In [1]:
```
import numpy as np
import pandas as pd
from sklearn import svm
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [2]:
```
diabetes_data=pd.read_csv(r"C:\Users\HP\AppData\Local\Programs\Python\Python31
```

In [3]:
```
diabetes_data
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

768 rows × 9 columns

In [4]:
```
diabetes_data.shape
```

Out[4]:
```
(768, 9)
```

In [5]:
```
diabetes_data.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [6]:
```
diabetes_data["Outcome"].value_counts()
```

Out[6]:
```
Outcome
0    500
1    268
Name: count, dtype: int64
```

In [7]:
```
diabetes_data.groupby("Outcome").mean()
```

Out[7]:

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabe
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	

In [8]:
```
x=diabetes_data.drop("Outcome",axis=1)
y=diabetes_data["Outcome"]
```

In [9]:
```
scaler=StandardScaler()
```

In [10]:
```
scaler.fit(x)
```

Out[10]:
```
StandardScaler()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [11]:
```
standardized_data=scaler.transform(x)
```

In [12]:
```
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]]
```

In [13]:
```
x=standardized_data
```

In [14]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,stratify=y,ra
```

In [15]:
```
classifier=svm.SVC(kernel="linear")
```

In [16]:
```
classifier.fit(x_train,y_train)
```

Out[16]:
```
SVC(kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [19]:
```
training_data_prediction=classifier.predict(x_train)
training_data_accuracy=accuracy_score(training_data_prediction,y_train)
```

In [20]:
```
print(training_data_accuracy)
```

```
0.7866449511400652
```

In [21]:
```
test_data_prediction=classifier.predict(x_test)
test_data_accuracy=accuracy_score(test_data_prediction,y_test)
```

In [22]:
```
print(test_data_accuracy)
```

```
0.7727272727272727
```

In [26]:
```
input_data=(4,110,92,0,0,37.6,0.191,30)
input_data_as_array=np.asarray(input_data)
reshaped_data=input_data_as_array.reshape(1,-1)
std_data=scaler.transform(reshaped_data)
prediction=classifier.predict(std_data)
print(prediction)
```

```
[0]

C:\Users\HP\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\base.py:464: UserWarning: X does not have valid feature names, but StandardS
caler was fitted with feature names
  warnings.warn(
```

In [ ]:
```
```