```
"""
Developer : Saran B ( 2205102 )
Date : 14/12/2024
Topic : Diabetes Prediction Model
Data Source : https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database
Credits : Lex Fridman, Jeremy Howards, Siddharthan

"""

import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
import pickle
```

Providing Data (diabetes.csv)

```
# Importing the data
DiabetesData_1 = pd.read_csv('/content/diabetes.csv')
# Check the Number of rows and columns in the data set
DiabetesData_1.shape
```

```
(768, 9)
```

We have 768 rows and 9 columns. to check the column name..

```
#displaying the 5 rows of the data
DiabetesData_1.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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

Checking the basic Statistics of the Data using description()

```
DiabetesData_1.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Ou
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.(
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.:
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.4
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.(
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.(
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.(
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.(
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.(

```
DiabetesData_1.mean()
```

```
                                      0
               Pregnancies     3.845052
                   Glucose   120.894531
             BloodPressure    69.105469
             SkinThickness    20.536458
                   Insulin    79.799479
                       BMI    31.992578
  DiabetesPedigreeFunction     0.471876
                       Age    33.240885
                   Outcome     0.348958

dtype: float64
```

Grouping the Outcome and getting mean value of those will give us a Idea that how the Prediction is going to be.

Here the 0 is the " Non Diabetic people " and the 1 is the " Diabetic people ".

```
DiabetesData_1.groupby("Outcome").mean()
```

```
         Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin        BMI  DiabetesPedigreeFunction        Age
Outcome
      0     3.298000  109.980000      68.184000      19.664000   68.792000  30.304200                  0.429734  31.190000
      1     4.865672  141.257463      70.824627      22.164179  100.335821  35.142537                  0.550500  37.067164
```

Dividing the data set into features and labesl and store them into the variables "Features" and "Label" respectively

```
Features = DiabetesData_1.drop(columns='Outcome', axis=1)
Label = DiabetesData_1['Outcome']
```

```
Features.head()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  DiabetesPedigreeFunction  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
```

```
Label.head()
```

```
   Outcome
0        1
1        0
2        1
3        0
4        1

dtype: int64
```

Now lets do standardize the data. In the next few steps , we are going to 1). Fit the data 2). Standardize (transform the data into a small range) these steps will make our model to predict high

```
Stool = StandardScaler()
```

```
Stool.fit_transform(Features)
```

```
array([[ 0.63994726,  0.84832379,  0.14964075, ...,  0.20401277,
         0.46849198,  1.4259954 ],
       [-0.84488505, -1.12339636, -0.16054575, ..., -0.68442195,
        -0.36506078, -0.19067191],
       [ 1.23388019,  1.94372388, -0.26394125, ..., -1.10325546,
         0.60439732, -0.10558415],
       ...,
       [ 0.3429808 ,  0.00330087,  0.14964075, ..., -0.73518964,
        -0.68519336, -0.27575966],
       [-0.84488505,  0.1597866 , -0.47073225, ..., -0.24020459,
        -0.37110101,  1.17073215],
       [-0.84488505, -0.8730192 ,  0.04624525, ..., -0.20212881,
        -0.47378505, -0.87137393]])
```

```
Standarized_Features = Stool.fit_transform(Features)
```

```
Features = Standarized_Features
Label = Label
```

Now all the basic data standardization is completed, Let's move to the Training section.

before that, we need to spllit the data into training portion and testing portion. Here is the steps..

```
Features_train, Features_test, Label_train, Label_test = train_test_split(Features, Label, test_size=0.2, stratify=Label, ra
```

```
print(Features.shape, Features_train.shape, Features_test.shape)
```

```
(768, 8) (614, 8) (154, 8)
```

```
print(Label.shape, Label_train.shape, Label_test.shape)
```

```
(768,) (614,) (154,)
```

The data splitted successfully, now get into the training session

```
Classifier = svm.SVC(kernel='linear')
```

```
Classifier.fit(Features_train, Label_train)
```

```
▾ SVC ⓘ ?
SVC(kernel='linear')
```

Hello shiva sir, since we have a small data set. the training wont take much time to complete

```
#Testing and evaluvating
Features_train_prediction = Classifier.predict(Features_train)
Training_data_accuracy = accuracy_score(Features_train_prediction, Label_train)
print("Traning data accuracy is", Training_data_accuracy)
```

```
Traning data accuracy is 0.7866449511400652
```

```
Features_test_prediction = Classifier.predict(Features_test)
Testing_data_accuracy = accuracy_score(Features_test_prediction, Label_test)
print("Testing data accuracy is", Testing_data_accuracy)
```

```
Testing data accuracy is 0.7727272727272727
```

We evaluated our modal and that is performing good after some oprimization in data.

Now lets build a predictive system by user's input

```
input_data = (2,112,66,22,0,25,0.307,24)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

Stooled_data = Stool.transform(input_data_reshaped)
print(Stooled_data)

prediction = Classifier.predict(Stooled_data)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

```
[[-0.54791859 -0.27837344 -0.16054575  0.09180513 -0.69289057 -0.88749274
  -0.497946   -0.78628618]]
[0]
The person is not diabetic
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but Stand
  warnings.warn(
```

Saving the Model

```
file_name = "trained_model.sav"
pickle.dump(Classifier, open(file_name,"wb"))
```

```
loaded_model = pickle.load(open("trained_model.sav","rb"))
```

```
input_data = (2,112,66,22,0,25,0.307,24)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

Stooled_data = Stool.transform(input_data_reshaped)
print(Stooled_data)

prediction = loaded_model.predict(Stooled_data)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

```
[[-0.54791859 -0.27837344 -0.16054575  0.09180513 -0.69289057 -0.88749274
  -0.497946   -0.78628618]]
[0]
The person is not diabetic
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but Stand
  warnings.warn(
```