```
import pandas as pd
```

```
df=pd.read_csv("Iris.csv")
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthC
0	1	5.1	3.5	1.4	0
1	2	4.9	3.0	1.4	0
2	3	4.7	3.2	1.3	0
3	4	4.6	3.1	1.5	0
4	5	5.0	3.6	1.4	0
...	...	...	...	...	
145	146	6.7	3.0	5.2	2
146	147	6.3	2.5	5.0	1

```
df.describe()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

```
x=df.drop(columns=['Id','Species'],axis=1)
print("\n",x)
```

```
      SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0               5.1           3.5            1.4           0.2
1               4.9           3.0            1.4           0.2
2               4.7           3.2            1.3           0.2
3               4.6           3.1            1.5           0.2
4               5.0           3.6            1.4           0.2
..              ...           ...            ...           ...
145             6.7           3.0            5.2           2.3
146             6.3           2.5            5.0           1.9
147             6.5           3.0            5.2           2.0
```

```
148            6.2           3.4            5.4           2.3
149            5.9           3.0            5.1           1.8

[150 rows x 4 columns]
```

```
# x=df.drop('Species',axis=1)
y=df['Species']
print("\n",y)
```

```
 0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
# print(x_train)
# print(x_test)
# print(y_train)
# print(y_test)
```

```
from sklearn.linear_model import LogisticRegression
```

```
model1=LogisticRegression()
model1.fit(x_train,y_train)
Pred_y=model1.predict(x_test)
# print(Pred_y)
```

```
c:\Python311\Lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
from sklearn.metrics import accuracy_score
acc=accuracy_score(y_test,Pred_y)
print(acc)
```

```
0.9777777777777777
```

```
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
x_test=std.fit_transform(x_test)
x_train=std.fit_transform(x_train)
# print(x_test)
# print(x_train)
```

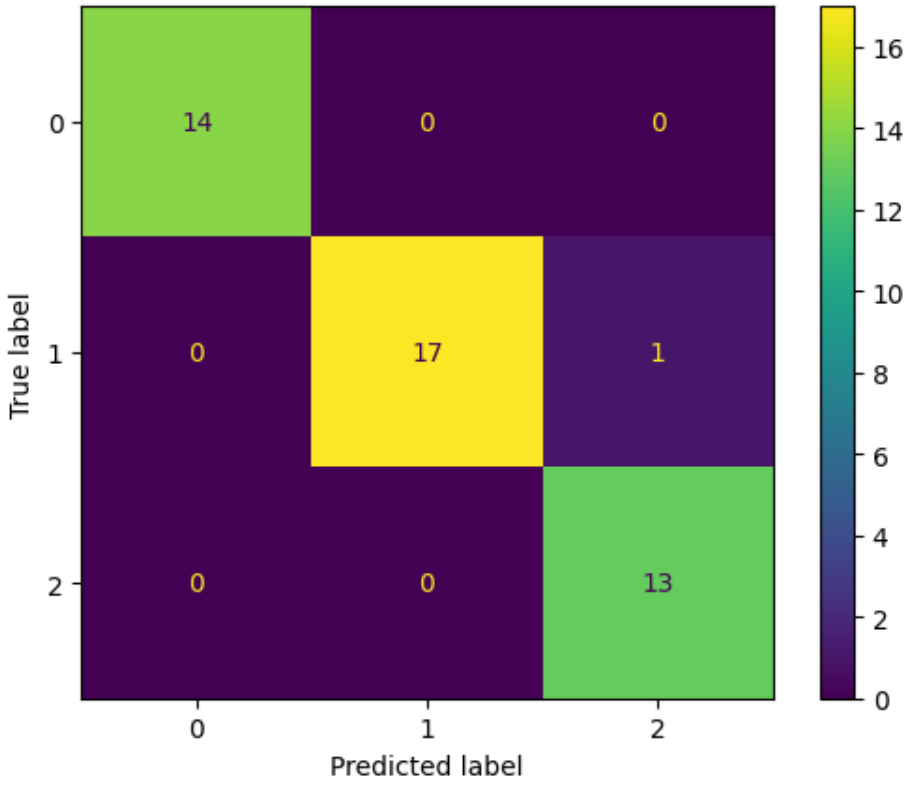
```
from sklearn.naive_bayes import GaussianNB


model2=GaussianNB()
NB_model=model2.fit(x_train,y_train)
y_pred=NB_model.predict(x_test)
print(y_pred)
acc2=accuracy_score(y_pred,y_test)
print(acc2)
```

```
['Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica'
 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor']
0.9555555555555556
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
cm=confusion_matrix(y_test,Pred_y)
ConfusionMatrixDisplay(cm).plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1ff6206c590>
```




```
from sklearn.metrics import recall_score,precision_score
print('recall = ',recall_score(y_test,Pred_y,average='micro'))
print('precision = ',precision_score(y_test,Pred_y,average='micro'))
```

```
recall =  0.9777777777777777
precision =  0.9777777777777777
```

```
from sklearn.metrics import classification_report
print(classification_report(y_pred,y_test))
```

```
                    precision    recall  f1-score   support

        Iris-setosa       1.00      1.00      1.00        14
    Iris-versicolor       0.94      0.94      0.94        18
     Iris-virginica       0.92      0.92      0.92        13

           accuracy                           0.96        45
          macro avg       0.96      0.96      0.96        45
       weighted avg       0.96      0.96      0.96        45
```

```
new_data=[[5.5,3.5,1.2,0.2],[4.2,5.6,3.4,2.8],[6.1,3.9,1.9,2.1]]
new_pred=NB_model.predict(new_data)
print(new_pred)
```

```
['Iris-virginica' 'Iris-virginica' 'Iris-virginica']
```

```
new_data2=[[1.5,1.3,1.2,1.9]]
new_pred2=NB_model.predict(new_data2)
print(new_pred2)
```

```
['Iris-virginica']
```

