



Lecture 3: Exploring Data

Data Exploration

- A preliminary exploration of the data to better understand its characteristics.
- Select the right tool for preprocessing or analysis
- Making use of humans' abilities to recognize patterns
- Related to the area of Exploratory Data Analysis (EDA)
 - Created by statistician John Tukey
 - Seminal book is Exploratory Data Analysis by Tukey
 - A nice online introduction can be found in Chapter 1 of the NIST Engineering Statistics Handbook

<http://www.itl.nist.gov/div898/handbook/index.htm>

Data Exploration methods

- Summary Statistics
- Visualization
- Clustering & Anomaly detection

Iris Sample Data set

- Can be obtained from the UCI Machine Learning Repository

<http://www.ics.uci.edu/~mlearn/MLRepository.html>

- Three flower types (classes):

- Setosa
- Virginica
- Versicolour

- Four attributes

- Sepal width and length
- Petal width and length



Virginica. Robert H. Mohlenbrock. USDA NRCS. 1995. Northeast wetland flora: Field office guide to plant species. Northeast National Technical Center, Chester, PA. Courtesy of USDA NRCS Wetland Science Institute.

A. Summary Statistics

- Summary statistics are numbers that summarize properties of the data
 - Frequency
 - Mean
 - Standard deviation
- Most summary statistics can be calculated in a single pass through the data

A.1 Frequency

- The frequency of an attribute value is the percentage of time the value occurs in the data set
 - ‘female’ occurs about 50% of the time.
- The mode of a an attribute is the most frequent attribute value
- The notions of frequency and mode are typically used with categorical data

A.2 Mean & Median

- The mean is the most common measure of the location of a set of points.
- Given a set of sample, mean & media is calculated as:

$$\text{mean}(x) = \bar{x} = \frac{1}{m} \sum_{i=1}^m x_i$$

$$\text{median}(x) = \begin{cases} x_{(r+1)} & \text{if } m \text{ is odd, i.e., } m = 2r + 1 \\ \frac{1}{2}(x_{(r)} + x_{(r+1)}) & \text{if } m \text{ is even, i.e., } m = 2r \end{cases}$$

- However, the mean is very sensitive to outliers.

A.3 Range and variance

- **Range** is the difference between the max and min
- The **variance** or standard deviation is the most common measure of the spread of a set of points.

$$\text{variance}(x) = s_x^2 = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})^2$$

- However, this is also sensitive to outliers, so that other measures are often used.

$$\text{AAD}(x) = \frac{1}{m} \sum_{i=1}^m |x_i - \bar{x}|$$

$$\text{MAD}(x) = \text{median}\left(\{|x_1 - \bar{x}|, \dots, |x_m - \bar{x}|\}\right)$$

$$\text{interquartile range}(x) = x_{75\%} - x_{25\%}$$

Quiz:

- Which measures are more sensitive to outliers?
Why?

$$\text{AAD}(x) = \frac{1}{m} \sum_{i=1}^m |x_i - \bar{x}|$$

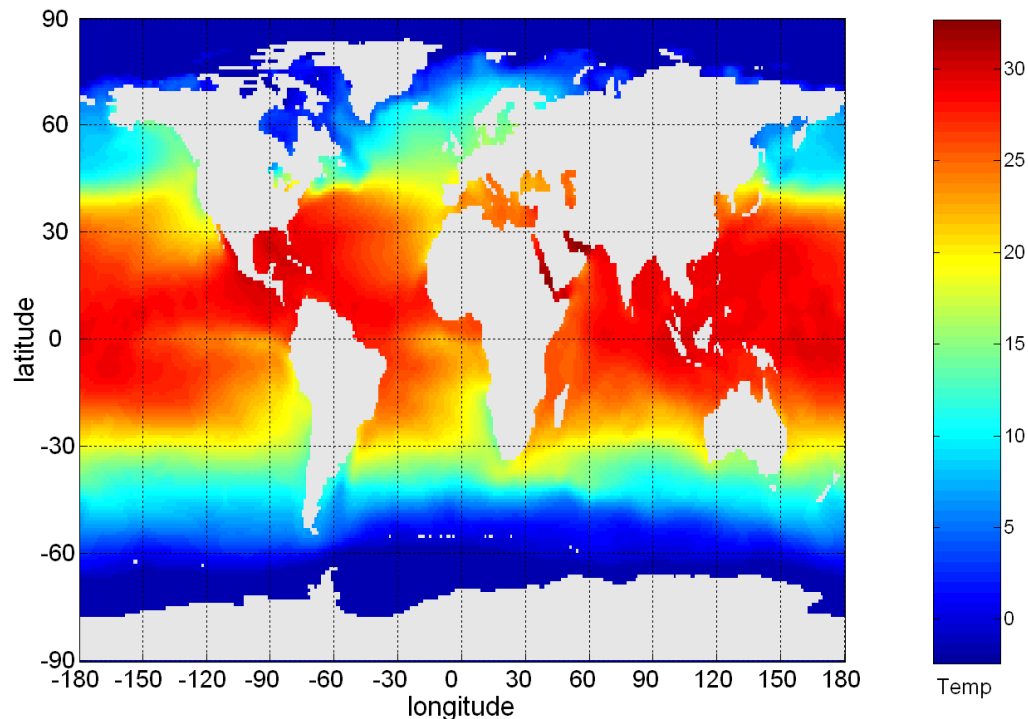
$$\text{variance}(x) = s_x^2 = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})^2$$

