DIABETES_PREDECTION

Diabetes, also known as diabetes mellitus, is a group of common endocrine diseases characterized by sustained high blood sugar levels.

Diabetes is caused by either a lack of insulin-secreting beta-cells in the pancreas due to an autoimmune response (type 1 diabetes), an imbalance between blood sugar level and insulin production (type 2 diabetes), and can be precipitated by pregnancy (gestational diabetes).

In this project we are going to predict whether a person has diabetes or not using logistic regression.

LOGISTIC REGRESSION

Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring. An example of logistic regression could be applying machine learning to determine if a person is likely to be infected with COVID-19 or not.

ABOUT THE DATASET

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes.

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Pregnancies: Number of times pregnant Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test BloodPressure: Diastolic blood pressure (mm Hg) SkinThickness: Triceps skin fold thickness (mm) Insulin: 2-Hour serum insulin (mu U/ml) BMI: Body mass index (weight in kg/(height in m)^2) DiabetesPedigreeFunction: Diabetes pedigree function Age: Age (years) Outcome: Class variable (0 or 1)

IMPORTING REQUIRED LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

READING DATASET

```
data=pd.read_csv("/content/diabetes.csv")
data
```

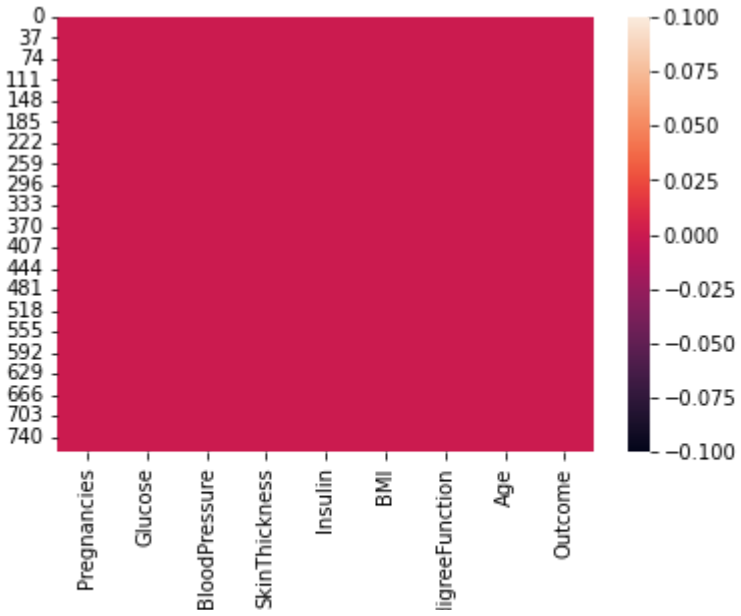
	Pregnancies	Glucose	BloodPressure	SkinT
0	6	148	72	
1	1	85	66	
2	8	183	64	
3	1	89	66	
4	0	137	40	
...	...	...	...	
763	10	101	76	
764	2	122	70	
765	5	121	72	
766	1	126	60	
767	1	93	70	

768 rows × 9 columns

Finding null values using heatmap

```
sns.heatmap(data.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at
0x7f326236b3d0>
```



From this heatmap there is no separate spaces like white lines . Therefore it has no null values .

Finding The correlation

```
correlation=data.corr()
print(correlation)
```

```
                          Pregnancies   Glucose  BloodPressure  SkinThickness
Pregnancies                  1.000000  0.129459       0.141282      -0.081672
Glucose                      0.129459  1.000000       0.152590       0.057328
BloodPressure                0.141282  0.152590       1.000000       0.207371
SkinThickness               -0.081672  0.057328       0.207371       1.000000
Insulin                     -0.073535  0.331357       0.088933       0.436783
BMI                          0.017683  0.221071       0.281805       0.392573
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265       0.183928
Age                          0.544341  0.263514       0.239528      -0.113970
Outcome                      0.221898  0.466581       0.065068       0.074752

                            Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies               -0.073535  0.017683                 -0.033523
Glucose                    0.331357  0.221071                  0.137337
BloodPressure              0.088933  0.281805                  0.041265
SkinThickness              0.436783  0.392573                  0.183928
Insulin                    1.000000  0.197859                  0.185071
BMI                        0.197859  1.000000                  0.140647
DiabetesPedigreeFunction   0.185071  0.140647                  1.000000
Age                       -0.042163  0.036242                  0.033561
Outcome                    0.130548  0.292695                  0.173844

                               Age   Outcome
Pregnancies               0.544341  0.221898
Glucose                   0.263514  0.466581
```

```
BloodPressure             0.239528  0.065068
SkinThickness            -0.113970  0.074752
Insulin                  -0.042163  0.130548
BMI                       0.036242  0.292695
DiabetesPedigreeFunction  0.033561  0.173844
Age                       1.000000  0.238356
Outcome                   0.238356  1.000000
```

