Workflow

Diabetes Data -> Data Processing -> Train Test split -> Support Vector machine Classifier -> Trained support vector machine classifer.

In [2]:

```
#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
```

In [3]:

```
#Data collection & Analysis
#Diabetes data from Pima Indians Diabetes Database

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

In [21]:

```
pd.read_csv?
```

In [33]:

```
diabetes_dataset.head()
```

Out[33]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPe
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	

In [34]:

```
diabetes_dataset.shape
```

Out[34]:

```
(768, 9)
```

In [35]:
```
diabetes_dataset.describe()
```

Out[35]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.00
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.9
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.8
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.3
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.0
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.6
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.1

In [4]:
```
diabetes_dataset['Outcome'].value_counts()
```

Out[4]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [5]:
```
diabetes_dataset.groupby('Outcome').mean()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
Outcome						
0	3.298000	109.980000	68.184000	19.664000	68.792000	30
1	4.865672	141.257463	70.824627	22.164179	100.335821	35

In [6]:
```
#Separating the data & labels
X = diabetes_dataset.drop(columns='Outcome', axis=1)
Y = diabetes_dataset['Outcome']
```

In [13]:
```
#Data Standardization
scaler = StandardScaler()
scaler.fit(X)
standardize_data = scaler.transform(X)
print(standardize_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]]
```

In [8]:
```
#Train test split

X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=
print(X.shape,X_train.shape,X_test.shape)
```

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

In [9]:
```
#Training the model

classifier = svm.SVC(kernel='linear')
#Training the support vector machine classifier
classifier.fit(X_train,Y_train)
```

Out[9]:
```
SVC(kernel='linear')
```

In [10]:
```
#Model Evaluation, Accuracy score
#Accuracy score on the training data

X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_trai
print('Accuracy score of the training data: ', training_data_accura
```

```
Accuracy score of the training data:  0.7833876221498371
```

In [11]:
```
#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
```

In [20]:

```
#Making a Predictive System
Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,BMI,Diabete

input_data = (Pregnancies,Glucose,BloodPressure,SkinThickness,Insul
#changing input to numpy array
input_data_as_numpy_array = np.asarray(input_data)

#reshaping the array as we are predicting for one instance.
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

#standardize the input data
std_data = scaler.transform(input_data_reshaped)
prediction = classifier.predict(std_data)

if (prediction[0]==0):
    print('The person is not diabetic')
else:
    print('The person is diabetic')
```

```
Please enter Pregnancies,Glucose,BloodPressure,SkinThickness,Insu
lin,BMI,DiabetesPedigreeFunction,Age seperated by commas :
3,83,58,31,18,34.3,0.336,2
The person is not diabetic
```

In [ ]:

In [ ]: