Module 3: Data Exploration

Dingyi Duan

University of San Diego

ADS 502

Module 2

3.1. Summary Statistics

Summary statistics are quantities, such as the mean and standard deviation, that capture various characteristics of a potentially large set of values with a single number or a small set of numbers. In this tutorial, we will use the Iris sample data, which contains information on 150 Iris flowers, 50 each from one of three Iris species: Setosa, Versicolour, and Virginica. Each flower is characterized by five attributes:

- sepal length in centimeters
- sepal width in centimeters
- petal length in centimeters
- petal width in centimeters
- class (Setosa, Versicolour, Virginica)

In this tutorial, you will learn how to:

- Load a CSV data file into a Pandas DataFrame object.
- Compute various summary statistics from the DataFrame.

To execute the sample program shown here, make sure you have installed the Pandas library (see Module 2).

1. First, you need to download the Iris dataset (http://archive.ics.uci.edu/ml/datasets/Iris) from the UCI machine learning repository.

Code: The following code uses Pandas to read the CSV file and store them in a DataFrame object named data. Next, it will display the first five rows of the data frame.

In [1]:
```
#Load the file
```

In [2]:
```
import pandas as pd

data = pd.read_csv('C:/Users/DDY/Desktop/2021-Spring-textbooks/ADS-502/Module2/We
data.columns = ['sepal length', 'sepal width', 'petal length', 'petal width', 'cl

data.head()
```

Out[2]:

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

2. For each quantitative attribute, calculate its average, standard deviation, minimum, and maximum values.

Code:

In [3]:
```
# For loop to print the mean, stdev, min and max for each col IF it contains nume
```

In [4]:
```
from pandas.api.types import is_numeric_dtype

for col in data.columns:
    if is_numeric_dtype(data[col]):
        print('%s:' % (col))
        print('\t Mean = %.2f' % data[col].mean())
        print('\t Standard deviation = %.2f' % data[col].std())
        print('\t Minimum = %.2f' % data[col].min())
        print('\t Maximum = %.2f' % data[col].max())
```

```
sepal length:
         Mean = 5.84
         Standard deviation = 0.83
         Minimum = 4.30
         Maximum = 7.90
sepal width:
         Mean = 3.05
         Standard deviation = 0.43
         Minimum = 2.00
         Maximum = 4.40
petal length:
         Mean = 3.76
         Standard deviation = 1.76
         Minimum = 1.00
         Maximum = 6.90
petal width:
         Mean = 1.20
         Standard deviation = 0.76
         Minimum = 0.10
         Maximum = 2.50
```

3. For the qualitative attribute (class), count the frequency for each of its distinct values.

Code:

In [5]:
```
# Count each iris class using value_counts()
```

In [6]:
```
data['class'].value_counts()
```

Out[6]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: class, dtype: int64
```

4. It is also possible to display the summary for all the attributes simultaneously in a table using the describe() function. If an attribute is quantitative, it will display its mean, standard deviation and various quantiles (including minimum, median, and maximum) values. If an attribute is qualitative, it will display its number of unique values and the top (most frequent) values.

Code:

In [7]:
```
# Very interesting way of using describe(), especially combining both qualitative
```

In [8]:
```
data.describe(include='all')
```

Out[8]:

	sepal length	sepal width	petal length	petal width	class
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	50
mean	5.843333	3.054000	3.758667	1.198667	NaN
std	0.828066	0.433594	1.764420	0.763161	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

Note that count refers to the number of non-missing values for each attribute.

5. For multivariate statistics, you can compute the covariance and correlation between pairs of attributes.

Code:

In [9]:
```
print('Covariance:')
data.cov()
```

```
Covariance:
```

Out[9]:

	sepal length	sepal width	petal length	petal width
sepal length	0.685694	-0.039268	1.273682	0.516904
sepal width	-0.039268	0.188004	-0.321713	-0.117981
petal length	1.273682	-0.321713	3.113179	1.296387
petal width	0.516904	-0.117981	1.296387	0.582414

In [10]:
```
# Correlation table can help justify for multicollinearity
```

In [11]:
```
print('Correlation:')
data.corr()
```

