```
import numpy as np
import pandas as pd
import seaborn as sea
import matplotlib.pyplot as plt
```

```
data = pd.read_csv('/content/iris.csv')
data.head()
```

| | sepallength | sepalwidth | petallength | petalwidth | class1 |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

Double-click (or enter) to edit

```
data.describe()
```

| | sepallength | sepalwidth | petallength | petalwidth |
|---|---|---|---|---|
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| **std** | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| **min** | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| **75%** | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

```
data['class1'].value_counts()
```

```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: class1, dtype: int64
```

```
data.isnull().sum()
```

```
sepallength    0
sepalwidth     0
petallength    0
```

```
petalwidth     0
class1         0
dtype: int64
```

```
data.corr()
```

| | sepallength | sepalwidth | petallength | petalwidth |
|---|---|---|---|---|
| **sepallength** | 1.000000 | -0.109369 | 0.871754 | 0.817954 |
| **sepalwidth** | -0.109369 | 1.000000 | -0.420516 | -0.356544 |
| **petallength** | 0.871754 | -0.420516 | 1.000000 | 0.962757 |
| **petalwidth** | 0.817954 | -0.356544 | 0.962757 | 1.000000 |

```
from sklearn.preprocessing import LabelEncoder
enc=LabelEncoder()
data['class1']=enc.fit_transform(data['class1'])
print(data['class1'])
```

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: class1, Length: 150, dtype: int64
```

```
data.head()
```

| | sepallength | sepalwidth | petallength | petalwidth | class1 |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | 0 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | 0 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | 0 |

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

```
X=data.drop(columns=['class1'])
Y=data['class1']
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.5,random_state=0)
```

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model=model.fit(X_train,Y_train)
pred=model.predict(X_test)
print("Accuracy = ",model.score(X_test,Y_test)*100)
```

```
Accuracy =  93.33333333333333
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
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
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

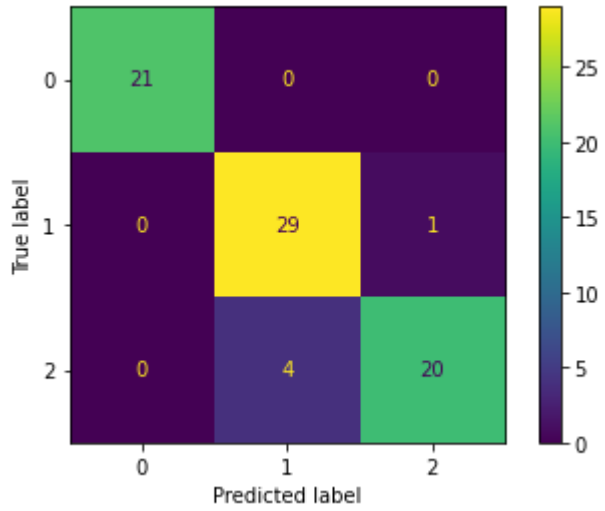
```
from sklearn.metrics import classification_report, plot_confusion_matrix,precision_score,recal
print(classification_report(Y_test,pred))
plot_confusion_matrix(model,X_test,Y_test)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       0.88      0.97      0.92        30
           2       0.95      0.83      0.89        24

    accuracy                           0.93        75
   macro avg       0.94      0.93      0.94        75
weighted avg       0.94      0.93      0.93        75

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fd9034fa4d0>
```




```
from sklearn.metrics import confusion_matrix
print('Confusion Matrix = ',confusion_matrix(Y_test,pred))
print('Precision = ',(precision_score(Y_test,pred,average='micro')))
```

```
Confusion Matrix =  [[21  0  0]
 [ 0 29  1]
```

```
 [ 0  4 20]]
Precision =  0.9333333333333333
```

```
pip install cv
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Collecting cv
  Downloading cv-1.0.0-py3-none-any.whl (7.3 kB)
Installing collected packages: cv
Successfully installed cv-1.0.0
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
```

Colab paid products - Cancel contracts here