01iris

December 21, 2021

1 Scikit-Learn

1.0.1 Iris - , (SVM),

1.

[ ]:
```
from sklearn.datasets import load_iris
import pandas as pd
iris = load_iris()
```

[ ]:
```
type(iris)
```

[ ]:
```
sklearn.utils.Bunch
```

[ ]:
```
iris.keys()
```

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

[ ]:
```
# Feature data - numpy dimensional array
iris.data[:5]
```

[ ]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

[ ]:
```
type(iris.data), iris.data.shape
```

[ ]:
```
(numpy.ndarray, (150, 4))
```

[ ]:
```
# Feature name
iris.feature_names
```

[ ]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

[ ]:
```
# map  lambda
iris_feature_names = list(map(lambda s : s[:-5], iris.feature_names))
iris_feature_names
```

[ ]:
```
['sepal length', 'sepal width', 'petal length', 'petal width']
```

[ ]:
```
feature_names = []
for name in iris.feature_names:
    feature_names.append(name[:-5])

    feature_names
```

[ ]:
```
#map  lmabda
feature_names = map(lambda s: s[:-5], iris.feature_names)
feature_names
```

[ ]:
```
<map at 0x7fd55b3f2ca0>
```

[ ]:
```
s = "special lenth (cm)"
s[:-5]
```

[ ]:
```
'special lenth'
```

[ ]:
```
a, b, c = map(int, input().split())
```

[ ]:
```
a,b,c
```

[ ]:
```
(10, 20, 100)
```

[ ]:
```
# Target data - Y
iris.target[:5]
```

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

[ ]:
```
import numpy as np
np.unique(iris.target, return_counts=True)
```

[ ]:
```
(array([0, 1, 2]), array([50, 50, 50]))
```

[ ]:
```
# iris.data  2        , trarget  y    .
import pandas as pd
df = pd.DataFrame(iris.data, columns = iris.feature_names)
df["target"] = iris.target
df.head()
```

[ ]:
```
   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

   target
0       0
1       0
2       0
3       0
4       0
```

[ ]:
```
iris.target_names
```

[ ]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

[ ]:
```
#
print(iris.DESCR)
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================
```

```
    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more …
```

[ ]:
```
df.describe()
```

[ ]:
```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)      target
```

```
count        150.000000  150.000000
mean           1.199333    1.000000
std            0.762238    0.819232
min            0.100000    0.000000
25%            0.300000    0.000000
50%            1.300000    1.000000
75%            1.800000    2.000000
max            2.500000    2.000000
```

[ ]:
```
df.groupby("target").describe()
```

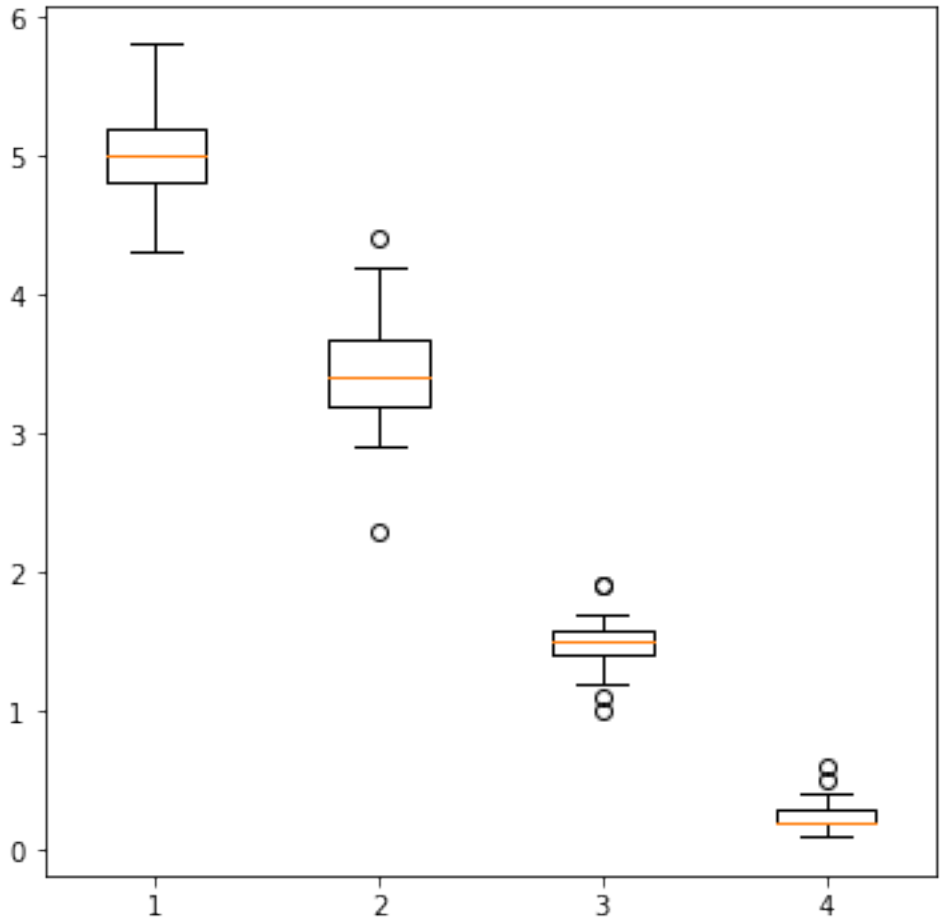
[ ]:
```
       sepal length (cm)                                                \
                   count   mean       std  min    25%  50%  75%  max
target
0                   50.0  5.006  0.352490  4.3  4.800  5.0  5.2  5.8
1                   50.0  5.936  0.516171  4.9  5.600  5.9  6.3  7.0
2                   50.0  6.588  0.635880  4.9  6.225  6.5  6.9  7.9

       sepal width (cm)         … petal length (cm)      petal width (cm)  \
                  count   mean  …                75%  max            count
target                         …
0                  50.0  3.428  …              1.575  1.9             50.0
1                  50.0  2.770  …              4.600  5.1             50.0
2                  50.0  2.974  …              5.875  6.9             50.0


         mean       std  min  25%  50%  75%  max
target
0       0.246  0.105386  0.1  0.2  0.2  0.3  0.6
1       1.326  0.197753  1.0  1.2  1.3  1.5  1.8
2       2.026  0.274650  1.4  1.8  2.0  2.3  2.5

[3 rows x 32 columns]
```