```
Correlation:
```

Out[11]:

	sepal length	sepal width	petal length	petal width
sepal length	1.000000	-0.109369	0.871754	0.817954
sepal width	-0.109369	1.000000	-0.420516	-0.356544
petal length	0.871754	-0.420516	1.000000	0.962757
petal width	0.817954	-0.356544	0.962757	1.000000

3.2. Data Visualization

Data visualization is the display of information in a graphic or tabular format. Successful visualization requires that the data (information) be converted into a visual format so that the characteristics of the data and the relationships among data items or attributes can be analyzed or reported.

In this tutorial, you will learn how to display the Iris data created in Section 3.1. To execute the sample program shown here, make sure you have installed the matplotlib library package (see Module 0 on how to install Python packages).

1. First, we will display the histogram for the sepal length attribute by discretizing it into 8 separate bins and counting the frequency for each bin.

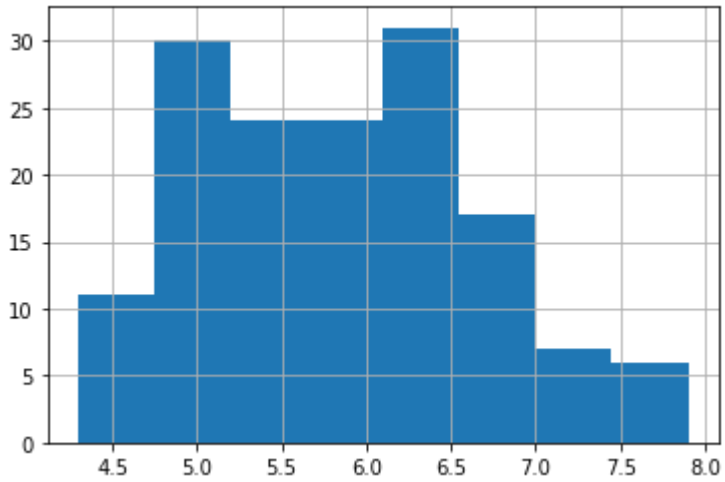
Code:

In [12]:
```
# Run histogram on sepal length using 8 bins
```

In [13]:
```
%matplotlib inline

