Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

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

Get Information of Dataframe

```
df.info()   #gives column name, count,  not null category, D-type(data type)
```

```
<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()   #gives the linear relation of each column with another column
```

```
       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.head()
```

```
   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
```

```
df.isnull().sum() #(df.isna().sum() gives same result)
#gives the sum of all null values columns-wise
```

```
pregnancies    0
glucose        0
diastolic      0
triceps        0
insulin        0
bmi            0
dpf            0
age            0
diabetes       0
dtype: int64
```

```
df.nunique()   #gives total no. of unique entries
```

```
pregnancies     17
glucose        136
diastolic       47
triceps         51
```

```
insulin        186
bmi            248
dpf            517
age             52
diabetes         2
dtype: int64
```

```
df.columns  #give column names in the dataframe
```

```
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
```

Setting X and y

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

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

```
X.shape,y.shape
```

```
((768, 8), (768,))
```

#Standardization

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

```
mn= MinMaxScaler()
```

```
X=mn.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]])
```

Train Test Split

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

```
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,stratify=y,random_state=2323)
```

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

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

Get Model Test

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

```
lr= LogisticRegression()
```

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

```
LogisticRegression()
```

Get Model Prediction

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

```
y_pred.shape
```

```
(231,)
```

```
y_pred
```

```
array([0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
0,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
1,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 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,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1])
```

Get Probability of Each Predicttted Class

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

```
array([[0.75404448, 0.24595552],
       [0.58011549, 0.41988451],
       [0.83087557, 0.16912443],
       [0.44991098, 0.55008902],
       [0.24067598, 0.75932402],
       [0.57446319, 0.42553681],
       [0.77914603, 0.22085397],
       [0.78824961, 0.21175039],
       [0.82014912, 0.17985088],
       [0.40766871, 0.59233129],
       [0.84173513, 0.15826487],
       [0.71011999, 0.28988001],
       [0.35276382, 0.64723618],
       [0.54258389, 0.45741611],
       [0.62175918, 0.37824082],
       [0.48752947, 0.51247053],
       [0.85285985, 0.14714015],
       [0.73075664, 0.26924336],
       [0.42678927, 0.57321073],
```

[0.87555195, 0.12444805],
[0.93145641, 0.06854359],
[0.78211464, 0.21788536],
[0.81093474, 0.18906526],
[0.52652421, 0.47347579],
[0.24033188, 0.75966812],
[0.3754181 , 0.6245819],
[0.28492561, 0.71507439],
[0.75048549, 0.24951451],
[0.78618566, 0.21381434],
[0.58885993, 0.41114007],
[0.65412037, 0.34587963],
[0.74828704, 0.25171296],
[0.89088498, 0.10911502],
[0.8227397 , 0.1772603],
[0.76786986, 0.23213014],
[0.68286013, 0.31713987],
[0.8354921 , 0.1645079],
[0.736903 , 0.263097],
[0.85925932, 0.14074068],
[0.47207179, 0.52792821],
[0.78211398, 0.21788602],
[0.71047114, 0.28952886],
[0.87483949, 0.12516051],
[0.69324611, 0.30675389],
[0.810624 , 0.189376],
[0.69028021, 0.30971979],
[0.85742291, 0.14257709],
[0.36142562, 0.63857438],
[0.58756841, 0.41243159],
[0.53292639, 0.46707361],
[0.77474382, 0.22525618],
[0.69334301, 0.30665699],
[0.61471889, 0.38528111],
[0.75166587, 0.24833413],
[0.81914688, 0.18085312],
[0.83953401, 0.16046599],
[0.71193223, 0.28806777],
[0.92795153, 0.07204847],
[0.48506462, 0.51493538],
[0.55888007, 0.44111993],
[0.64426592, 0.35573408],
[0.77053961, 0.22946039],
[0.84821982, 0.15178018],
[0.51724931, 0.48275069],
[0.27138916, 0.72861084],
[0.53912382, 0.46087618],
[0.82811475, 0.17188525],
[0.21434206, 0.78565794],
[0.62703133, 0.37296867],

[0.43278673, 0.56721327],
[0.37571434, 0.62428566],
[0.86300984, 0.13699016],
[0.85956419, 0.14043581],
[0.8431914 , 0.1568086],
[0.54279577, 0.45720423],
[0.8632459 , 0.1367541],
[0.38233981, 0.61766019],
[0.45581406, 0.54418594],
[0.63011888, 0.36988112],
[0.78166241, 0.21833759],
[0.3749337 , 0.6250663],
[0.61347218, 0.38652782],
[0.66526904, 0.33473096],
[0.68913389, 0.31086611],
[0.58306733, 0.41693267],
[0.7159413 , 0.2840587],
[0.50652028, 0.49347972],
[0.94660945, 0.05339055],
[0.79374314, 0.20625686],
[0.45537602, 0.54462398],
[0.48025852, 0.51974148],
[0.58393132, 0.41606868],
[0.33198793, 0.66801207],
[0.66499443, 0.33500557],
[0.75931154, 0.24068846],
[0.59691092, 0.40308908],
[0.77536111, 0.22463889],
[0.55620275, 0.44379725],
[0.79268144, 0.20731856],
[0.84703395, 0.15296605],
[0.66581107, 0.33418893],
[0.70450916, 0.29549084],
[0.91808544, 0.08191456],
[0.19421329, 0.80578671],
[0.88297208, 0.11702792],
[0.77956932, 0.22043068],
[0.49758117, 0.50241883],
[0.24896875, 0.75103125],
[0.77713535, 0.22286465],
[0.86180405, 0.13819595],
[0.87532523, 0.12467477],
[0.3745498 , 0.6254502],
[0.7149339 , 0.2850661],
[0.98143204, 0.01856796],
[0.89544552, 0.10455448],
[0.74157272, 0.25842728],
[0.8438545 , 0.1561455],
[0.5222139 , 0.4777861],
[0.13547318, 0.86452682],

