class1

March 13, 2022

0.1 1.4.2 Numpy

[8]:
```
import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6]])
print("x:\n{}".format(x))
```

```
x:
[[1 2 3]
 [4 5 6]]
```

0.2 1.4.3 SciPy

[2]:
```
from scipy import sparse

#    Numpy    1   0
eye = np.eye(4)
print("Numpy array:\n{}".format(eye))
```

```
Numpy array:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

[3]:
```
# NumPy   CSR  SciPy
#
spare_matrix = sparse.csr_matrix(eye)
print("\nSciPy sparse CSR matrix:\n{}".format(spare_matrix))
```

```
SciPy sparse CSR matrix:
  (0, 0)        1.0
  (1, 1)        1.0
  (2, 2)        1.0
  (3, 3)        1.0
```

[4]:
```
data = np.ones(4)
row_indices = np.arange(4)
```

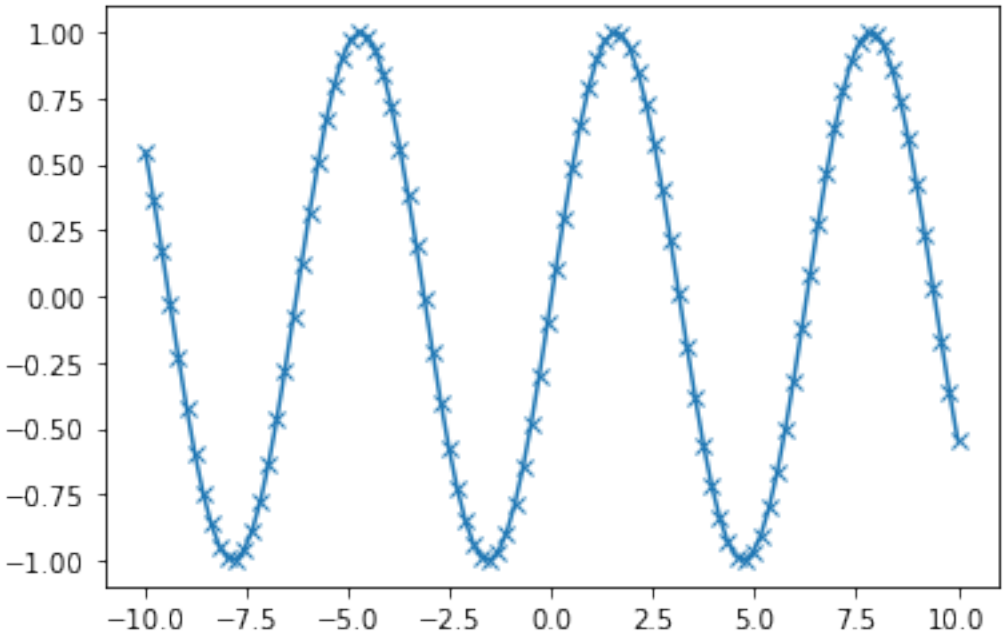
```
col_indices = np.arange(4)
eye_coo = sparse.coo_matrix((data, (row_indices, col_indices)))
print("COO representation:\n{}".format(eye_coo))
```

```
COO representation:
  (0, 0)        1.0
  (1, 1)        1.0
  (2, 2)        1.0
  (3, 3)        1.0
```

0.3 1.4.4 matplotlib

[9]:
```
# % matplotlib inline
import matplotlib.pyplot as plt

#  -10 10      100
x = np.linspace(-10, 10, 100)
#
y = np.sin(x)
# plot
plt.plot(x, y, marker="x")
#    show()    [<matplotlib.lines.Line2D at 0x252768a3040>]
plt.show()
```



0.4 1.4.5 pandas

[10]:
```
import pandas as pd
from IPython.display import display

#
data = {'Name': ["John", "Anna", "Peter", "Linda"],
        'Location': ["New York", "Paris", "Berlin", "London"],
        'Age': [24, 13, 53, 33]
        }

data_pandas = pd.DataFrame(data)
# Ipython.display  Jupyter Notebook  "  "DataFrame
display(data_pandas)
```

