[1]:
```
import pandas as pd
import numpy as np
```

[3]:
```
df=pd.read_csv('https://raw.githubusercontent.com/YBI-Foundation/Dataset/main/
↪Diabetes.csv')
```

[4]:
```
df.head()
```

[4]:

	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	

	diabetes
0	1
1	0
2	1
3	0
4	1

[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
```

[6]:
```
df.describe()
```

[6]:

	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

[7]:
```
df.columns
```

[7]:
```
Index(['pregnancies', 'glucose', 'diastolic', 'triceps', 'insulin', 'bmi',
       'dpf', 'age', 'diabetes'],
      dtype='object')
```

[8]:
```
df.shape
```

[8]:
```
(768, 9)
```

[9]:
```
df['diabetes'].value_counts()
```

[9]:
```
0    500
1    268
Name: diabetes, dtype: int64
```

[10]:
```
df.groupby('diabetes').mean()
```

[10]:

diabetes	pregnancies	glucose	diastolic	triceps	insulin	\
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
```

[11]:
```
y=df['diabetes']
```

[12]:
```
y.shape
```

[12]:
```
(768,)
```

[13]:
```
y
```

[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
```

[14]:
```
x=df[['pregnancies','glucose','diastolic','triceps','insulin','bmi','dpf','age']]
```

[15]:
```
x.shape
```

[15]:
```
(768, 8)
```

[16]:
```
x
```

[16]:
```
     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]
```

[17]:
```
from sklearn.preprocessing import MinMaxScaler
```

[18]:
```
mm=MinMaxScaler()
```

[19]:
```
x=mm.fit_transform(x)
```

[20]:
```
x
```

[20]:
```
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]])
```

[21]:
```
from sklearn.model_selection import train_test_split
```

[22]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪3,stratify=y,random_state=2529)
```

[23]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

[23]:
```
((537, 8), (231, 8), (537,), (231,))
```

[24]:
```
from sklearn.linear_model import LogisticRegression
```

[25]:
```
lr=LogisticRegression()
```

[26]:
```
lr.fit(x_train,y_train)
```

[26]:
```
LogisticRegression()
```

[27]:
```
y_pred=lr.predict(x_test)
```

[28]:
```
y_pred.shape
```

[28]:
```
(231,)
```

[29]:
```
y_pred
```

[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)
```

[30]:
```
lr.predict_proba(x_test)
```

[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 ],
       [0.6035823 , 0.3964177 ],
       [0.51982645, 0.48017355],
       [0.65174255, 0.34825745],
       [0.89662971, 0.10337029],
       [0.88362346, 0.11637654],
       [0.50529753, 0.49470247],
       [0.74048922, 0.25951078],
       [0.38010129, 0.61989871],
       [0.86743064, 0.13256936],
       [0.63051126, 0.36948874],
       [0.54593476, 0.45406524],
       [0.16753486, 0.83246514],
       [0.62023267, 0.37976733],
```

[0.45959131, 0.54040869],
[0.75494925, 0.24505075],
[0.71542708, 0.28457292],
[0.75628148, 0.24371852],
[0.77697399, 0.22302601],
[0.67351753, 0.32648247],
[0.79929534, 0.20070466],
[0.74875692, 0.25124308],
[0.54844936, 0.45155064],
[0.32997346, 0.67002654],
[0.22941801, 0.77058199],
[0.7448794 , 0.2551206],
[0.34550793, 0.65449207],
[0.18608056, 0.81391944],
[0.26374953, 0.73625047],
[0.26523806, 0.73476194],
[0.62304765, 0.37695235],
[0.68999299, 0.31000701],
[0.45692191, 0.54307809],
[0.62161158, 0.37838842],
[0.79037029, 0.20962971],
[0.89387086, 0.10612914],
[0.73010557, 0.26989443],
[0.2052682 , 0.7947318],
[0.39976501, 0.60023499],
[0.83575072, 0.16424928],
[0.71212552, 0.28787448],
[0.63424484, 0.36575516],
[0.77606187, 0.22393813],
[0.87827346, 0.12172654],
[0.16798753, 0.83201247],
[0.70394651, 0.29605349],
[0.31241475, 0.68758525],
[0.82160552, 0.17839448],
[0.78797643, 0.21202357],
[0.42213055, 0.57786945],
[0.73989432, 0.26010568],
[0.45067804, 0.54932196],
[0.57268746, 0.42731254],
[0.64118353, 0.35881647],
[0.49212219, 0.50787781],
[0.79012394, 0.20987606],
[0.16338081, 0.83661919],
[0.86204005, 0.13795995],
[0.82874624, 0.17125376],
[0.90111805, 0.09888195],
[0.42743135, 0.57256865],

[0.85635466, 0.14364534],
[0.72906087, 0.27093913],
[0.90634904, 0.09365096],
[0.98369882, 0.01630118],
[0.82421937, 0.17578063],
[0.6888735 , 0.3111265],
[0.17744042, 0.82255958],
[0.7589098 , 0.2410902],
[0.61017603, 0.38982397],
[0.61159465, 0.38840535],
[0.68572538, 0.31427462],
[0.69269697, 0.30730303],
[0.6388536 , 0.3611464],
[0.47555626, 0.52444374],
[0.65691226, 0.34308774],
[0.80652822, 0.19347178],
[0.69140851, 0.30859149],
[0.61027426, 0.38972574],
[0.72605388, 0.27394612],
[0.9770489 , 0.0229511],
[0.89462332, 0.10537668],
[0.78246264, 0.21753736],
[0.43402032, 0.56597968],
[0.83264104, 0.16735896],
[0.63215189, 0.36784811],
[0.67536022, 0.32463978],
[0.89600414, 0.10399586],
[0.13939148, 0.86060852],
[0.27983916, 0.72016084],
[0.57736633, 0.42263367],
[0.66005884, 0.33994116],
[0.30464973, 0.69535027],
[0.82983229, 0.17016771],
[0.58001121, 0.41998879],
[0.83908052, 0.16091948],
[0.40127932, 0.59872068],
[0.78945502, 0.21054498],
[0.76778092, 0.23221908],
[0.54452662, 0.45547338],
[0.60544688, 0.39455312],
[0.44618523, 0.55381477],
[0.66314271, 0.33685729],
[0.34350835, 0.65649165],
[0.59893756, 0.40106244],
[0.62634271, 0.37365729],
[0.77608313, 0.22391687],
[0.72677569, 0.27322431],

