```
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
```

imported numerical python as np, pandas as pd. standard scalar for standardizing the data. and to train the model we use sklearn and import train test split. for the accuracy output we use sklearn.metrics (accuracy score parameter)

PIMA Diabetes Dataset

data collected from 768 women by National Institute of Diabetes and Digestive and Kidney Diseases.

```
#load the data
data = pd.read_csv('/content/diabetesdata.csv')
```

```
data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Out
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	

```
data.shape
```

```
(768, 9)
```

```
#we found the total rows and columns
```

```
data.columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

We find out all the parameters and outcome(whether she is diabetic or not)

```
data.isnull().values.any()
```

```
False
```

nto find null values

```
data.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.

We found out the statistical values of this data set.

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

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

0 == Non Diabetic 1 == Diabetic

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

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

```
X=data.drop(columns='Outcome',axis=0)
y=data['Outcome']
```

```
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
```

Standardizing the data

we need to standardize the data. because if we directly took the data and train it. It will be biased towards Non diabetic as we found out no diabetic women are more than diabetic women.

```
scalar = StandardScaler()
```

```
scalar.fit(X)
```

```
StandardScaler()
```

```
standardized_data = scalar.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']
```

```
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
```

Train test Split

training the data set with 30% of given data.

```
X_train,X_test,Y_train,Y_test = train_test_split(X,y,test_size=0.3,stratify=y,random_state= 2)
```

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

```
     (768, 8) (537, 8) (231, 8)
```

Training the model

We use Support Vector Machine since it is accurate from other classifications.

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

```
classifier.fit(X_train,Y_train)
```

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

Accuracy Score

```
X_train_prediction = classifier.predict(X_train)
training_data_accuracy =accuracy_score(X_train_prediction,Y_train)
```

```
print("Accuracy Score of training data is",training_data_accuracy)
```

```
Accuracy Score of training data is 0.7821229050279329
```

```
X_test_prediction = classifier.predict(X_test)
test_data_accuracy =accuracy_score(X_test_prediction,Y_test)
```

```
print("Accuracy Score of test data is",test_data_accuracy)
```

```
Accuracy Score of test data is 0.7748917748917749
```

Prediction System

We test a data from input source to confirm the algorithm.

```
input = (4,110,92,0,0,37.6,0.191,30)
#convert input to numpy

input_as_numpy_array = np.asarray(input)

#reshape the array data to predict for one instance

input_reshape  = input_as_numpy_array.reshape(1,-1)

#standadise the input like training data
std_data = scalar.transform(input_reshape)
print(std_data)

#prediction

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

if (prediction[0]==0):
  print("The person is Non-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 Non-diabetic
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but StandardScaler wa

  warnings.warn(
```