In [1]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

In [2]:

```
df = pd.read_csv('diabetes.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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

In [4]:

```
df.shape
```

Out[4]:

```
(768, 9)
```

In [5]:

```
df.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	C
count	768.000000	768.000000	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	0.471876	33.240885	(
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	(
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	(
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	(
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	(
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1

In [7]:

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

Out[7]:

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

0 - non diabetic

1 - diabetic

In [8]:

```
X = df.drop(columns='Outcome',axis=1)
y = df['Outcome']
```

In [9]:

```
X
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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 × 8 columns

In [10]:

```
y
```

Out[10]:

```
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
```

In [11]:

```
scaler = StandardScaler()
```

In [12]:

```
scaler.fit(X)
```

Out[12]:

```
StandardScaler()
```

In [13]:

```
standardized_data = scaler.transform(X)
```

In [14]:

```
standardized_data
```

Out[14]:

```
array([[ 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 [20]:

```
X = standardized_data
```

In [21]:

```
X
```

Out[21]:

```
array([[ 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 [22]:

```
y
```

```
Out[22]: 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
```

```
In [23]: X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
```

```
In [24]: X_train.shape
```

```
Out[24]: (614, 8)
```

```
In [25]: X_test.shape
```

```
Out[25]: (154, 8)
```

Train the model

```
In [26]: clf = svm.SVC(kernel='linear')
```

```
In [27]: clf.fit(X_train,y_train)
```

```
Out[27]: SVC(kernel='linear')
```

```
In [28]: X_train_prediction = clf.predict(X_train)
         accuracy_score(X_train_prediction,y_train)
```

```
Out[28]: 0.7850162866449512
```

Accuracy on test data

```
In [29]: X_test_prediction = clf.predict(X_test)
         accuracy_score(X_test_prediction,y_test)
```

```
Out[29]: 0.7727272727272727
```

```
In [30]: input_sample = (5,166,72,19,175,22.7,0.6,51)
```

```
In [31]: input_np_array = np.asarray(input_sample)
```

```
In [32]: input_np_array_reshaped = input_np_array.reshape(1,-1)
```

```
In [33]: std_data = scaler.transform(input_np_array_reshaped)
```

```
c:\users\admin\appdata\local\programs\python\python39\lib\site-packages\sklearn\base.py:450: UserWarning: X does
not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

```
In [34]: std_data
```

```
Out[34]: array([[ 0.3429808 ,  1.41167241,  0.14964075, -0.09637905,  0.82661621,
                 -1.179407  ,  0.38694877,  1.51108316]])
```

```
In [38]: prediction = clf.predict(std_data)
```

```
In [39]: prediction
```

```
Out[39]: array([1], dtype=int64)
```

```
In [41]: if (prediction[0]==0):
             print("Person is not diabetic")
         else:
             print("Person is diabetic")
```

```
Person is diabetic
```

```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js