.000000  199.000000  122.000000   99.000000  846.000000

              bmi         dpf         age    diabetes
count  768.000000  768.000000  768.000000  768.000000
mean    31.992578    0.471876   33.240885    0.348958
std      7.884160    0.331329   11.760232    0.476951
min      0.000000    0.078000   21.000000    0.000000
25%     27.300000    0.243750   24.000000    0.000000
50%     32.000000    0.372500   29.000000    0.000000
75%     36.600000    0.626250   41.000000    1.000000
max     67.100000    2.420000   81.000000    1.000000
```

```
df.columns
```

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

```
df.shape
```

```
(768, 9)
```

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

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

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

```
           pregnancies     glucose  diastolic    triceps     insulin  \
diabetes
0             3.298000  109.980000  68.184000  19.664000   68.792000
1             4.865672  141.257463  70.824627  22.164179  100.335821

                 bmi       dpf        age
diabetes
0          30.304200  0.429734  31.190000
1          35.142537  0.550500  37.067164
```

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

```
y.shape
```

```
(768,)
```

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

```
X.shape
```

```
(768, 8)
```

```
X
```

```
     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 x 8 columns]
```

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

```
mm = MinMaxScaler()
```

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

```
X
```

```
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]])
```

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

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

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

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

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

```
lr= LogisticRegression()
```

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

```
LogisticRegression()
```

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

```
y_pred.shape
```

```
(231,)
```

```
y_pred
```

```
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, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0,
1,
       0, 0, 0, 1, 0, 1, 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, 1, 0, 0, 1, 0,
```

```
0,
       0, 1, 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])
```

```
lr.predict_proba(X_test)
```

```
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],
       [0.89719093, 0.10280907],
       [0.81016432, 0.18983568],
       [0.46241262, 0.53758738],
       [0.46405765, 0.53594235],
       [0.72861372, 0.27138628],
       [0.93939119, 0.06060881],
       [0.79376047, 0.20623953],
       [0.80186616, 0.19813384],
       [0.73978527, 0.26021473],
       [0.51847931, 0.48152069],
       [0.88234404, 0.11765596],
       [0.69813424, 0.30186576],
       [0.83657192, 0.16342808],
       [0.59342286, 0.40657714],
       [0.72484771, 0.27515229],
       [0.57114241, 0.42885759],
       [0.41477956, 0.58522044],
       [0.64509232, 0.35490768],
       [0.82352482, 0.17647518],
       [0.56799328, 0.43200672],
       [0.67719094, 0.32280906],
       [0.09494306, 0.90505694],
       [0.87871417, 0.12128583],
       [0.9536179 , 0.0463821 ],
```

[0.87167582, 0.12832418],
[0.553921 , 0.446079],
[0.84850974, 0.15149026],
[0.44021055, 0.55978945],
[0.27858939, 0.72141061],
[0.86708573, 0.13291427],
[0.56490259, 0.43509741],
[0.57918011, 0.42081989],
[0.90345211, 0.09654789],
[0.21480901, 0.78519099],
[0.65441664, 0.34558336],
[0.85164027, 0.14835973],
[0.71149375, 0.28850625],
[0.65555732, 0.34444268],
[0.51846263, 0.48153737],
[0.90189862, 0.09810138],
[0.84636396, 0.15363604],
[0.42151763, 0.57848237],
[0.62161121, 0.37838879],
[0.84755578, 0.15244422],
[0.54507536, 0.45492464],
[0.57535021, 0.42464979],
[0.65333783, 0.34666217],
[0.06811223, 0.93188777],
[0.92416976, 0.07583024],
[0.92474842, 0.07525158],
[0.23500507, 0.76499493],
[0.71430332, 0.28569668],
[0.21261979, 0.78738021],
[0.73685153, 0.26314847],
[0.87628786, 0.12371214],
[0.71687358, 0.28312642],
[0.6839625 , 0.3160375],
[0.91460737, 0.08539263],
[0.88963275, 0.11036725],
[0.84389251, 0.15610749],
[0.85244033, 0.14755967],
[0.86094998, 0.13905002],
[0.79589851, 0.20410149],
[0.77074122, 0.22925878],
[0.43993416, 0.56006584],
[0.47162187, 0.52837813],
[0.77190026, 0.22809974],
[0.66643374, 0.33356626],
[0.23061136, 0.76938864],
[0.89791071, 0.10208929],
[0.5169116 , 0.4830884],
[0.58477607, 0.41522393],
[0.45677241, 0.54322759],
[0.73145682, 0.26854318],

