```
from sklearn import datasets
import pandas as pd
import numpy as np
```

```
iris=datasets.load_iris()
```

```
iris
```

```
            [6.4, 2.8, 5.6, 2.1],
            [7.2, 3. , 5.8, 1.6],
            [7.4, 2.8, 6.1, 1.9],
            [7.9, 3.8, 6.4, 2. ],
            [6.4, 2.8, 5.6, 2.2],
            [6.3, 2.8, 5.1, 1.5],
            [6.1, 2.6, 5.6, 1.4],
            [7.7, 3. , 6.1, 2.3],
            [6.3, 3.4, 5.6, 2.4],
            [6.4, 3.1, 5.5, 1.8],
            [6. , 3. , 4.8, 1.8],
            [6.9, 3.1, 5.4, 2.1],
            [6.7, 3.1, 5.6, 2.4],
            [6.9, 3.1, 5.1, 2.3],
            [5.8, 2.7, 5.1, 1.9],
            [6.8, 3.2, 5.9, 2.3],
            [6.7, 3.3, 5.7, 2.5],
            [6.7, 3. , 5.2, 2.3],
            [6.3, 2.5, 5. , 1.9],
            [6.5, 3. , 5.2, 2. ],
            [6.2, 3.4, 5.4, 2.3],
            [5.9, 3. , 5.1, 1.8]]),
     'target': 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,
            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, 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, 2, 2, 2, 2, 2, 2, 2, 2]),
     'frame': None,
     'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
     'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 150 (50
    in each of three classes)\n    :Number of Attributes: 4 numeric, predictive attributes and the class\n    :Attribute Information:\n        -
    sepal length in cm\n        - sepal width in cm\n        - petal length in cm\n        - petal width in cm\n        - class:\n                -
    Iris-Setosa\n                - Iris-Versicolour\n                - Iris-Virginica\n                \n    :Summary Statistics:\n\n
    ============== ==== ==== ======= ===== ====================\n                    Min  Max   Mean    SD   Class Correlation\n    ==============
    ==== ==== ======= ===== ====================\n    sepal length:   4.3  7.9   5.84   0.83    0.7826\n    sepal width:    2.0  4.4   3.05   0.43
    -0.4194\n    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\n    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)\n
    ============== ==== ==== ======= ===== ====================\n\n    :Missing Attribute Values: None\n    :Class Distribution: 33.3% for each of 3
    classes.\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe famous Iris
    database, first used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but not as in the
    UCI\nMachine Learning Repository, which has two wrong data points.\n\nThis is perhaps the best known database to be found in the\npattern
    recognition literature.  Fisher\'s paper is a classic in the field and\nis referenced frequently to this day.  (See Duda & Hart, for example.)
    The\ndata set contains 3 classes of 50 instances each, where each class refers to a\ntype of iris plant.  One class is linearly separable from
    the other 2; the\nlatter are NOT linearly separable from each other.\n\n.. topic:: References\n\n   - Fisher, R.A. "The use of multiple
    measurements in taxonomic problems"\n     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to\n     Mathematical Statistics"
    (John Wiley, NY, 1950).\n   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n     (Q327.D83) John Wiley & Sons.
    ISBN 0-471-22361-1.  See page 218.\n   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n     Structure and Classification
    Rule for Recognition in Partially Exposed\n     Environments".  IEEE Transactions on Pattern Analysis and Machine\n     Intelligence, Vol. PAMI-
    2, No. 1, 67-71.\n   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n     on Information Theory, May 1972, 431-
    433.\n   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n     conceptual clustering system finds 3 classes in the
    data.\n   - Many, many more ...',
     'feature_names': ['sepal length (cm)',
      'sepal width (cm)',
      'petal length (cm)',
      'petal width (cm)'],
     'filename': 'iris.csv',
     'data_module': 'sklearn.datasets.data'}
```

```
iris.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

```
iris=pd.DataFrame(data=np.c_[iris['data'],iris['target']],columns=iris['feature_names']+['target'])
```

```
species=[]
for i in range(len(iris['target'])):
    if iris['target'][i]==0:
        species.append('setosa')
    elif iris['target'][i]==1:
        species.append('versicolor')
    elif iris['target'][i]==2:
        species.append('verginica')
```

