Diabetes Prediction using Machine Learning

July 7, 2023

1 Importing the Dependencies

[1]:

```
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
```

2 Data Collection and Analysis

[2]:

```
df = pd.read_csv('diabetes.csv')
```

[3]:

```
df
```

[3]:

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …       …              …              …       …   …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        …  …      …
763                     0.171   63        0
```

```
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

[4]:
```
#printing first 5 rows of the dataset
df.head()
```

[4]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

[5]:
```
# no. of rows and columns in this dataset
df.shape
```

[5]:
```
(768, 9)
```

[6]:
```
#Getting the statistical measures of the data
df.describe()
```

[6]:
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     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  DiabetesPedigreeFunction         Age     Outcome
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['Outcome'].value_counts()
```

```
[7]: 0    500
     1    268
     Name: Outcome, dtype: int64
```

0 <- Non diabetic 1 <- Diabetic

```
[8]: df.groupby('Outcome').mean()
```

```
[8]:          Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
     Outcome
     0           3.298000  109.980000      68.184000      19.664000   68.792000
     1           4.865672  141.257463      70.824627      22.164179  100.335821

                    BMI  DiabetesPedigreeFunction        Age
     Outcome
     0        30.304200                  0.429734  31.190000
     1        35.142537                  0.550500  37.067164
```

```
[9]: # seperating data and labels
     x = df.drop(columns = 'Outcome', axis= 1)
     y= df['Outcome']
```

```
[10]: x
```

```
[10]:      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
      0              6      148             72             35        0  33.6
      1              1       85             66             29        0  26.6
      2              8      183             64              0        0  23.3
      3              1       89             66             23       94  28.1
      4              0      137             40             35      168  43.1
      ..           …        …             …              …        …     …
      763           10      101             76             48      180  32.9
      764            2      122             70             27        0  36.8
      765            5      121             72             23      112  26.2
      766            1      126             60              0        0  30.1
      767            1       93             70             31        0  30.4

           DiabetesPedigreeFunction  Age
      0                       0.627   50
      1                       0.351   31
```

```
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        …  …
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
```

[11]:
```
y
```

[11]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

3 Data Standardization

[12]:
```
scaler = StandardScaler()
```

[13]:
```
scaler.fit(x)
```

[13]:
```
StandardScaler()
```

[15]:
```
standardized_data = scaler.transform(x)
```

[16]:
```
print(standardized_data)
```

```
[[ 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]]
```

[17]:

```
x = standardized_data
y = df['Outcome']
```

[18]:

```
x
```

[18]:

```
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]])
```

[19]:

```
y
```

[19]:

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

4 Train Test Split

[20]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.
 ↪2,stratify = y, random_state=2)
```

[21]:
```
print(x.shape, x_train.shape, x_test.shape)
```

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

5 Training the model

Support Vector Machine

[23]:
```
classifier = svm.SVC(kernel = 'linear')
```

[24]:
```
#training the support vector machine classifier
classifier.fit(x_train, y_train)
```

[24]:
```
SVC(kernel='linear')
```

6 Model Evaluation

Accuracy score

[25]:
```
# accuracy score on the training data
x_train_prediction = classifier.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction, y_train)
```

[26]:
```
print('Accuracy score of the training data :' , training_data_accuracy)
```

```
Accuracy score of the training data : 0.7866449511400652
```

[27]:
```
# accuracy score on the test data
x_test_prediction = classifier.predict(x_test)
test_data_accuracy = accuracy_score(x_test_prediction, y_test)
```

[28]:
```
print('Accuracy score of the test data :' , test_data_accuracy)
```

```
Accuracy score of the test data : 0.7727272727272727
```

7 Building a predictive system

[30]:
```
input_data = (4,110,92,0,0,37.6,0.191,30)
```

[32]:
```
input_data = (4,110,92,0,0,37.6,0.191,30)
# Changing the input data into numpy array
input_data_as_numpy_array = np.asarray(input_data)

#reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
```

```python
#standardize the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)

prediction = classifier.predict(std_data)
print(prediction)


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

```
[[ 0.04601433 -0.34096773  1.18359575 -1.28821221 -0.69289057  0.71168975
  -0.84827977 -0.27575966]]
[0]
The person is not diabetic
```

```
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but StandardScaler was fitted with feature
names
  warnings.warn(
```