```
from sklearn import datasets
iris = datasets.load_iris()
type(iris)
```

```
sklearn.utils.Bunch
```

```
iris.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filenam
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
iris.target_names,iris.target
```

```
(array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]))
```

```
iris.data.shape
```

```
(150, 4)
```

```
iris.target.shape
```

```
(150,)
```

```
import pandas as pd
x = iris.data
y = iris.target
df = pd.DataFrame(x, columns = iris.feature_names)
df['target'] = y
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
model =  KNeighborsClassifier(n_neighbors = 6)
```

```
model.fit(x,y)
```

```
KNeighborsClassifier(n_neighbors=6)
```

```
import numpy as np
sample = np.array([[5.3,4.1,3,0.2],[4.8,3.1,1.4,0.2],[4.5,3.1,1.45,0.2]])
result = model.predict(sample)
result
```

```
array([0, 0, 0])
```

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size = 0.2, shuffle =True )
xtrain.shape ,xtest.shape ,ytrain.shape ,ytest.shape
```

```
((120, 4), (30, 4), (120,), (30,))
```

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors = 6)
model.fit(xtrain , ytrain)
```

```
KNeighborsClassifier(n_neighbors=6)
```

```
results = model.predict(xtest)
result
```

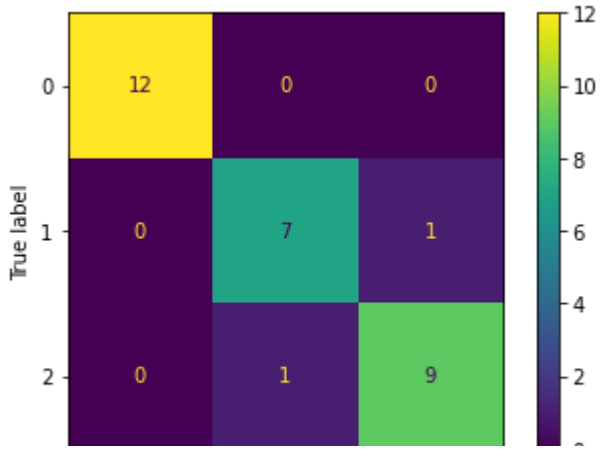
```
array([0, 0, 0])
```

```
from sklearn.metrics import accuracy_score
accuracy  = accuracy_score(results,ytest)
accuracy
```

```
0.9333333333333333
```

```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(model,xtest,ytest)
```

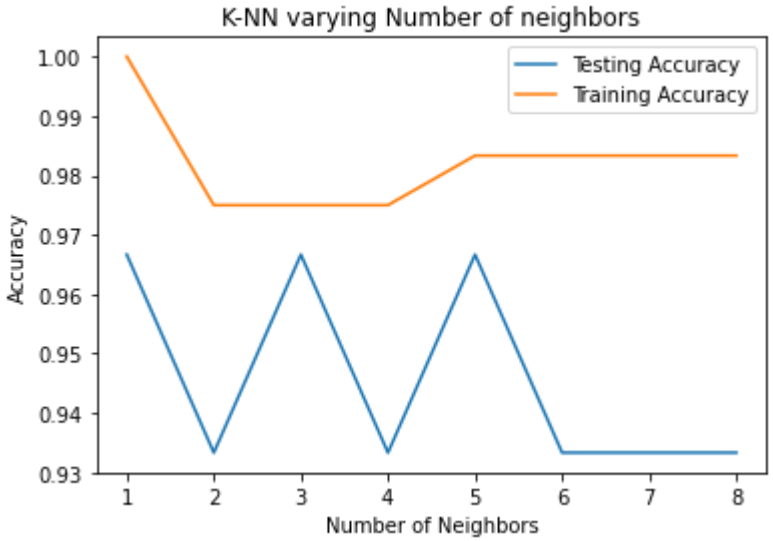
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f370e1749d0>
```




```
from sklearn.metrics import classification_report
print(classification_report(ytest , results))
```

```
                  precision    recall  f1-score   support

               0       1.00      1.00      1.00        12
               1       0.88      0.88      0.88         8
               2       0.90      0.90      0.90        10

        accuracy                           0.93        30
       macro avg       0.92      0.92      0.92        30
    weighted avg       0.93      0.93      0.93        30
```

```
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
total_train_accuracy= []
total_test_accuracy = []
neighbors=np.arange(1,9)


for i in range(8):
    knn  = KNeighborsClassifier(n_neighbors= i+1)
    knn.fit(xtrain, ytrain)
    train_accuracy = knn.score(xtrain,ytrain)
    test_accuracy = knn.score(xtest,ytest)
    total_train_accuracy.append(train_accuracy)
    total_test_accuracy.append(test_accuracy)


plt.title("K-NN varying Number of neighbors")
plt.plot(neighbors,total_test_accuracy,label = 'Testing Accuracy')
plt.plot(neighbors,total_train_accuracy,label = 'Training Accuracy')
plt.legend()
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.show()
```




```
from sklearn import datasets
mnist = datasets.load_digits()
mnist.target_names
x = mnist.data
y = mnist.target


sample = 100
image = x[sample]
image = image.reshape(8,8)
plt.imshow(image)
plt.show()
```