B. visualization

- Conversion of data into a visual or tabular format so that the characteristics of the data and the relationships among data items or attributes can be analyzed or reported
- Most powerful and appealing techniques in data exploration
 - Humans have a well developed ability to analyze large amounts of information that is presented visually
 - Can detect general patterns and trends
 - Can detect outliers and unusual patterns

Example: Sea Surface Temperature

- The following shows the Sea Surface Temperature (SST) for July 1982
 - Tens of thousands of data points are summarized in a single figure



Key factors: Representation

- Mapping of information to **visual format**
 - Points, lines, shapes, colors?
- Example:
 - Objects are often represented as points
 - Their attribute values can be represented as the position of the points or the characteristics of the points, e.g., color, size, and shape
 - If position is used, then the relationships of points, i.e., whether they form groups or a point is an outlier, is easily perceived.

Key factor 2: Arrangement

- Is the placement of visual elements within a display
- Can make a large difference in how easy it is to understand the data
- Example:

	1	2	3	4	5	6
1	0	1	0	1	1	0
2	1	0	1	0	0	1
3	0	1	0	1	1	0
4	1	0	1	0	0	1
5	0	1	0	1	1	0
6	1	0	1	0	0	1
7	0	1	0	1	1	0
8	1	0	1	0	0	1
9	0	1	0	1	1	0

	6	1	3	2	5	4
4	1	1	1	0	0	0
2	1	1	1	0	0	0
6	1	1	1	0	0	0
8	1	1	1	0	0	0
5	0	0	0	1	1	1
3	0	0	0	1	1	1
9	0	0	0	1	1	1
1	0	0	0	1	1	1
7	0	0	0	1	1	1

