Lab1

October 10, 2025

1 Lab 1

1.1 Testing NumPy

[1]:
```
import numpy as np
x = np.array([[1,2,3], [4,5,6]])
x
```

[1]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

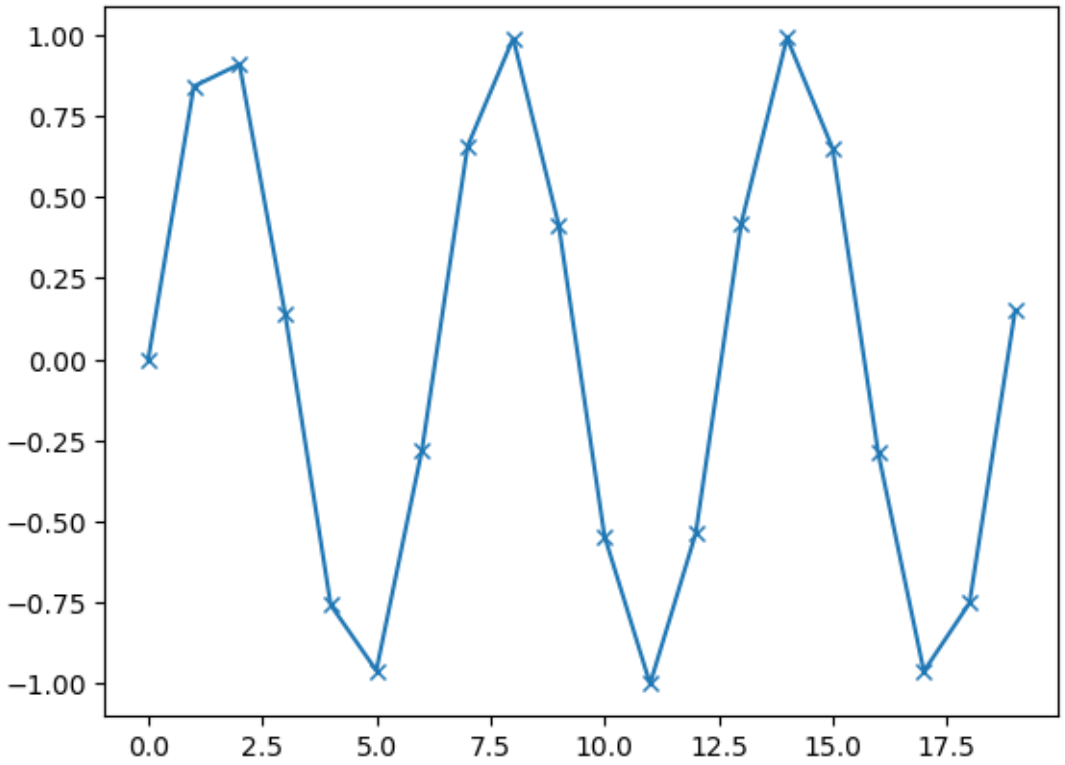
1.2 Testing NumPy and matplotlib

[2]:
```
%matplotlib inline
import matplotlib.pyplot as plt

x = np.arange(20)
y = np.sin(x)

plt.plot(x,y,marker="x")
```

[2]:
```
[<matplotlib.lines.Line2D at 0x7f3fb60408d0>]
```



1.3 Testing Iris

[3]:
```
from sklearn.datasets import load_iris
iris = load_iris()
```

[4]:
```
iris.keys()
```

[4]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
'filename', 'data_module'])
```

[5]:
```
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 …
```

[6]:
```
iris['target_names']
```

[6]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

[7]:
```
print(iris['target_names'])
```

```
['setosa' 'versicolor' 'virginica']
```

[8]:
```
iris['feature_names']
```

[8]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

[9]:
```
type(iris['data'])
```

[9]:
```
numpy.ndarray
```

[10]:
```
iris['data'].shape
```

[10]:
```
(150, 4)
```

[11]:
```
iris['data'][:3]
```

[11]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2]])
```