[0.8892975 , 0.1107025],
[0.19530363, 0.80469637],
[0.44488204, 0.55511796],
[0.64065098, 0.35934902],
[0.65534889, 0.34465111],
[0.68592346, 0.31407654],
[0.88924631, 0.11075369],
[0.58205869, 0.41794131],
[0.77921388, 0.22078612],
[0.60244652, 0.39755348],
[0.56292184, 0.43707816],
[0.91770172, 0.08229828],
[0.70883263, 0.29116737],
[0.66686797, 0.33313203],
[0.43335094, 0.56664906],
[0.738877 , 0.261123],
[0.50261465, 0.49738535],
[0.88226797, 0.11773203],
[0.72186195, 0.27813805],
[0.77073343, 0.22926657],
[0.89946708, 0.10053292],
[0.83579707, 0.16420293],
[0.42447122, 0.57552878],
[0.54979367, 0.45020633],
[0.60940055, 0.39059945],
[0.75572808, 0.24427192],
[0.55082349, 0.44917651],
[0.39951251, 0.60048749],
[0.68814555, 0.31185445],
[0.7518128 , 0.2481872],
[0.67833514, 0.32166486],
[0.79561106, 0.20438894],
[0.84018664, 0.15981336],
[0.82288345, 0.17711655],
[0.56953665, 0.43046335],
[0.5380142 , 0.4619858],
[0.6579394 , 0.3420606],
[0.83632279, 0.16367721],
[0.67582403, 0.32417597],
[0.43004045, 0.56995955],
[0.87755577, 0.12244423],
[0.79325166, 0.20674834],
[0.67265409, 0.32734591],
[0.90125202, 0.09874798],
[0.72390658, 0.27609342],
[0.7271594 , 0.2728406],
[0.71183539, 0.28816461],
[0.69814841, 0.30185159],
[0.32015269, 0.67984731],
[0.65104446, 0.34895554],

[0.71715744, 0.28284256],
[0.85059312, 0.14940688],
[0.84556033, 0.15443967],
[0.21633391, 0.78366609],
[0.71956365, 0.28043635],
[0.82043329, 0.17956671],
[0.26909491, 0.73090509],
[0.83906667, 0.16093333],
[0.7193597 , 0.2806403],
[0.51346787, 0.48653213],
[0.78400475, 0.21599525],
[0.20265841, 0.79734159],
[0.88677259, 0.11322741],
[0.22883937, 0.77116063],
[0.45261808, 0.54738192],
[0.85199807, 0.14800193],
[0.48990139, 0.51009861],
[0.87621026, 0.12378974],
[0.50837405, 0.49162595],
[0.94376962, 0.05623038],
[0.67054359, 0.32945641],
[0.52221534, 0.47778466],
[0.36513331, 0.63486669],
[0.83299387, 0.16700613],
[0.82467835, 0.17532165],
[0.70198396, 0.29801604],
[0.78764157, 0.21235843],
[0.80443778, 0.19556222],
[0.87367299, 0.12632701],
[0.86065462, 0.13934538],
[0.33184058, 0.66815942],
[0.76074848, 0.23925152],
[0.97442309, 0.02557691],
[0.54335277, 0.45664723],
[0.63984891, 0.36015109],
[0.75988904, 0.24011096],
[0.75398636, 0.24601364],
[0.87036973, 0.12963027],
[0.61075733, 0.38924267],
[0.83727483, 0.16272517],
[0.77846712, 0.22153288],
[0.70789929, 0.29210071],
[0.49098084, 0.50901916],
[0.24863672, 0.75136328],
[0.86558688, 0.13441312],
[0.88674299, 0.11325701],
[0.84707417, 0.15292583],
[0.34920029, 0.65079971],
[0.90863854, 0.09136146],
[0.82356149, 0.17643851],

```
       [0.81740494, 0.18259506],
       [0.63925408, 0.36074592],
       [0.89340384, 0.10659616],
       [0.86123184, 0.13876816],
       [0.80224294, 0.19775706],
       [0.48737051, 0.51262949],
       [0.64359075, 0.35640925],
       [0.82836393, 0.17163607],
       [0.40493077, 0.59506923],
       [0.88275665, 0.11724335],
       [0.29899972, 0.70100028],
       [0.45917908, 0.54082092]])
```

Get Model Evaluation

```
from sklearn.metrics import confusion_matrix, classification_report

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

```
              precision    recall  f1-score   support

           0       0.78      0.95      0.86       150
           1       0.84      0.51      0.63        81

    accuracy                           0.79       231
   macro avg       0.81      0.73      0.74       231
weighted avg       0.80      0.79      0.78       231
```

#Get Fututre Predictions

```
X_new= df.sample(1)

X_new
```

```
    pregnancies  glucose  diastolic  triceps  insulin   bmi    dpf
age  \
15            7      100          0        0        0  30.0  0.484
32

    diabetes
15         1
```

```
X_new.shape
```

```
(1, 9)
```

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

X_new
```

```
    pregnancies  glucose  diastolic  triceps  insulin   bmi    dpf
age
15            7      100          0        0        0  30.0  0.484
32
```

```
X_new.shape
```

```
(1, 8)
```

```
X_new= mn.fit_transform(X_new)
```

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

```
y_pred_new
```

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

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

```
array([[0.9918974, 0.0081026]])
```