Key factor 3: Selection

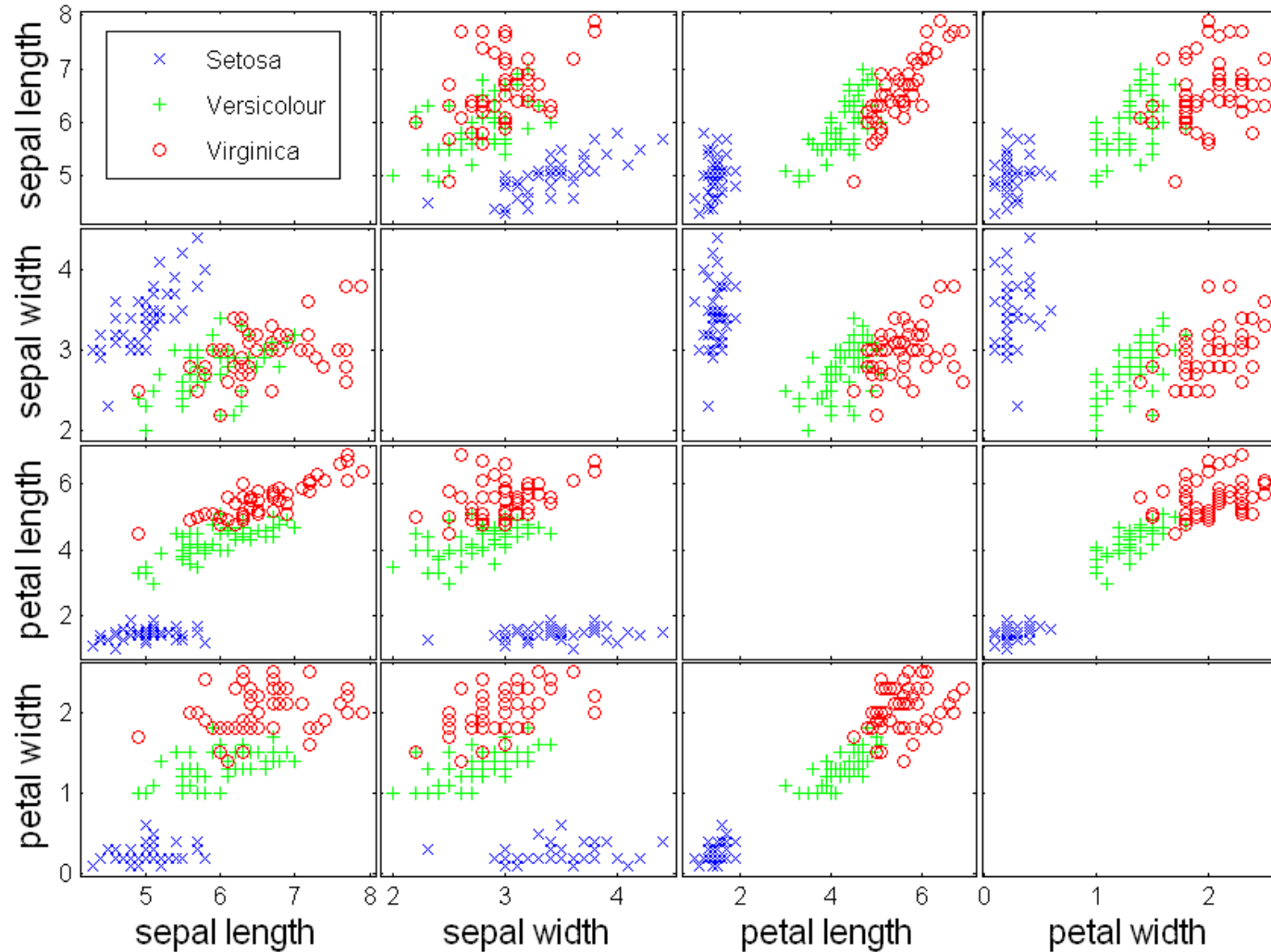
- elimination of certain objects and attributes
- Choosing a subset of attributes
 - Dimensionality reduction
 - Pairs of attributes can be considered
- Selection may also involve choosing a subset of objects -- sampling
 - A region of the screen can only show so many points
 - Can sample, but want to preserve points in sparse areas

Visualization Techniques: Scatter Plots

- Scatter plots

- Attributes values determine the position
- Two-dimensional scatter plots most common, but can have three-dimensional scatter plots
- Often additional attributes can be displayed by using the size, shape, and color of the markers that represent the objects
- It is useful to have arrays of scatter plots can compactly summarize the relationships of several pairs of attributes

Scatter Plot Array of Iris Attributes

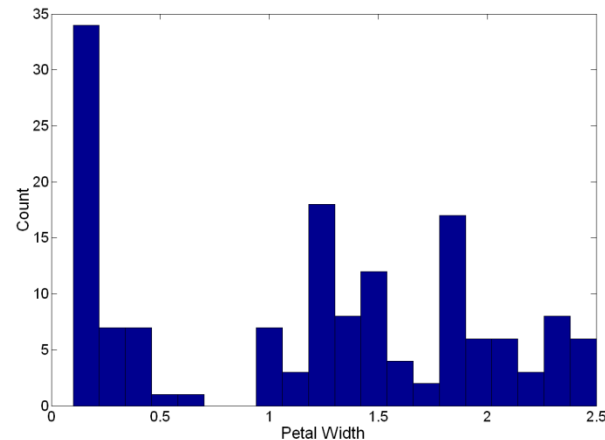
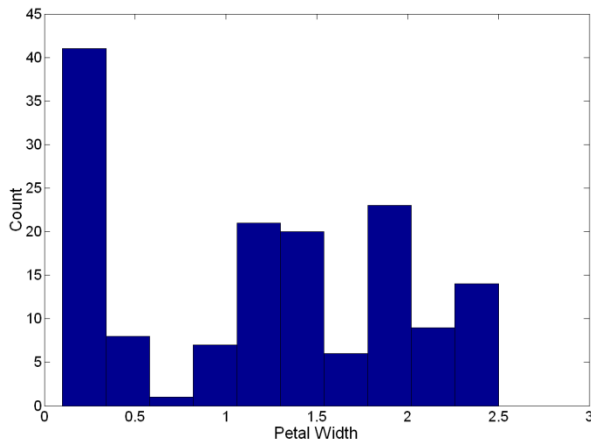