```
    Name  Location  Age
0   John  New York   24
1   Anna     Paris   13
2  Peter    Berlin   53
3  Linda    London   33
```

[11]:
```
#    30
display(data_pandas[data_pandas.Age > 30])
display(data_pandas[data_pandas.Name > "John"])
```

```
    Name Location  Age
2  Peter   Berlin   53
3  Linda   London   33
```

```
    Name Location  Age
2  Peter   Berlin   53
3  Linda   London   33
```

1 1.7

1.1 1.7.1

[12]:
```
import mglearn
from sklearn.datasets import load_iris

iris_dataset = load_iris()
```

[13]:
```
print("Keys of iris_dataset :\n{}".format(iris_dataset.keys()))
```

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

[14]:
```
print(iris_dataset['DESCR'][:193] + "\n...")
```

```
.. _iris_dataset:

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

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, pre
…
```

[15]:
```
print("Target names : {}".format(iris_dataset['target_names']))
```

```
Target names : ['setosa' 'versicolor' 'virginica']
```

[16]:
```
print("Feature names : \n{}".format(iris_dataset['feature_names']))
```

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

[17]:
```
print("Type of data : {}".format(type(iris_dataset['data'])))
```

```
Type of data : <class 'numpy.ndarray'>
```

[18]:
```
print("Shape of data : {}".format(iris_dataset['data'].shape))
```

```
Shape of data : (150, 4)
```

[19]:
```
print("First five rows of data:\n{}".format(iris_dataset['data'][:5]))
```

```
First five rows of data:
[[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]]
```

[20]:
```
print("Type of target : {}".format(type(iris_dataset['target'])))
```

```
Type of target : <class 'numpy.ndarray'>
```

[21]:
```
print("Shape of target: {}".format(iris_dataset['target'].shape))
```

```
Shape of target: (150,)
```

[22]:
```
print("Target:\n{}".format(iris_dataset['target']))
```

```
Target:
[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 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]
```

1.2 1.7.2

[23]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    iris_dataset['data'], iris_dataset['target'], random_state=0)
```