From The correlation we can say that what features have direct impact on output class variable.

Dropping Skinthickness and Diabetes pedigree function because an ordinary person cannot have knowledge of these things.

```
data=data.drop(['SkinThickness','DiabetesPedigreeFunction'],axis=1)
data
```

	Pregnancies	Glucose	BloodPressure	Insul
0	6	148	72	
1	1	85	66	
2	8	183	64	
3	1	89	66	
4	0	137	40	1
...	...	...	...	
763	10	101	76	1
764	2	122	70	
765	5	121	72	1
766	1	126	60	
767	1	93	70	

768 rows × 7 columns

Test and Training data

We take X as features such as drop outcome and take remaining as features. Y dataset is the outcome class variable Which determines the person has diabetes or not. We take 20% of data for training.

1-person has diabetes.

0-person has no diabetes.

```
X=data.drop("Outcome", axis=1)
Y=data["Outcome"]


X_train, X_test, Y_train, Y_test=train_test_split(X, Y, test_size=0.2)

X_train
```

	Pregnancies	Glucose	BloodPressure	Insul
674	8	91	82	
536	0	105	90	
589	0	73	0	
516	9	145	88	1
550	1	116	70	
...	...	...	...	
518	13	76	60	
77	5	95	72	
267	2	128	64	
759	6	190	92	
145	0	102	75	

614 rows × 6 columns

predicting the output :

```
predict_model=LogisticRegression ()
predict_model.fit(X_train, Y_train)
```

```
LogisticRegression()
```

```
prediction_results=predict_model.predict(X_test)
print(prediction_results)
```

```
[0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 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 1 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1
 1 1 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 1 0 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0
 0 1 0 0 0 0]
```

ACCURACY :

In machine learning, accuracy is one of the most important performance evaluation metrics for a classification model. The mathematical formula for calculating the accuracy of a machine learning model is

1-(Number of misclassified samples / Total number of samples)

Good accuracy in machine learning is subjective. But in our opinion, anything greater than 70% is a great model performance. In fact, an accuracy measure of anything between 70%-90% is not only ideal, it's realistic. This is also consistent with industry standards.

```
Accuracy=accuracy_score(prediction_results, Y_test)
print(Accuracy*100,"%")
```

```
     81.16883116883116 %
```

After we trained our model we get a total accuracy of 0.8116 which is 81%

By now we check our model by giving some input to the model that we created. After get input we take the input which is converted into a numpy array.

```
print("Pregnancy,Glucose,Blood_pressure,Insulin,BMI,Age")
n=6


input_data=[]
for i in range(0, n):

    ele = float(input())



    input_data.append(ele) # adding the element



print(input_data)
user_input=np.asarray(input_data)
User_data=user_input.reshape(1, -1)
prediction=predict_model.predict(User_data)

if (prediction [0]==0):
   print("The person has no diabetes")
```

```
else:
   print("The person has diabetes")
```

```
Pregnancy,Glucose,Blood_pressure,Insulin,BMI,Age
0
105
90
0
29.6
46
[0.0, 105.0, 90.0, 0.0, 29.6, 46.0]
The person has no diabetes
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does
  warnings.warn(
```

I checked my model which give correct value for the input .

Thank you

If any suggestion please message me.