Visualization Techniques: Histograms

● Histogram

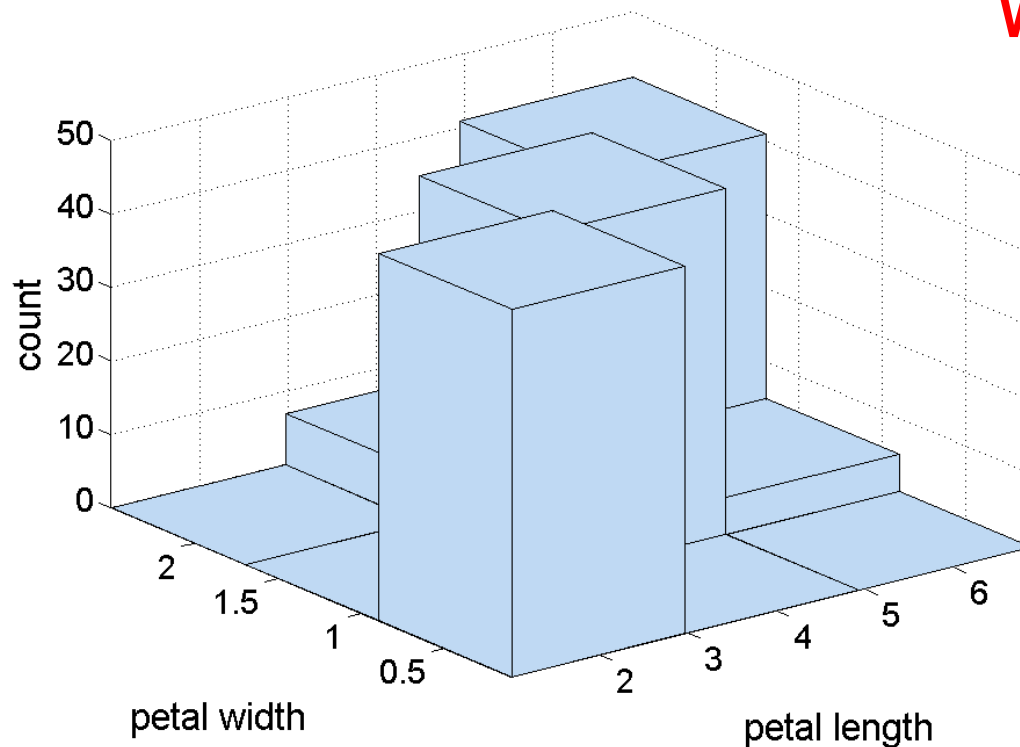
- Usually shows the distribution of values of a single variable
- Divide the values into bins and show a bar plot of the number of objects in each bin.
- The height of each bar indicates the number of objects
- Shape of histogram depends on the number of bins

● Example: Petal Width (10 and 20 bins, respectively)



Two-Dimensional Histograms

- Show the joint distribution of the values of two attributes
- Example: petal width and petal length



What does this tell us?

Python: Scatter & Histogram

● Step 1/3: Load Dataset

```
%matplotlib inline  
  
from sklearn.datasets import load_iris  
  
import matplotlib.pyplot as plt  
  
from sklearn import datasets  
  
iris = load_iris()  
  
print(iris.target_names)  
print(iris.data.shape)
```

```
['setosa' 'versicolor' 'virginica']  
(150, 4)
```

