```
import numpy as np
import pandas as pd
df=pd.read_csv("/content/diabetes.csv")
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
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	

768 rows × 9 columns

```
print(df.columns)
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insuli
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

```
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabet
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	

```
df.tail()
```

✓ 1s completed at 3:10 PM

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

```
df.shape
```

```
(768, 9)
```

```
print(df.isna().sum())
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
x=df.iloc[:,:-1].values
x
```

```
array([[  6.   , 148.   ,  72.   , ...,  33.6  ,   0.627,  50.   ],
       [  1.   ,  85.   ,  66.   , ...,  26.6  ,   0.351,  31.   ],
       [  8.   , 183.   ,  64.   , ...,  23.3  ,   0.672,  32.   ],
       ...,
       [  5.   , 121.   ,  72.   , ...,  26.2  ,   0.245,  30.   ],
       [  1.   , 126.   ,  60.   , ...,  30.1  ,   0.349,  47.   ],
       [  1.   ,  93.   ,  70.   , ...,  30.4  ,   0.315,  23.   ]])
```

```
y=df.iloc[:,-1].values
y
```

```
array([1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
```

```
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0])
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
print(x_train)
```

```
    [[1.200e+01 8.400e+01 7.200e+01 ... 2.970e+01 2.970e-01 4.600e+01]
     [7.000e+00 1.140e+02 6.400e+01 ... 2.740e+01 7.320e-01 3.400e+01]
     [3.000e+00 1.230e+02 1.000e+02 ... 5.730e+01 8.800e-01 2.200e+01]
     ...
     [4.000e+00 1.970e+02 7.000e+01 ... 3.670e+01 2.329e+00 3.100e+01]
     [8.000e+00 1.120e+02 7.200e+01 ... 2.360e+01 8.400e-01 5.800e+01]
     [1.000e+01 1.110e+02 7.000e+01 ... 2.750e+01 1.410e-01 4.000e+01]]
```

```
#standard scaler    z=(x-u)/s      u-mean of training data , s-standard deviatic
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
print(x_train)
```

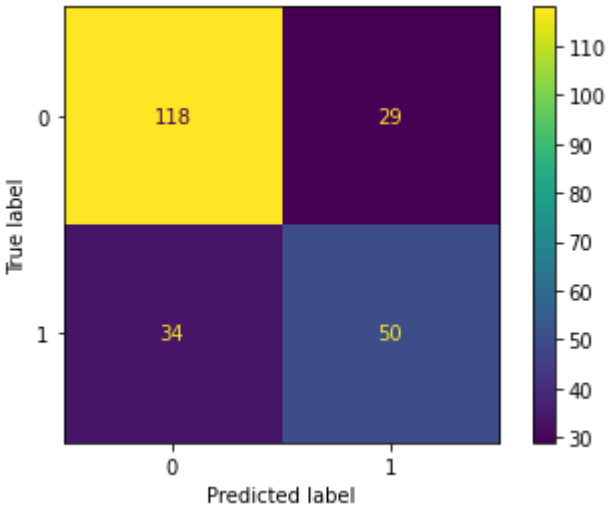
```
    [[ 2.42649861 -1.13872537  0.13870722 ... -0.27290805 -0.52355375
       1.0534661 ]
     [ 0.93013922 -0.19501089 -0.28967971 ... -0.56550243  0.81845058
       0.05132031]
     [-0.26694829  0.08810345  1.6380615  ...  3.23822457  1.27504056
      -0.95082547]
     ...
     [ 0.03232359  2.4159325   0.03161049 ...  0.6175966   5.74530327
      -0.19921613]
     [ 1.2294111  -0.25792519  0.13870722 ... -1.04891924  1.15163786
       2.05561188]
     [ 1.82795485 -0.28938234  0.03161049 ... -0.55278094 -1.00482427
       0.55239321]]
```

```
#KNN Algorithm
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
print(y_pred)
print(classifier.predict([[0,137,400,35,168,43.1,2.288,33]]))
```

```
[0 0 1 0 0 0 0 0 1 1 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 1 0
 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0
 0 1 0 1 0 0 0 1 0 1 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0
 1 0 0 0 0 1 1 0 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0
 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0
 1 1 0 0 1 1 1 1 0]
[1]
```

