Name: Nishendu Mishra

Topic: Diabetes Detection System

Email: nishendumishra73@gmail.com

```
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
```

Data Collection and Analysis

PIMA Diabetes Dataset

```
#loading the diabetes dataset to a pandas dataframe
diabetes_dataset = pd.read_csv('/content/diabetes.csv')
```

run pd.read_csv? to know about the function of read_csv

```
diabetes_dataset.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
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	

```
diabetes_dataset.shape
```

```
(768, 9)
```

```
diabetes_dataset.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
count	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.
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.

```
diabetes_dataset['Outcome'].value_counts()
```

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

```
diabetes_dataset.groupby('Outcome').mean()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeF
Outcome							
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	(
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	(

so we can see diabetic patients hav high glucose level in them and they have more average age also

```
x = diabetes_dataset.drop(columns='Outcome', axis=1)
y = diabetes_dataset['Outcome']
```

Data Standardization

```
scaler = StandardScaler()
```

```
standardized_data= scaler.fit_transform(x)
```

```
x= standardized_data
```

```
print(x)
```

```
[[ 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]]
```

```
print(y)
```

```
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
```

```
print(x.std())
```

```
1.0
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,stratify=y, random_state=2)
```

```
print(x.shape, x_train.shape, x_test.shape)
```

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

Training the Model

SVC represents support vector classifier

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

```
#training the support vector machine classifier
classifier.fit(x_train, y_train)
```

```
SVC
SVC(kernel='linear')
```

Evaluation of the Model

Accuracy Score

```
#accuracy score of the training data
x_train_prediction = classifier.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction, y_train)


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

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

```
#accuracy score of the testing data
x_test_prediction = classifier.predict(x_test)
testing_data_accuracy = accuracy_score(x_test_prediction, y_test)


print('Accuracy score of the testing data: ', testing_data_accuracy)
```

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

Making a predictive System

```
input_data = (3,158,76,36,245,31.6,0.851,28)
#change the input data to a numpy array

input_data_as_numpy_array = np.asarray(input_data)

#reshape the array as we are predicting for one instance
#because our model has been trained on 768 data values , so to remove any confusion we are reshaping the data

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# 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 Non Diabetic')
else:
  print('The person is Diabetic')
```

```
[[-0.25095213  1.16129525  0.35643175  0.96999799  1.43441893 -0.04982572
   1.14499856 -0.44593516]]
[1]
The person is Diabetic
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler w
  warnings.warn(
```