Python: Scatter & Histogram

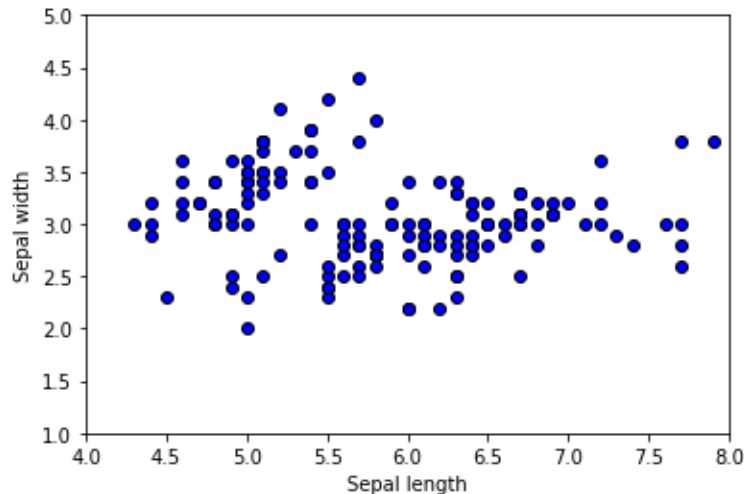
● Step 2/3: Scatter

```
X=iris.data

# Plot the training points
plt.scatter(X[:, 0], X[:, 1], c="blue", cmap=plt.cm.Set1,
            edgecolor='k')
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

plt.xlim(4, 8)
plt.ylim(1, 5)
```

(1.0, 5.0)



Python: Scatter & Histogram

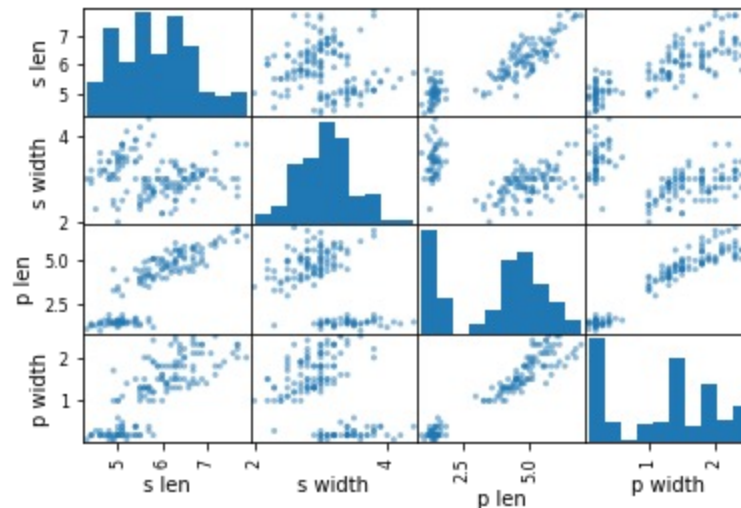
● Step 2/3: Scatter Matrix

```
import pandas as pd

iris = datasets.load_iris()
X_iris = iris.data
Y_iris = iris.target

X_df=pd.DataFrame(X_iris,columns=['s len', 's width', 'p len', 'p width'])

pd.plotting.scatter_matrix(X_df)
```



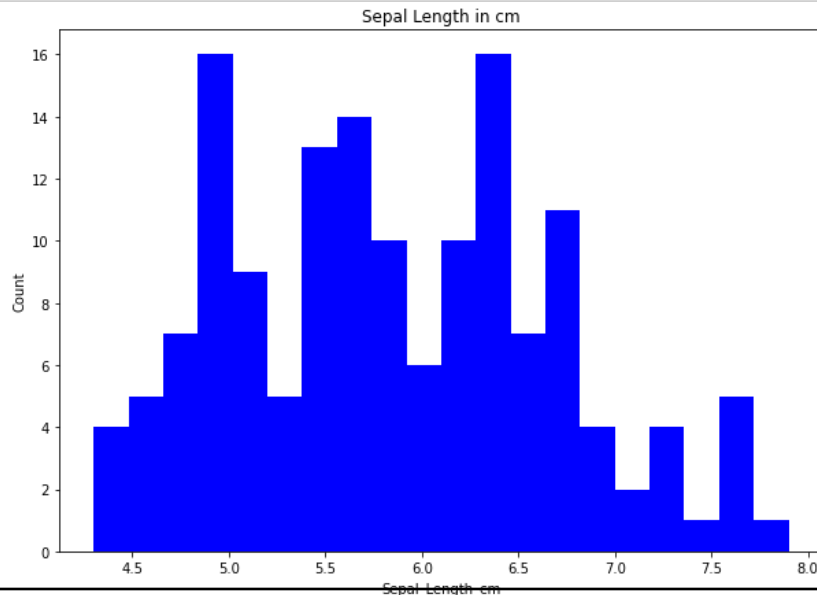
Python: Scatter & Histogram

- Step 3/3: histogram

```
# histogram for sepal length
plt.figure(figsize = (10, 7))

x = iris.data[:,0]

plt.hist(x, bins = 20, color = "blue")
plt.title("Sepal Length in cm")
plt.xlabel("Sepal_Length_cm")
plt.ylabel("Count")
```



Visualization Techniques: Box Plots

- Percentiles?