```
[0.37146126, 0.62853874],
[0.63810902, 0.36189098],
[0.90786586, 0.09213414],
[0.66751434, 0.33248566],
[0.72492366, 0.27507634],
[0.74697793, 0.25302207],
[0.86403863, 0.13596137],
[0.7102379 , 0.2897621 ],
[0.39966339, 0.60033661],
[0.75963403, 0.24036597],
[0.7813349 , 0.2186651 ],
[0.17834736, 0.82165264],
[0.3536907 , 0.6463093 ],
[0.78624291, 0.21375709],
[0.86262741, 0.13737259],
[0.38557914, 0.61442086],
[0.81053764, 0.18946236],
[0.2919247 , 0.7080753 ],
[0.6051475 , 0.3948525 ],
[0.8273839 , 0.1726161 ],
[0.93162424, 0.06837576],
[0.23180927, 0.76819073],
[0.43186986, 0.56813014],
[0.57282164, 0.42717836],
[0.81051094, 0.18948906],
[0.91427868, 0.08572132],
[0.8756462 , 0.1243538 ],
[0.81819644, 0.18180356],
[0.96319716, 0.03680284],
[0.33163505, 0.66836495],
[0.86514402, 0.13485598],
[0.42315142, 0.57684858],
[0.92748233, 0.07251767],
[0.77039684, 0.22960316],
[0.20583091, 0.79416909],
[0.85148598, 0.14851402],
[0.34505764, 0.65494236],
[0.80627697, 0.19372303],
[0.75098324, 0.24901676],
[0.28819963, 0.71180037],
[0.77728558, 0.22271442],
[0.82589476, 0.17410524],
[0.796746  , 0.203254  ],
[0.17507984, 0.82492016],
[0.65221042, 0.34778958],
[0.19711733, 0.80288267],
[0.85258415, 0.14741585],
[0.63010674, 0.36989326],
[0.59597098, 0.40402902],
[0.86158419, 0.13841581],
```

[0.92507023, 0.07492977],
[0.27290246, 0.72709754],
[0.80838528, 0.19161472],
[0.51108718, 0.48891282],
[0.92176678, 0.07823322],
[0.80450452, 0.19549548],
[0.40840359, 0.59159641],
[0.87767655, 0.12232345],
[0.79830982, 0.20169018],
[0.83773197, 0.16226803],
[0.82455107, 0.17544893],
[0.5683309 , 0.4316691],
[0.55346901, 0.44653099],
[0.37372747, 0.62627253],
[0.34152306, 0.65847694],
[0.75005039, 0.24994961],
[0.89622943, 0.10377057],
[0.7680207 , 0.2319793],
[0.87326608, 0.12673392],
[0.66811506, 0.33188494],
[0.79926994, 0.20073006],
[0.70446539, 0.29553461],
[0.73328241, 0.26671759],
[0.93717963, 0.06282037],
[0.9117848 , 0.0882152],
[0.67819169, 0.32180831],
[0.88819295, 0.11180705],
[0.7835728 , 0.2164272],
[0.48985685, 0.51014315],
[0.52881368, 0.47118632],
[0.66731401, 0.33268599],
[0.45997085, 0.54002915],
[0.88738774, 0.11261226],
[0.83444701, 0.16555299],
[0.68903274, 0.31096726],
[0.46069194, 0.53930806],
[0.68984368, 0.31015632],
[0.80706073, 0.19293927],
[0.8788617 , 0.1211383],
[0.81944606, 0.18055394],
[0.7056525 , 0.2943475],
[0.64591096, 0.35408904],
[0.77320963, 0.22679037],
[0.86180236, 0.13819764],
[0.73438344, 0.26561656],
[0.94885722, 0.05114278],
[0.36951477, 0.63048523],
[0.90782335, 0.09217665],
[0.7539445 , 0.2460555],
[0.84642448, 0.15357552],

```
       [0.78149376, 0.21850624],
       [0.5867649 , 0.4132351 ],
       [0.79824569, 0.20175431],
       [0.78464293, 0.21535707],
       [0.58425606, 0.41574394],
       [0.75562197, 0.24437803],
       [0.54093834, 0.45906166],
       [0.73967456, 0.26032544],
       [0.64834867, 0.35165133],
       [0.60771315, 0.39228685],
       [0.59999211, 0.40000789],
       [0.86593051, 0.13406949],
       [0.7529388 , 0.2470612 ],
       [0.3754114 , 0.6245886 ],
       [0.91340651, 0.08659349],
       [0.51641857, 0.48358143],
       [0.58374308, 0.41625692],
       [0.57742012, 0.42257988],
       [0.82290398, 0.17709602],
       [0.73823106, 0.26176894],
       [0.54146027, 0.45853973],
       [0.54038396, 0.45961604],
       [0.63549036, 0.36450964],
       [0.88789892, 0.11210108],
       [0.73770319, 0.26229681],
       [0.88245076, 0.11754924],
       [0.35815819, 0.64184181],
       [0.87803515, 0.12196485],
       [0.46308121, 0.53691879],
       [0.83385811, 0.16614189],
       [0.4959721 , 0.5040279 ],
       [0.29372698, 0.70627302],
       [0.79971627, 0.20028373],
       [0.8500601 , 0.1499399 ],
       [0.86085878, 0.13914122],
       [0.62447982, 0.37552018],
       [0.84467594, 0.15532406],
       [0.67191841, 0.32808159],
       [0.42838914, 0.57161086]])
```

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

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

```
[[133  12]
 [ 46  40]]
```

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

```
              precision    recall  f1-score   support
```

```
           0       0.74      0.92      0.82       145
           1       0.77      0.47      0.58        86

    accuracy                           0.75       231
   macro avg       0.76      0.69      0.70       231
weighted avg       0.75      0.75      0.73       231
```

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

```
X_new
```

```
    pregnancies  glucose  diastolic  triceps  insulin   bmi    dpf  age
\
96            2       92         62       28        0  31.6  0.13   24


    diabetes
96         0
```

```
X_new.shape
```

```
(1, 9)
```

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

```
X_new
```

```
    pregnancies  glucose  diastolic  triceps  insulin   bmi    dpf  age
96            2       92         62       28        0  31.6  0.13   24
```

```
X_new.shape
```

```
(1, 8)
```

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

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

```
y_pred_new
```

```
array([0])
```

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

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

```
## Actual and Predicted Class is 0 that is Non-Diabetic
```