Importing the dependencies

```
import pandas as pd
import numpy as np
```

```
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 dataset
diabetes_dataset = pd.read_csv('/content/Diabetes.csv')
```

```
# Printng the first five values of the dataset
diabetes_dataset.head()
```

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

Next steps: Generate code with diabetes_dataset View recommended plots New interactive sheet

```
# number of rows and columns in this dataset
diabetes_dataset.shape
```

```
(768, 9)
```

```
# Getting the statistical measures of the data
diabetes_dataset.describe()
```

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

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

	count
Outcome	
0	500
1	268

dtype: int64

0 -->Non Diabetic 1-->Diabetic

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
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	

```
# Seperating the data and labels
X = diabetes_dataset.drop(columns='Outcome',axis=1)
Y = diabetes_dataset['Outcome']
```

```
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)
```

```
     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]
```

Data Standardization

```
scaler = StandardScaler()
```

```
scaler.fit(X)
```

```
▾ StandardScaler
StandardScaler()
```

```
standardized_data = scaler.transform(X)
```

```
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]]
```

```
X = standardized_data
Y = diabetes_dataset['Outcome']
```

Train Test Split

```
X_train , X_test , Y_train , Y_test = train_test_split(X,Y,test_size=0.2,stratify=Y,random_sta
```

```
print(X.shape , X_train.shape , X_test.shape)
```

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

Training the model

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

```
# Training the SVM
classifier.fit(X_train,Y_train)
```

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

Model Evaluation

```
# Accuracy Score
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 on test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction,Y_test)
```

```
print('Accuracy Score of the Test Data :', test_data_accuracy)
```

```
Accuracy Score of the Test Data : 0.7727272727272727
```

Making a predictive system

```
import numpy as np
import pandas as pd

# Input data
input_data = (2, 108, 62, 32, 56, 25.2, 0.128, 21)

# Column names used when fitting the scaler
column_names = ['Pregnancies', 'Glucose', 'BloodPressure',  'SkinThickness',  'Insulin' ,  'BM

# Convert the input data into a DataFrame with the appropriate column names
input_data_df = pd.DataFrame([input_data], columns=column_names)

# Standardize the data using the fitted scaler
std_data = scaler.transform(input_data_df)
print(std_data)

# Make a prediction
prediction = classifier.predict(std_data)
print(prediction)
```

```
[[-0.54791859 -0.40356202 -0.36733675  0.71908574 -0.2066484  -0.86210889
  -1.03854724 -1.04154944]]
[0]
```

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

```
The person is not diabetic
```

Start coding or generate with AI.