Mini Project 6: Disease Prediction

Step 1: Import library

```
import pandas as pd
```

```
import numpy as np
```

```
import seaborn as sns
```

Step 2: Import Data

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

```
df
```

	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	bmi	dpf	age	diabetes
count	768.000000	768.000000	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	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000

```
df.shape
```

```
(768, 9)
```

```
df.columns
```

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

Auxillary Step: Get Unique Values in y

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

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

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

diabetes	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000

Step 3. Define y X

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

```
y.shape
```

```
(768,)
```

```
y
```

```
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
```

```
X=df.drop(['diabetes'],axis=1)
#or
#X=df[['pregnancies', 'glucose', 'diastolic', 'triceps', 'insulin', 'bmi','dpf', 'age']]
```

```
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 × 8 columns

Step 5: Standardising X

```
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]])
```

Step 5: Splitting Data

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

```
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.7,stratify=y,random_state=2529)
```

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

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

Step 6: Creating Model

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

Step 7: Training Model

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

```
LogisticRegression()
```

Step 8: Predicting Model

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

```
y_pred.shape
```

```
(231,)
```

```
y_pred
```

```
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])
```

Auxillary Step: Probability for Each Predicted Class

```
model.predict_proba(X_test)
```

```
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],
```

Step 9: Accuracy

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

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

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

```
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
```

Step 10: Sample Prediction

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

```
df_new
```

	pregnancies	glucose	diastolic	triceps	insulin	bmi	dpf	age	diabetes
506	0	180	90	26	90	36.5	0.314	35	1

```
df_new.shape
```

```
(1, 9)
```

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

```
X_new
```

```
          pregnancies  glucose  diastolic  triceps  insulin   bmi    dpf  age
X_new.shape
```

```
(1, 8)
```

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

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

```
y_pred_new
```

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

```
model.predict_proba(X_new)
```

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

Link of the same: https://colab.research.google.com/drive/1GP__HNbVCjYpSmbFekx9am0O1kPTH954?usp=sharing