[0.43937286, 0.56062714],
[0.54233391, 0.45766609],
[0.25377755, 0.74622245],
[0.45760086, 0.54239914],
[0.76638944, 0.23361056],
[0.83819204, 0.16180796],
[0.61346835, 0.38653165],
[0.90356153, 0.09643847],
[0.7710563 , 0.2289437],
[0.87707299, 0.12292701],
[0.65745324, 0.34254676],
[0.68817657, 0.31182343],
[0.80657917, 0.19342083],
[0.35894968, 0.64105032],
[0.68564186, 0.31435814],
[0.59189724, 0.40810276],
[0.51674781, 0.48325219],
[0.81212504, 0.18787496],
[0.67742839, 0.32257161],
[0.86369508, 0.13630492],
[0.33984056, 0.66015944],
[0.4070685 , 0.5929315],
[0.82227943, 0.17772057],
[0.44964935, 0.55035065],
[0.38206823, 0.61793177],
[0.71804561, 0.28195439],
[0.70550659, 0.29449341],
[0.79448399, 0.20551601],
[0.85483671, 0.14516329],
[0.82496174, 0.17503826],
[0.67704275, 0.32295725],
[0.77732205, 0.22267795],
[0.7360905 , 0.2639095],
[0.32375744, 0.67624256],
[0.85116323, 0.14883677],
[0.72815553, 0.27184447],
[0.21054529, 0.78945471],
[0.87112955, 0.12887045],
[0.44318092, 0.55681908],
[0.69806954, 0.30193046],
[0.52312157, 0.47687843],
[0.30924146, 0.69075854],
[0.80715878, 0.19284122],
[0.44392423, 0.55607577],
[0.94261157, 0.05738843],
[0.59928428, 0.40071572],
[0.87563746, 0.12436254],

[0.58419607, 0.41580393],
[0.52821088, 0.47178912],
[0.40153771, 0.59846229],
[0.64992152, 0.35007848],
[0.63185221, 0.36814779],
[0.93554367, 0.06445633],
[0.31681777, 0.68318223],
[0.54968842, 0.45031158],
[0.80598623, 0.19401377],
[0.77826867, 0.22173133],
[0.49958936, 0.50041064],
[0.19034282, 0.80965718],
[0.26781888, 0.73218112],
[0.57488252, 0.42511748],
[0.81074537, 0.18925463],
[0.76494237, 0.23505763],
[0.64189875, 0.35810125],
[0.80876086, 0.19123914],
[0.79446906, 0.20553094],
[0.83421376, 0.16578624],
[0.78869817, 0.21130183],
[0.87464945, 0.12535055],
[0.86368861, 0.13631139],
[0.76845696, 0.23154304],
[0.12908971, 0.87091029],
[0.49597443, 0.50402557],
[0.74991355, 0.25008645],
[0.77513673, 0.22486327],
[0.88357496, 0.11642504],
[0.83739596, 0.16260404],
[0.25695317, 0.74304683],
[0.57208324, 0.42791676],
[0.6407614 , 0.3592386],
[0.79016699, 0.20983301],
[0.70325927, 0.29674073],
[0.86981183, 0.13018817],
[0.77606671, 0.22393329],
[0.82007573, 0.17992427],
[0.79975104, 0.20024896],
[0.62766827, 0.37233173],
[0.51468356, 0.48531644],
[0.35427703, 0.64572297],
[0.76513826, 0.23486174],
[0.47215038, 0.52784962],
[0.32587725, 0.67412275],
[0.53598641, 0.46401359],
[0.74870476, 0.25129524],

```
[0.22461514, 0.77538486],
[0.1603111 , 0.8396889 ],
[0.64150356, 0.35849644],
[0.61681536, 0.38318464],
[0.74800921, 0.25199079],
[0.69366962, 0.30633038],
[0.67352852, 0.32647148],
[0.75764762, 0.24235238],
[0.89764562, 0.10235438],
[0.41733016, 0.58266984],
[0.67419914, 0.32580086],
[0.78132855, 0.21867145]])
```

[31]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

[32]:
```
print(confusion_matrix(y_test,y_pred))
```

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

[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
```

[34]:
```
x_new=df.sample(1)
```

[35]:
```
x_new
```

[35]:

	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age	diabetes
20	3	126	88	41	235	39.3	0.704	27	0

[36]:
```
x_new.shape
```

[36]:
```
(1, 9)
```

[37]:
```
x_new=x_new.drop('diabetes',axis=1)
```

[38]:
```
x_new
```

[38]:

	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age
20	3	126	88	41	235	39.3	0.704	27

[39]:
```
x_new.shape
```

[39]:
```
(1, 8)
```

[41]:
```
x_new=mm.fit_transform(x_new)
```

[42]:
```
y_pred_new=lr.predict(x_new)
```

[43]:
```
y_pred_new
```

[43]:
```
array([0], dtype=int64)
```

[44]:
```
lr.predict_proba(x_new)
```

[44]:
```
array([[0.99508059, 0.00491941]])
```

[ ]:
```
#predicted and actual class is zero that is non diabetic
```