- For continuous data, the notion of a percentile is more useful.
- For instance, the 50th percentile is the value p such that 50% of all values are less than p .

Visualization Techniques: Box Plots

● Box Plots

- Invented by J. Tukey
- Another way of displaying the distribution of data
- Following figure shows the basic part of a box plot

← outlier

← 10th percentile

← 75th percentile

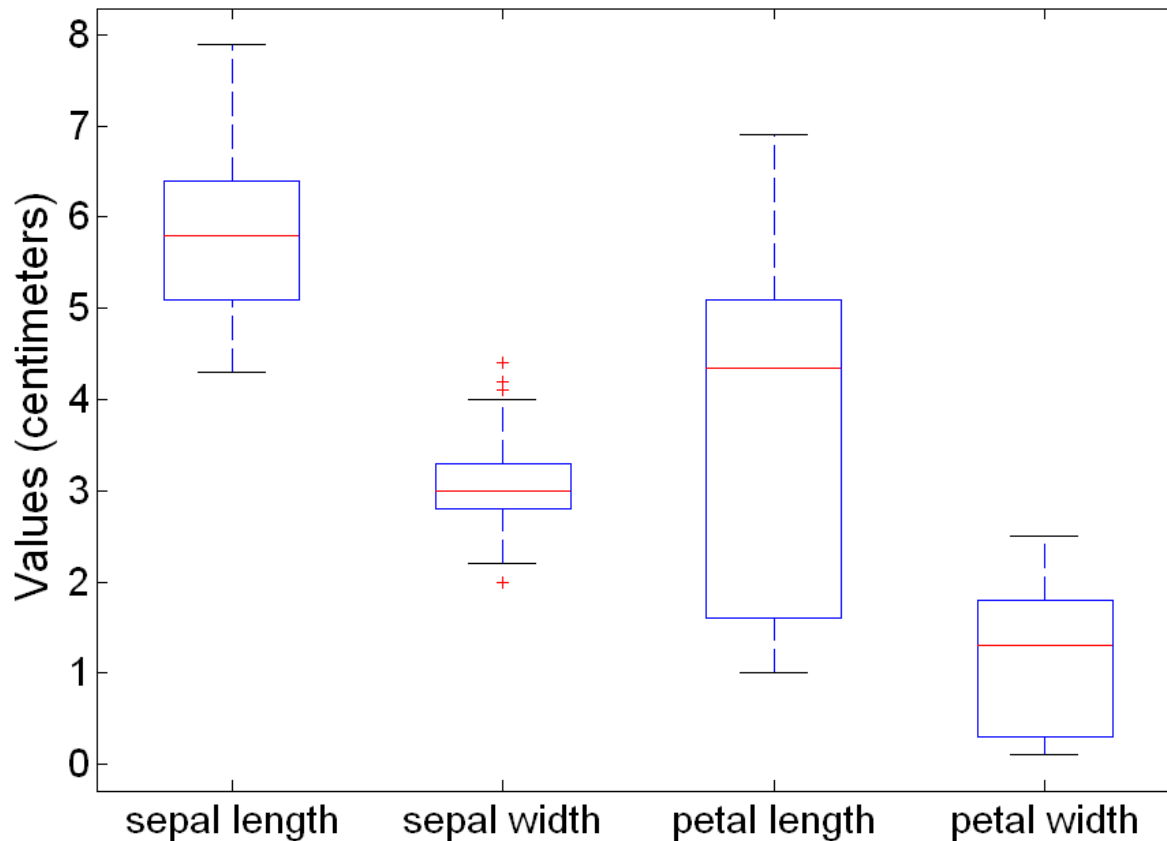
← 50th percentile

← 25th percentile

← 10th percentile

Example of Box Plots

- Box plots can be used to compare attributes



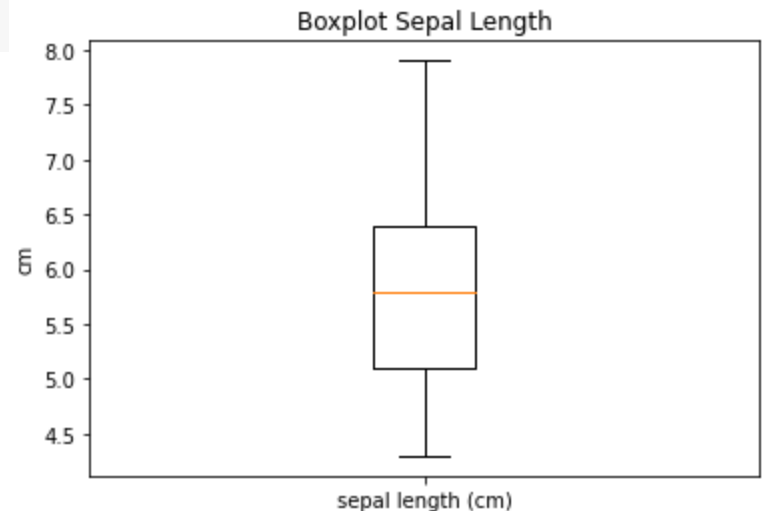
Python: Box Plots

- Box Plotting a single feature

```
# BOXPLOT

from sklearn import datasets
import matplotlib.pyplot as plt
iris = datasets.load_iris()
X_iris = iris.data
X_sepal = X_iris[:, 0]

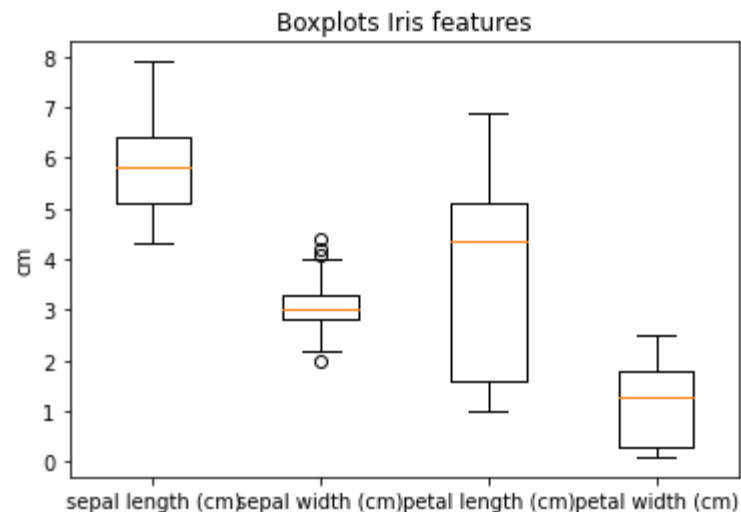
plt.boxplot(X_sepal, labels=[iris.feature_names[0]])
plt.title("Boxplot Sepal Length")
plt.ylabel("cm")
plt.show
```



Python: Box Plots

- Box Plotting multiple features

```
plt.boxplot(X_iris, labels=[iris.feature_names[0], iris.feature_names[1], iris.feature_names[2], iris.feature_names[3]],  
            plt.title("Boxplots Iris features")  
            plt.ylabel("cm")  
            plt.show
```



Visualization Techniques: Contour Plots

- Contour plots

- Useful when a continuous attribute is measured on a spatial grid
- They partition the plane into regions of similar values
- The contour lines that form the boundaries of these regions connect points with equal values
- The most common example is contour maps of elevation
- Can also display temperature, rainfall, air pressure, etc.
 - ◆ An example for Sea Surface Temperature (SST) is provided on the next slide

Python Reference

● Matplotlib.pyplot.contour

```
contour([X, Y,] Z, [levels], **kwargs)
```



`contour` and `contourf` draw contour lines and filled contours, respectively. Except as noted, function signatures and return values are the same for both versions.

Parameters:

X, Y : array-like, optional

The coordinates of the values in `Z`.

`X` and `Y` must both be 2-D with the same shape as `Z` (e.g. created via `numpy.meshgrid`), or they must both be 1-D such that `len(X) == M` is the number of columns in `Z` and `len(Y) == N` is the number of rows in `Z`.

