```
#Separating the data and the labels
from sklearn.model_selection import train_test_split

X = data.drop(['Outcome'], axis =1)
y = data['Outcome']
```

```
#Data Standardization
scaler = StandardScaler()
```

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

```
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 =data['Outcome']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify =y, random_state = 2)
```

```
#Training the Model
classifier = svm.SVC(kernel='linear')
```

```
#training the SVM
classifier.fit(X_train, y_train)
```

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

```
#Model Evaluation
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 the 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
```

```python
#Making a predictive System
#Using an example from a dataset
input_data = (
    int(input("Enter number of Pregnancies (0 to 17): ")),
    float(input("Enter Glucose level (0 to 199): ")),
    float(input("Enter Blood Pressure (0 to 122): ")),
    float(input("Enter Skin Thickness (0 to 99): ")),
    float(input("Enter Insulin level (0 to 846): ")),
    float(input("Enter BMI (0 to 67.1): ")),
    float(input("Enter Diabetes Pedigree Function (0.078 to 2.42): ")),
    int(input("Enter Age (21 to 81): "))
)

#Changing the input data to numy 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)

#Standardise 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 Patient is possibly Not Diabetic')
else:
    print('The Patient is possibly Diabetic')
```

```
Enter number of Pregnancies (0 to 17): 4
Enter Glucose level (0 to 199): 110

Enter Blood Pressure (0 to 122):
```

```
Enter number of Pregnancies (0 to 17): 4
Enter Glucose level (0 to 199): 110
Enter Blood Pressure (0 to 122): 92
Enter Skin Thickness (0 to 99): 0
Enter Insulin level (0 to 846): 0
Enter BMI (0 to 67.1): 37.6
Enter Diabetes Pedigree Function (0.078 to 2.42): 0.191
Enter Age (21 to 81): 30
[[ 0.04601433 -0.34096773  1.18359575 -1.28821221 -0.69289057  0.71168975
  -0.84827977 -0.27575966]]
[0]
The Patient is possibly Not Diabetic
```

	A	B	C	D	E	F	G	H	I
1	Pregnancie	Glucose	BloodPres	SkinThickn	Insulin	BMI	DiabetesP	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1
9	10	115	0	0	0	35.3	0.134	29	0
10	2	197	70	45	543	30.5	0.158	53	1
11	8	125	96	0	0	0	0.232	54	1
12	4	110	92	0	0	37.6	0.191	30	0