Diabetes Dataset

```
df2=pd.read_csv("diabetes.csv")
df2
```

	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 × 9 columns

```
x2=df2.drop('Outcome',axis=1)
print(x)
y2=df2['Outcome']
print(y)
```

```
     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]
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
```

```
x_train2,x_test2,y_train2,y_test2=train_test_split(x2,y2,test_size=0.3,random_state=1)
```

```
model3=LogisticRegression()
model3.fit(x_train2,y_train2)
Pred_y2=model3.predict(x_test2)
print(Pred_y2)
```

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

```
acc3=accuracy_score(y_test2,Pred_y2)
print(acc3)
```

```
0.7835497835497836
```

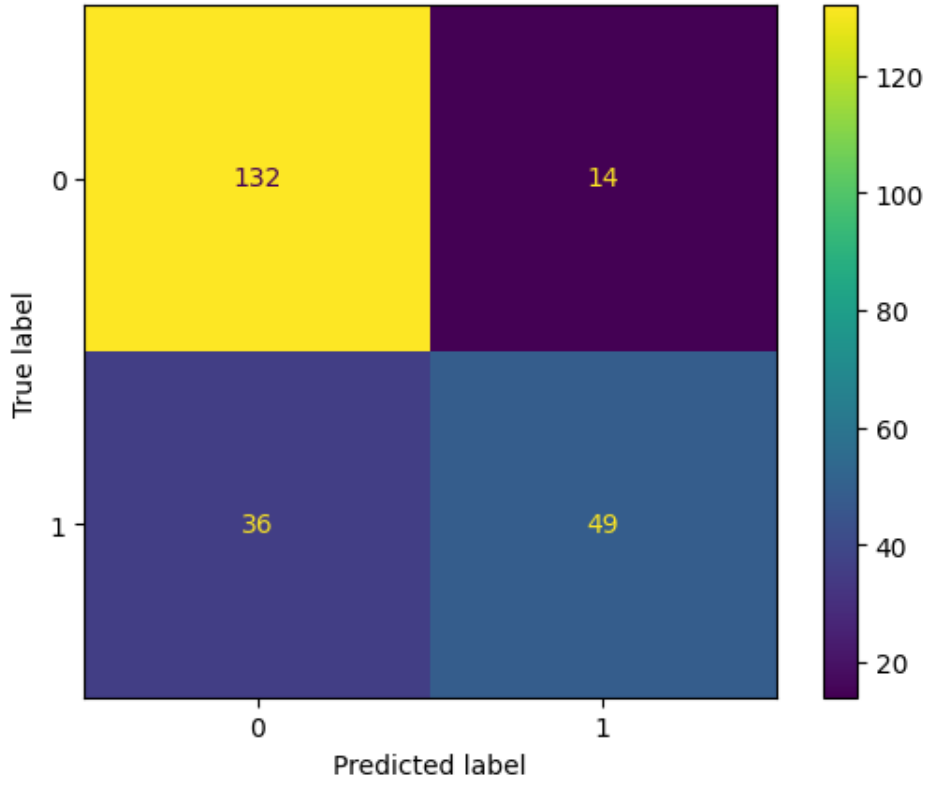
```
std2=StandardScaler()
x_test2=std2.fit_transform(x_test2)
x_train2=std2.fit_transform(x_train2)
```

```
model4=GaussianNB()
NB_model2=model4.fit(x_train2,y_train2)
y_pred2=NB_model2.predict(x_test2)
print(y_pred2)
acc4=accuracy_score(y_pred2,y_test2)
print(acc4)
```

```
[0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 0
 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 1 0 0
 1 0 0 0 0 1 1 0 0 0 0 1 1 0 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 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1 0 0 1 1 1 0 0
 0 0 0 1 0 1 1 1 0 0 0 0 1 1 0 0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 1 0 0 1 0
 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 1
 0 1 0 0 0 0 0 1 0]
0.7835497835497836
```

```
cm2=confusion_matrix(y_test2,Pred_y2)
ConfusionMatrixDisplay(cm2).plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1ff6257ab50>
```




```
print(classification_report(y_pred2,y_test2))
```

```
              precision    recall  f1-score   support

           0       0.88      0.80      0.84       160
           1       0.62      0.75      0.68        71

    accuracy                           0.78       231
   macro avg       0.75      0.77      0.76       231
weighted avg       0.80      0.78      0.79       231
```