```
iris['species']=species
```

```
iris
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	species
0	5.1	3.5	1.4	0.2	0.0	setosa
1	4.9	3.0	1.4	0.2	0.0	setosa
2	4.7	3.2	1.3	0.2	0.0	setosa
3	4.6	3.1	1.5	0.2	0.0	setosa
4	5.0	3.6	1.4	0.2	0.0	setosa
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2.0	verginica
146	6.3	2.5	5.0	1.9	2.0	verginica
147	6.5	3.0	5.2	2.0	2.0	verginica

```
from sklearn.model_selection import train_test_split
x=iris.drop(['target','species'],axis=1)
y=iris['target']
```

```
x
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
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 × 4 columns

```
y
```

```
0      0.0
1      0.0
2      0.0
3      0.0
4      0.0
      ...
145    2.0
146    2.0
147    2.0
148    2.0
149    2.0
Name: target, Length: 150, dtype: float64
```

```
x_train,x_valid_test,y_train,y_valid_test=train_test_split(x,y,test_size=0.3)
```

```
x_valid,x_test,y_valid,y_test=train_test_split(x_valid_test,y_valid_test,test_size=0.5)
```

```
print(len(x_train),len(x_valid),len(x_test))
```

```
105 22 23
```

```
print(len(x_train),len(x_valid),len(x_test))
```

```
105 22 23
```

```
from sklearn.linear_model import LogisticRegression
logmodel=LogisticRegression()
```

```
logmodel.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
val=logmodel.predict(x_valid)
```

```
val
```

```
array([0., 2., 2., 1., 1., 2., 1., 2., 2., 2., 1., 0., 0., 0., 1., 2., 1.,
       2., 2., 0., 0., 2.])
```

```
valiid_prediction=logmodel.predict(x_valid)
```

```
valiid_prediction
```

```
array([0., 2., 2., 1., 1., 2., 1., 2., 2., 2., 1., 0., 0., 0., 1., 2., 1.,
       2., 2., 0., 0., 2.])
```

```
training_prediction=logmodel.predict(x_train)
```

```
training_prediction
```

```
array([1., 0., 1., 1., 2., 0., 1., 0., 1., 1., 0., 1., 1., 1., 2., 2., 1.,
       0., 1., 1., 2., 2., 0., 0., 2., 2., 1., 2., 2., 2., 0., 2., 2., 0.,
       2., 1., 1., 2., 0., 2., 1., 2., 1., 1., 0., 0., 2., 1., 2., 2., 0.,
       1., 0., 2., 1., 0., 1., 1., 2., 2., 1., 0., 0., 1., 1., 0., 1., 0.,
       1., 0., 0., 0., 0., 2., 0., 2., 0., 0., 0., 1., 0., 0., 0., 0., 1.,
       1., 0., 1., 1., 1., 1., 1., 0., 1., 0., 0., 0., 2., 1., 1., 1.,
       0., 2., 0.])
```

```
from sklearn.metrics import classification_report, confusion_matrix
```

```
print(classification_report(y_train,training_prediction))
```

```
print(classification_report(y_valid,valiid_prediction))
```

```
                  precision    recall  f1-score   support

             0.0       1.00      1.00      1.00        38
             1.0       0.93      1.00      0.96        38
             2.0       1.00      0.90      0.95        29

        accuracy                           0.97       105
       macro avg       0.98      0.97      0.97       105
    weighted avg       0.97      0.97      0.97       105

                  precision    recall  f1-score   support

             0.0       1.00      1.00      1.00         6
             1.0       0.83      1.00      0.91         5
             2.0       1.00      0.91      0.95        11

        accuracy                           0.95        22
       macro avg       0.94      0.97      0.95        22
    weighted avg       0.96      0.95      0.96        22
```

```
print(confusion_matrix(y_train,training_prediction))
```

```
[[38  0  0]
 [ 0 38  0]
 [ 0  3 26]]
```

```
print(confusion_matrix(y_valid,valiid_prediction))
```

```
[[ 6  0  0]
 [ 0  5  0]
 [ 0  1 10]]
```