```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler#used to standardize the data into a common range
from sklearn.model_selection import train_test_split #used to split the data into training and test data
from sklearn import svm
from sklearn.metrics import accuracy_score
```

importing the dependencies

Data collection and analysis

```
#loading the dataset to a pandas dataset
diabetes_dataset=pd.read_csv('/content/diabetes.csv')
```

```
diabetes_dataset.shape # finding number of rows and columns
```

```
(768, 9)
```

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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

```
# to see how many values for dibetes (1) and how many for non diabetes(0)
diabetes_dataset['Outcome'].value_counts()
```

Outcome	count
0	500
1	268

dtype: int64

```
diabetes_dataset.groupby('Outcome').mean() # we have grouped the data on the basis of their labels
```

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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

by looking the above table following observations can be made i.e for example the person who is diabetic have higher glucose level then person whi is non diabetic by looking at their mean values

Separating the data and labels

```
X=diabetes_dataset.drop(columns='Outcome',axis=1)
Y=diabetes_dataset['Outcome']
```

```
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) # transforming the data in common range
```

```
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]]
```

By looking above,we can see now,all the values are in range of 0's and 1's, this will help our model to make better predictions

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

now we have all the data in 'X' and all the labels in 'Y'.

```
print(X)
print(Y)
```

```
[[ 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]]
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
```

Split the data into training data and test data

```
X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
# X_train denotes the train data
# X_test denotes the test data
# y_train and Y_test are the labels of train and test data respectively
```

the number of record in test data and train data are:-

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

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

Training the model

```
classifier=svm.SVC(kernel='linear') # to load the svm model
```

```
#training the support vector Machine Classifier
classifier.fit(X_train,Y_train)
```

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

Model Evaluation

Accuracy Score

```
# accuracy score on the training data
X_train_prediction=classifier.predict(X_train) # all the predictions are  stored in X_train_prediction
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)# compairing of the predicted data with the original labels
```

```
print('Accuracy score of training data', training_data_accuracy)
```

```
Accuracy score of training data 0.7866449511400652
```

finding accuracy score of test data

```
# accuracy score on the test data
X_test_prediction=classifier.predict(X_test) # all the predictions are  stored in X_test_prediction
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)# compairing of the predicted data with the original labels
```

```
print(test_data_accuracy)
```

```
0.7727272727272727
```

Making a Predictive System

```
input_data=(6,148,72,35,0,33.6,0.627,50)
# changing the input data to numpy array
input_data_as_numpy_array=np.asarray(input_data)
# reshape the array as we are predicting for one instance, we need this because our model is trained in 768 examples and there are 8 columns,I
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)# this will tell the model that we are giving only1 istance
# we have to standardized 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 person is not diabetic")
else:
  print("the person is diabetic")
```

```
[[ 0.63994726  0.84832379  0.14964075  0.90726993 -0.69289057  0.20401277
   0.46849198  1.4259954 ]]
[1]
the person is diabetic
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Standarc
  warnings.warn(
```