[12]:
```
type(iris['target'])
```

[12]:
```
numpy.ndarray
```

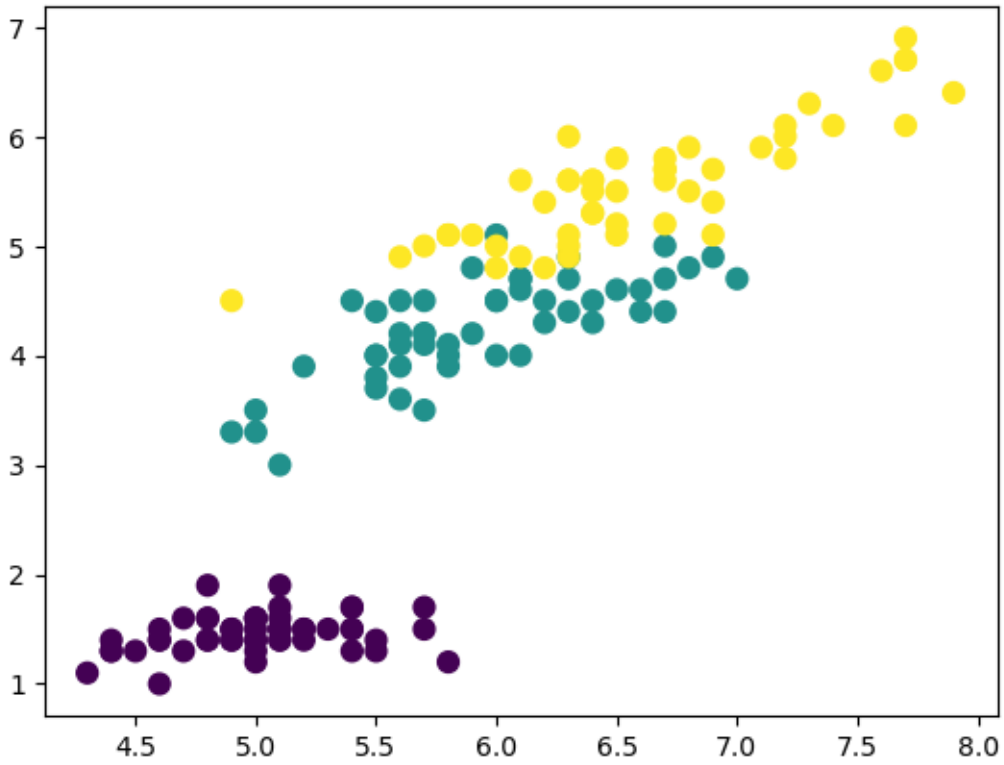
[13]:
```
iris['target'].shape
```

[13]:
```
(150,)
```

[14]:
```
X = iris['data']
y = iris['target']
```

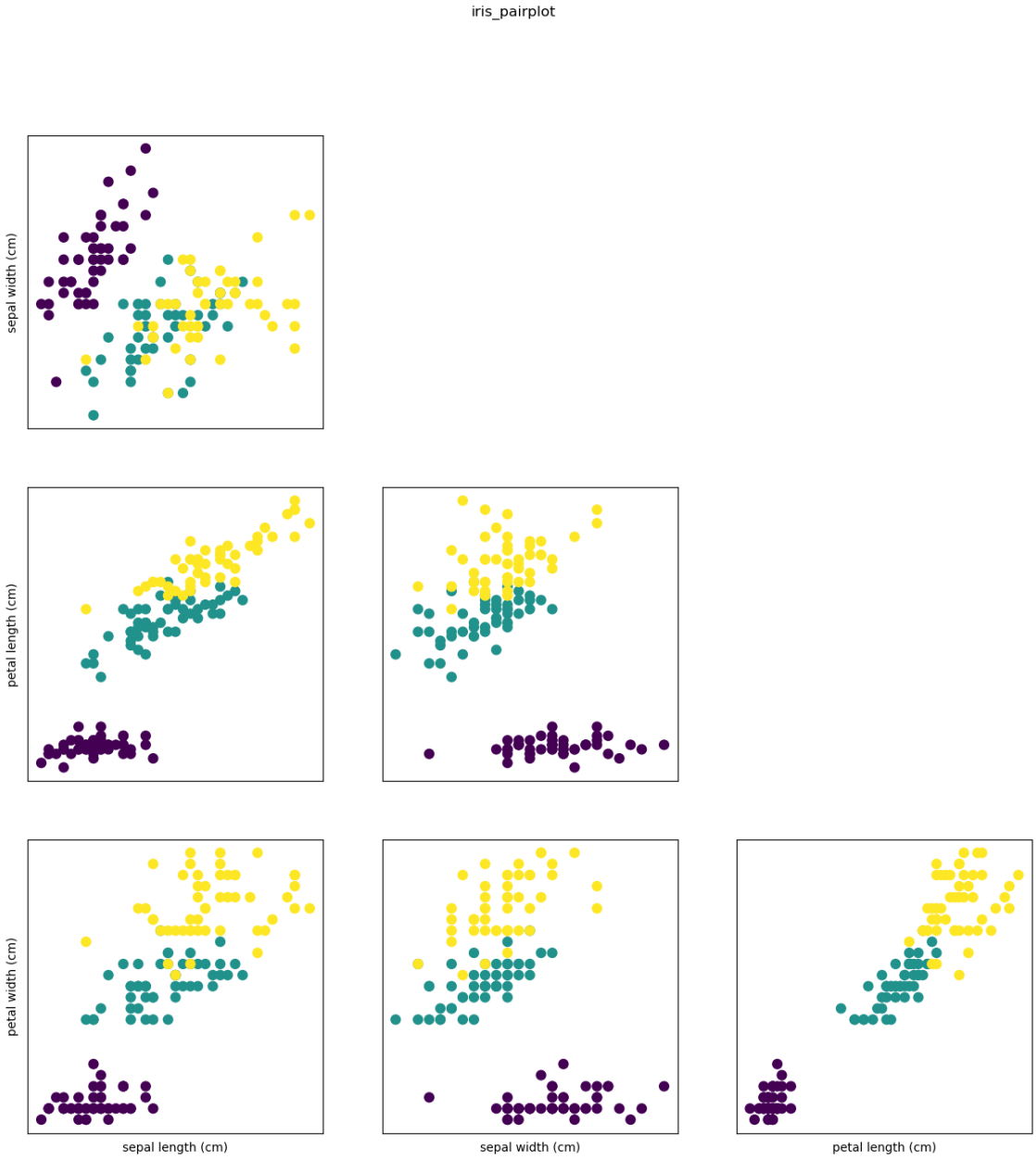
[15]:
```
plt.scatter(X[:,0], X[:,2], c=y, s=60)
```

[15]:
```
<matplotlib.collections.PathCollection at 0x7f3fa9d3c290>
```



[16]:
```
fig, ax = plt.subplots(3, 3, figsize=(15, 15))
plt.suptitle("iris_pairplot")

for i in range(3):
    for j in range(3):
        ax[i,j].scatter(X[:,j], X[:,i+1], c=y, s=60)
        ax[i,j].set_xticks(())
        ax[i,j].set_yticks(())
        if i == 2:
            ax[i,j].set_xlabel(iris['feature_names'][j])
        if j == 0:
            ax[i,j].set_ylabel(iris['feature_names'][i+1])
        if j > i:
            ax[i,j].set_visible(False)
```



1.4 Exercises

1. It is safe to say that features 0 (sepal length) and 2 (petal length) are as informative as any other pair of features, as there is a correlation that can lead to interpretations between the different irises. Each different group can be clearly distinguished from one another as seen when comparing sepal length and petal length, to sepal width and petal length.

2. The code hides comparisons that would plot repeated features in the check "if j > i". This avoids a "mirror" effect that would plot redundant features such as when both features are the same (e.g. plotting sepal width against sepal width) or when they have already been plotted

(e.g. plotting sepal width against petal length and then petal length against sepal width).