data['sepal length'].hist(bins=8)
```

Out[13]: <AxesSubplot:>



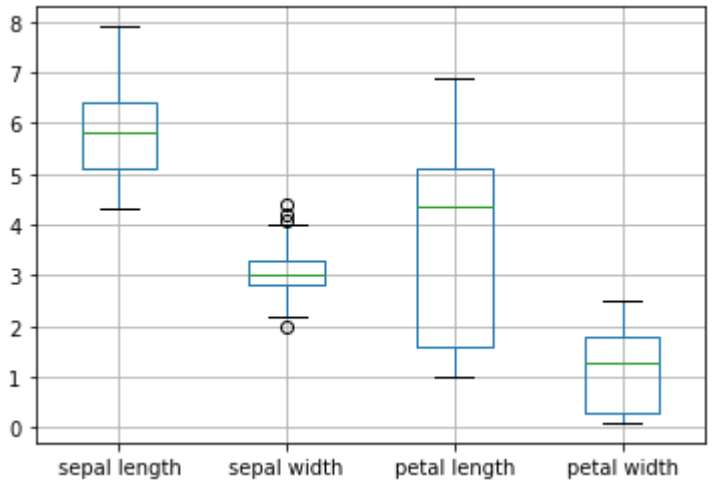
2. A boxplot can also be used to show the distribution of values for each attribute.

Code:

In [14]:
```
# Boxplot for all quantitative variables
```

In [15]:
```
data.boxplot()
```

Out[15]: <AxesSubplot:>

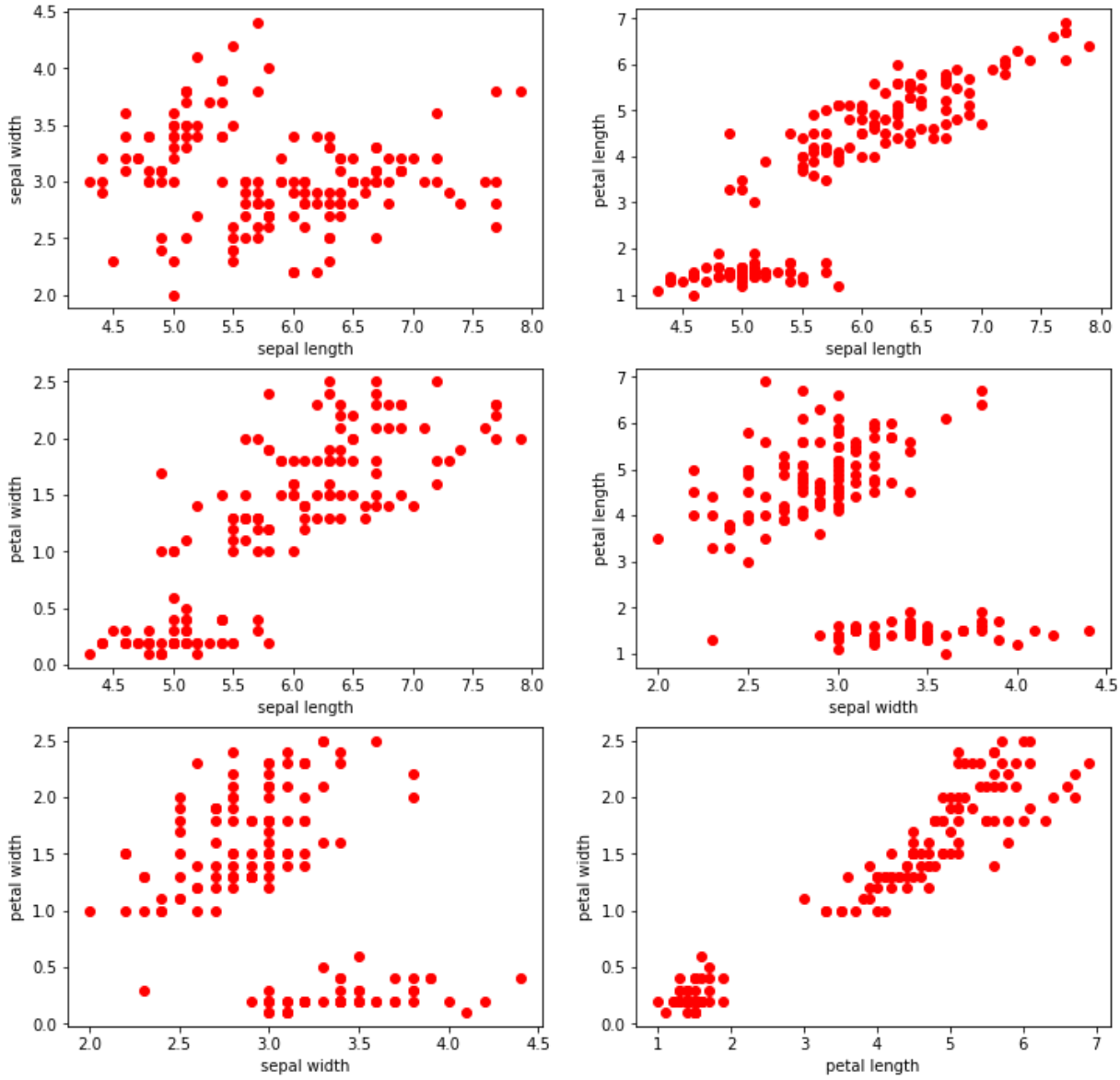


3. For each pair of attributes, we can use a scatter plot to visualize their joint distribution.

Code:

In [16]:
```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(3, 2, figsize=(12,12))
index = 0
for i in range(3):
    for j in range(i+1,4):
        ax1 = int(index/2)
        ax2 = index % 2
        axes[ax1][ax2].scatter(data[data.columns[i]], data[data.columns[j]], colo
        axes[ax1][ax2].set_xlabel(data.columns[i])
        axes[ax1][ax2].set_ylabel(data.columns[j])
        index = index + 1
```



4. Parallel coordinates can be used to display all the data points simultaneously. Parallel coordinates have one coordinate axis for each attribute, but the different axes are parallel to one

other instead of perpendicular, as is traditional. Furthermore, an object is represented as a line instead of as a point. In the example below, the distribution of values for each class can be identified in a separate color.

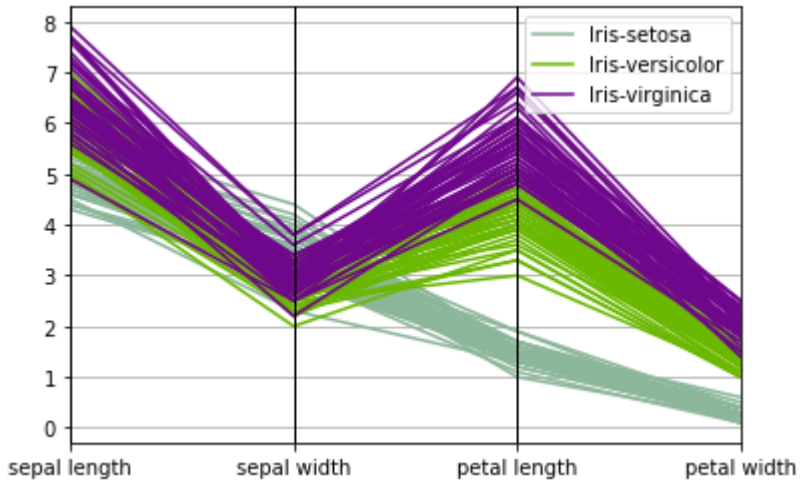
Code:

In [17]:
```
# Nice graph!
```

In [18]:
```
from pandas.plotting import parallel_coordinates
%matplotlib inline

parallel_coordinates(data, 'class')
```

Out[18]: <AxesSubplot:>



3.3. Summary

This tutorial presents several examples for data exploration and visualization using the Pandas and matplotlib library packages available in Python.

References:

1. Documentation on Pandas. https://pandas.pydata.org/ (https://pandas.pydata.org/)
2. Documentation on matplotlib. https://matplotlib.org/ (https://matplotlib.org/)
3. Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml] (http://archive.ics.uci.edu/ml%5D). Irvine, CA: University of California, School of Information and Computer Science.