[ ]:
```
import matplotlib.pyplot as plt
setosa = df[df.target == 0].iloc[:,:-1]
plt.figure(figsize=(6,6))
plt.boxplot(setosa)
plt.show()
```



2.

```
[ ]: from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(
         iris.data, iris.target, test_size = 0.2, random_state = 2021
     )
     X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
[ ]: ((120, 4), (30, 4), (120,), (30,))
```

```
[ ]: # de facto standard
     np.unique(y_train, return_counts = True)
```

```
[ ]: (array([0, 1, 2]), array([36, 40, 44]))
```

```
[ ]: # y
     X_train, X_test, y_train, y_test = train_test_split(
         iris.data, iris.target, stratify = iris.target, test_size = 0.2,␣
      ↪random_state = 2021
```

```
)
np.unique(y_train,return_counts = True)
```

[ ]: (array([0, 1, 2]), array([40, 40, 40]))

[ ]:
```
np.random.seed(2021)
np.random.randint(0, 101, 10)
```

[ ]: array([85, 57, 0, 94, 86, 44, 62, 91, 29, 21])

2

- Decision Tree ( )

[ ]:
```
from sklearn.tree import DecisionTreeClassifier
```

[ ]:
```
#     -
dtc = DecisionTreeClassifier(random_state = 2021)
```

[ ]:
```
#
dtc.get_params()
```

[ ]:
```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': 2021,
 'splitter': 'best'}
```

[ ]:
```
#  ( )
dtc.fit(X_train, y_train)
```

[ ]: DecisionTreeClassifier(random_state=2021)

4.

[ ]:
```
pred_dt = dtc.predict(X_test)
```

[ ]:
```
pred_dt
```

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

[ ]:
```
res = pd.DataFrame({'y':y_test, "DT" : pred_dt})
res.head()
```

[ ]:

	y	DT
0	0	0
1	1	1
2	1	2
3	2	2
4	0	0

5.

[ ]:
```
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, pred_dt) # (y ,   )
print(f"  (DT) : {acc:.4f}")
```

```
  (DT) : 0.9000
```

[ ]:
```
# 4, 5
dtc.score(X_test, y_test)
```

[ ]:
```
0.9
```

- Support Vector Machine (SVM)

[ ]:
```
# 3)
from sklearn.svm import SVC
svc = SVC(random_state = 2021)
svc.fit(X_train, y_train)
```

[ ]:
```
SVC(random_state=2021)
```

[ ]:
```
# 4)
pred_sv = svc.predict(X_test)
```

[ ]:
```
# 5)
accuracy_score(y_test, pred_sv)
svc.score(X_test, y_test)
```

[ ]:
```
0.9
```

- Logistic Regression

[ ]:
```
# 3)
from sklearn.linear_model import LogisticRegression
lrc = LogisticRegression(max_iter = 500, random_state = 2021)
lrc.fit(X_train, y_train)
```

[ ]:
```
LogisticRegression(max_iter=500, random_state=2021)
```

[ ]:
```
# 4)
pred_lr = lrc.predict(X_test)
```

[ ]:
```
# 5)
accuracy_score(y_test, pred_lr), lrc.score(X_test, y_test)
```

[ ]:
```
(0.9333333333333333, 0.9333333333333333)
```

3 3

[ ]:
```
res["SV"] = pred_sv
res["LR"] = pred_lr
res.head(10)
```

[ ]:

	y	DT	SV	LR
0	0	0	0	0
1	1	1	1	1
2	1	2	1	1
3	2	2	2	2
4	0	0	0	0
5	1	1	1	1
6	0	0	0	0
7	1	1	1	1
8	2	2	2	2
9	0	0	0	0