Importing dependencies :

```
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 Analysis :

```
#loading the data
dataset = pd.read_csv('/content/diabetes.csv')
```

```
dataset.head()
```

	PG	GL	BP	ST	INS	BMI	DPF	AGE	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
# no. of rows and column in dataset
dataset.shape
```

```
(768, 9)
```

```
# stats. data
dataset.describe()
```

	PG	GL	BP	ST	INS	BMI	DPF	AGE	Outcome
count	768.000000	768.000000	768.000000	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	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
dataset['Outcome'].value_counts()
# 0 --> Non diabetic
# 1 --> Diabetic
```

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

```
dataset.groupby('Outcome').mean()
# mean value of the inputs for a particular coutcomes
```

Outcome	PG	GL	BP	ST	INS	BMI	DPF	AGE
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

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

```
print(X)
```

```
          PG   GL  BP  ST  INS   BMI    DPF  AGE
     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
     ..   ..  ...  ..  ..  ...   ...    ...  ...
     763  10  101  76  48  180  32.9  0.171   63
     764   2  122  70  27    0  36.8  0.340   27
     765   5  121  72  23  112  26.2  0.245   30
     766   1  126  60   0    0  30.1  0.349   47
     767   1   93  70  31    0  30.4  0.315   23

     [768 rows x 8 columns]
```

```
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
```

Data Standardization

```
scaler = StandardScaler()
```

```
xs = scaler.fit_transform(X)
X = xs
```

```
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]]
```

Splitting Data :

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, stratify = Y, random_state = 42)
```

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

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

Training Model :

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

```
classifier.fit(x_train, y_train)
```

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

Model Evaluation :

Accuracy Score :

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

```
print('Accuracy Score for SVC on training data is : ', train_accuracy)
```

```
Accuracy Score for SVC on training data is :  0.7915309446254072
```

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

```
print('Accuracy Score for SVC on testing data is : ', test_accuracy)
```

```
Accuracy Score for SVC on testing data is :  0.7207792207792207
```

Making Predictive Model :

```
input_data = (8, 183, 64, 0, 0, 23.3, 0.672, 32)

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

#reshaping
input_reshaped = input_numpyarray.reshape(1, -1)

#standardizing
inp = scaler.transform(input_reshaped)
print(inp)

#predicting
out = classifier.predict(inp)
print(out)

#printing result
if(out[0] == 0) :
  print('non Diabetic')
else :
  print('Diabetic')
```

```
[[ 1.23388019  1.94372388 -0.26394125 -1.28821221 -0.69289057 -1.10325546
   0.60439732 -0.10558415]]
[1]
Diabetic
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fi
  warnings.warn(
```

```
import pickle
```

```
filename = 'diabetes_model.sav'
pickle.dump(classifier, open(filename, 'wb'))
```