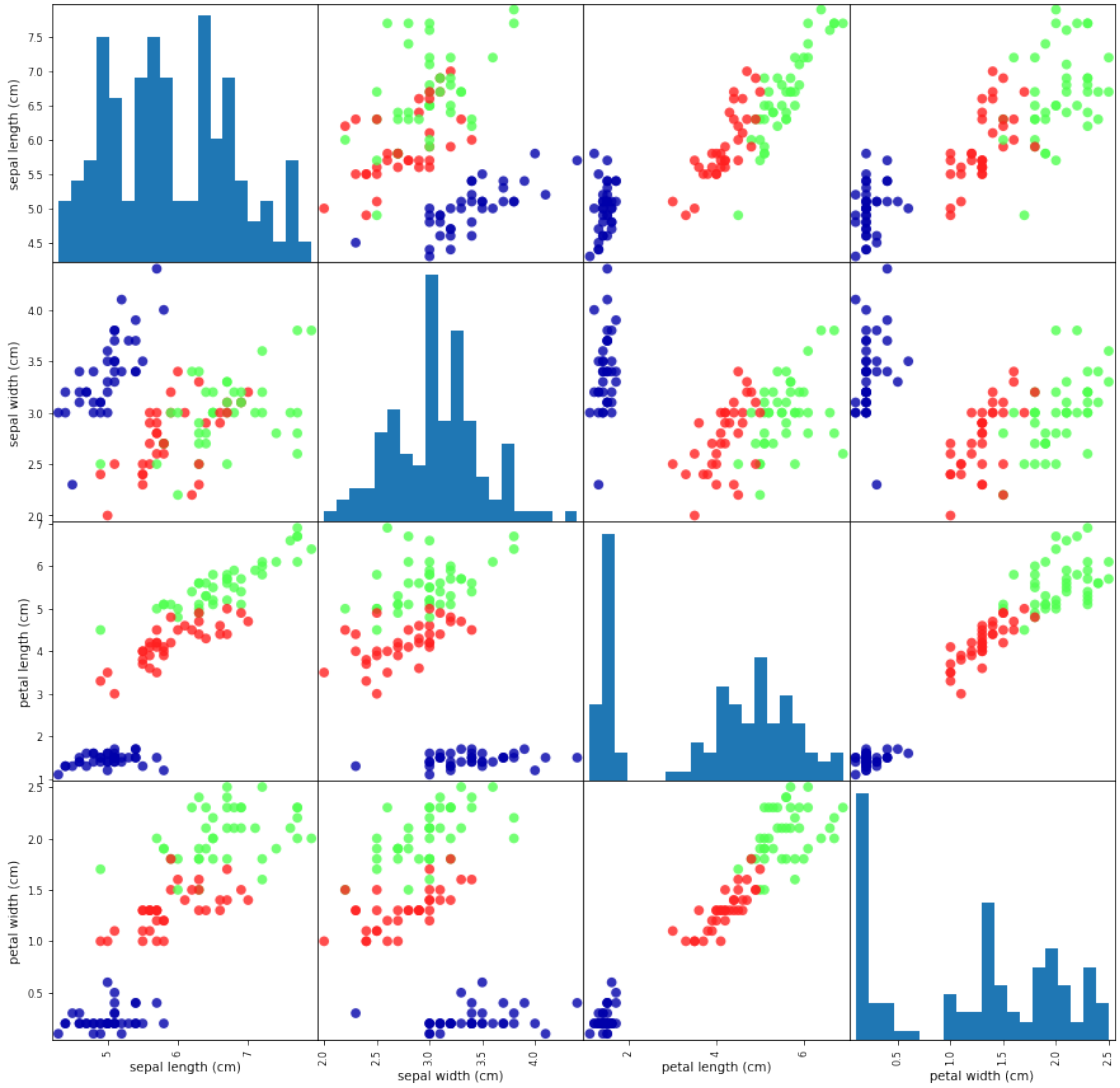
[24]:
```
print("X_train shape: {}".format(X_train.shape))
print("y_train shape: {}".format(y_train.shape))
```

```
X_train shape: (112, 4)
y_train shape: (112,)
```

[25]:
```
print("X_test shape: {}".format(X_test.shape))
print("y_test shape: {}".format(y_test.shape))
```

```
X_test shape: (38, 4)
y_test shape: (38,)
```

[26]:
```
#  X_train   DataFrame
#  iris_dataset.feature_names
iris_dataframe = pd.DataFrame(X_train, columns=iris_dataset.feature_names)
#  DataFrame     y_train
# grr = pd.scatter_matrix(iris_dataframe, c=y_train, figsize=(15, 15),␣
 ↪marker="o",
grr = pd.plotting.scatter_matrix(iris_dataframe, c=y_train, figsize=(15, 15),␣
 ↪marker="o",
                                 hist_kwds={'bins': 20}, s=60, alpha=.8,␣
 ↪cmap=mglearn.cm3)
```



1.3 1.7.4 k

[27]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=1)
```

[28]:
```
knn.fit(X_train, y_train)
```

[28]:
```
KNeighborsClassifier(n_neighbors=1)
```

1.4 1.7.5

[29]:
```
X_new = np.array([[5, 2.9, 1, 0.2]])
print("X_new.shape: {}".format(X_new.shape))
```

```
X_new.shape: (1, 4)
```

[30]:
```
prediction = knn.predict(X_new)
print("Prediction: {}".format(prediction))
print("Predicted target name: {}".format(
    iris_dataset['target_names'][prediction]
))
```

```
Prediction: [0]
Predicted target name: ['setosa']
```

1.5 1.7.6

[31]:
```
y_pred = knn.predict(X_test)
print("Test set predictions:\n {}".format(y_pred))
```

```
Test set predictions:
 [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 2]
```

[32]:
```
print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
```

```
Test set score: 0.97
```

2 1.8

[33]:
```
X_train, X_test, y_train, y_test = train_test_split(
    iris_dataset['data'], iris_dataset['target'], random_state=0)

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
print("Test set score: {:.2f}".format(knn.score(X_test, y_test)))
```

```
Test set score: 0.97
```