```
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    #support vector machine
from sklearn.metrics import accuracy_score
```

Data collection

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

```
diabetes_dataset.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeF
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)
```

```
#getting the statistical measure of the data
diabetes_dataset.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Di
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	

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

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

0 --> Non-Diabetic

1 -->Diabetic

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

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	D
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 label
x=diabetes_dataset.drop(columns='Outcome',axis=1)  #axis =1 show-- you are using the drop fu
y=diabetes_dataset['Outcome']
```

```
x
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	

```
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
```

767	1	93	70	31	0	30.4	

Data standardization

```
scaler=StandardScaler()
```

standardized_data=scaler.fit_transform(x)

```
scaler.fit(x)
```

```
StandardScaler()
```

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

```
standardized_data
```

```
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]])
```

```
x=standardized_data
```

```
x
```

```
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]])
```

```
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
```

Train Test Split

we split x data into two part x_train n x_test. and same for y.

we feed machine with x_train Y_train and after machine make some algorithm, we will check with x_test and y_test data.

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

provide the old data set so (x,y)

test_size=0.2 mean you are assigning 20% data of x as test data.

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

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

```
print(y.shape,y_train.shape,y_test.shape)
```

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

Training the model

```
classifier=svm.SVC(kernel='linear')  #svm=support vector machine, svc=support vector classifi
```

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

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

by above line of code the machine is trained.

Model Evaluation

Accuracy Score on the basis of training data.

```
x_train_prediction=classifier.predict(x_train)
training_data_accuracy=accuracy_score(x_train_prediction,y_train)
```

```
training_data_accuracy
```

```
0.7866449511400652
```

our model is 78% accurate. on the basis of training data..

Accuracy Score on the basis of test data

```
x_test_prediction=classifier.predict(x_test)
test_data_accuracy=accuracy_score(x_test_prediction,y_test)
```

```
test_data_accuracy
```

```
0.7727272727272727
```

Making a predictive system

```
#provide a input from diabites data set. and outcome must be 0.
input_data=(4,110,90,0,0,37.6,0.191,30)

#changing the input_data to numpy array
input_data_as_numpy_array=np.asarray(input_data)

#our model is trained on 768 point and 8 column, not for 1 intance, so reshape the input so t
#can understand
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1) #pass perameter as (1,-1) as one

#we have trained our machine on standarized data(on some range).so convert user input as star

#standarize the input data
std_data=scaler.transform(input_data_reshaped)
print(std_data)

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

#This all the extra stuff.
if prediction[0]==0:print("Person is non-diabetes")
else:print("Person is Diabetes")
```

```
[[ 0.04601433 -0.34096773  1.08020025 -1.28821221 -0.69289057  0.71168975
  -0.84827977 -0.27575966]]
[0]
Person is non-diabetes
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not hav
  "X does not have valid feature names, but"
```

✓ 0s completed at 11:23 AM