```
from sklearn.metrics import classification_report,accuracy_score,ConfusionMatri›
result=confusion_matrix(y_test,y_pred)
cm=ConfusionMatrixDisplay(result)
cm.plot()
score=accuracy_score(y_test,y_pred)
print(result)
print(score)
```

```
[[118  29]
 [ 34  50]]
0.7272727272727273
```



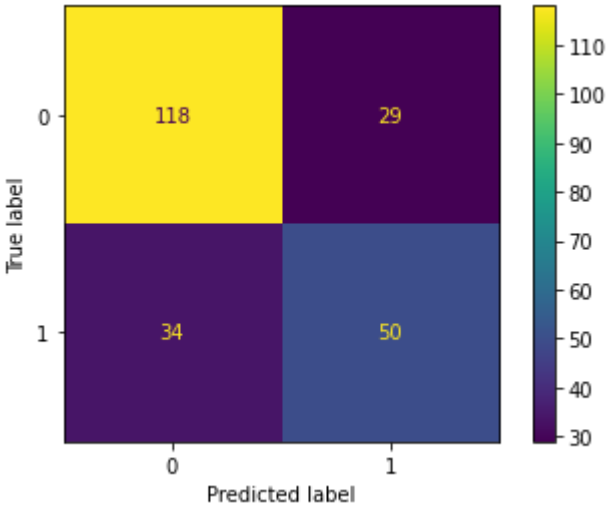

```
#NAIVE BAYES Algorithm
from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print(y_pred)
```

```
[0 0 1 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1 0 0 0 0 1 0 1 0 1 0 1 1 1 1 0
 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0
```

```
 0 1 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1 0
 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 0
 0 0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 0 0 0 0 1 1 0 1 1 0 0 0 1 0 0 1 0 0 1 1 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 1 0 1
 1 1 0 0 1 0 1 1 0]
```

```
from sklearn.metrics._plot.confusion_matrix import confusion_matrix
from sklearn.metrics import classification_report,accuracy_score,ConfusionMatri›
result=confusion_matrix(y_test,y_pred)
cm=ConfusionMatrixDisplay(result)
cm.plot()
score=accuracy_score(y_test,y_pred)
print(result)
print(score)
```

```
[[118  29]
 [ 34  50]]
0.7272727272727273
```



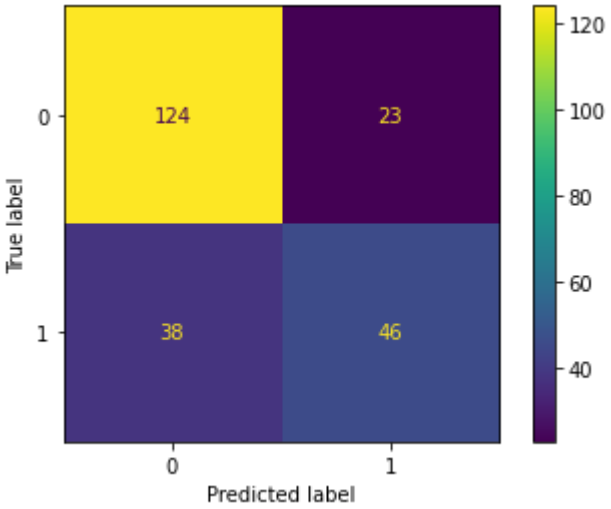

```
#SUPPORT VECTOR MACHINE(SVM) Algorithm
from sklearn.svm import SVC
model=SVC()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
print(y_pred)
```

```
[0 0 1 0 0 0 1 0 1 1 0 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0
 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 1 1 0 1 0 1 0
 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0
 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0
 0 0 0 1 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0
 0 1 0 0 1 0 1 0 0]
```

```
from sklearn.metrics import classification_report,accuracy_score,ConfusionMatri›
result=confusion_matrix(y_test,y_pred)
cm=ConfusionMatrixDisplay(result)
cm.plot()
score=accuracy_score(y_test,y_pred)
```

```
print(result)
print(score)
```

```
[[124  23]
 [ 38  46]]
0.7359307359307359
```



New Section

New Section