If not given, they are assumed to be integer indices, i.e. `X = range(M)`, `Y = range(N)`.

Z : array-like(N, M)

The height values over which the contour is drawn.

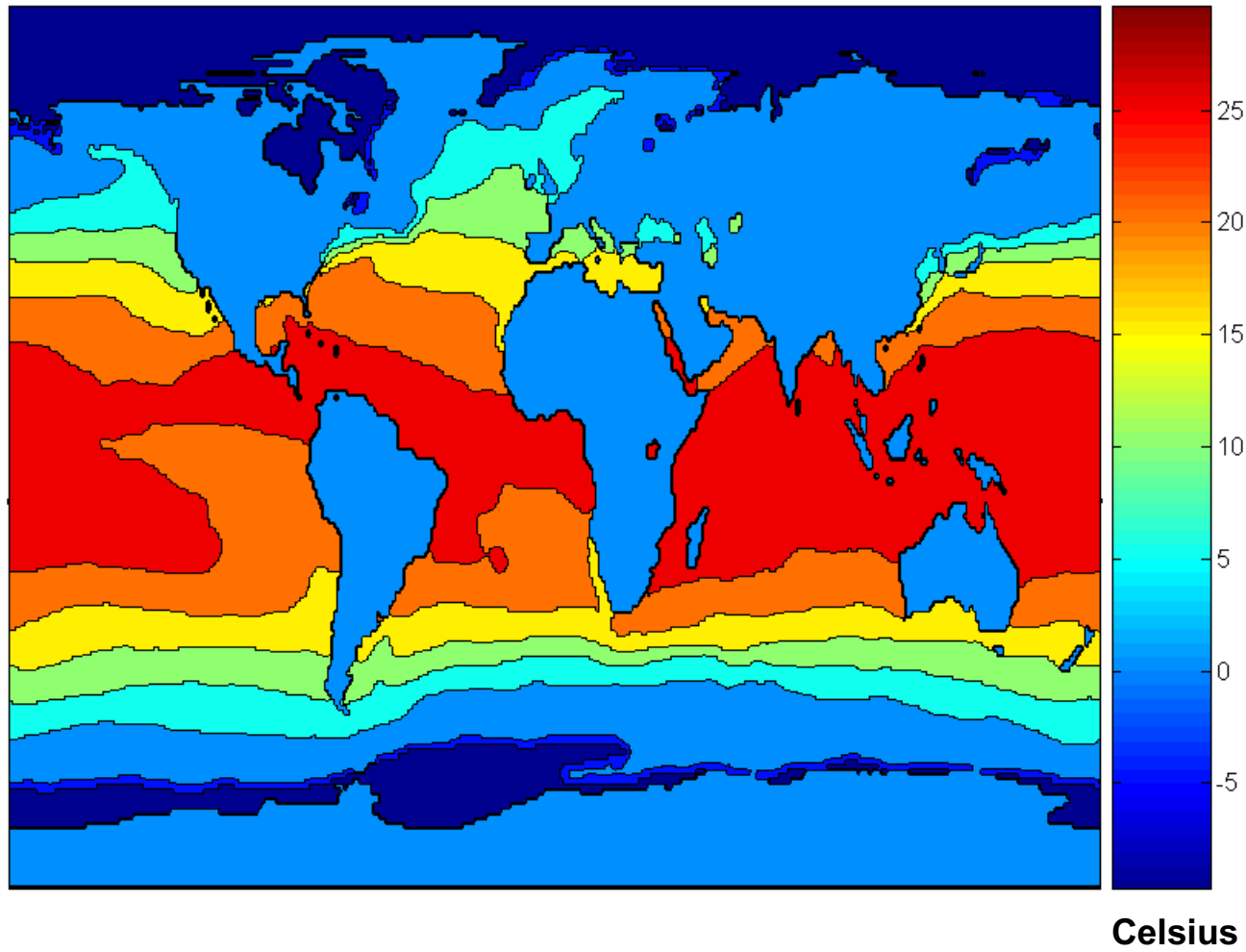
levels : int or array-like, optional

Determines the number and positions of the contour lines / regions.

If an int `n`, use `n` data intervals; i.e. draw `n+1` contour lines. The level heights are automatically chosen.

If array-like, draw contour lines at the specified levels. The values must be in increasing order.

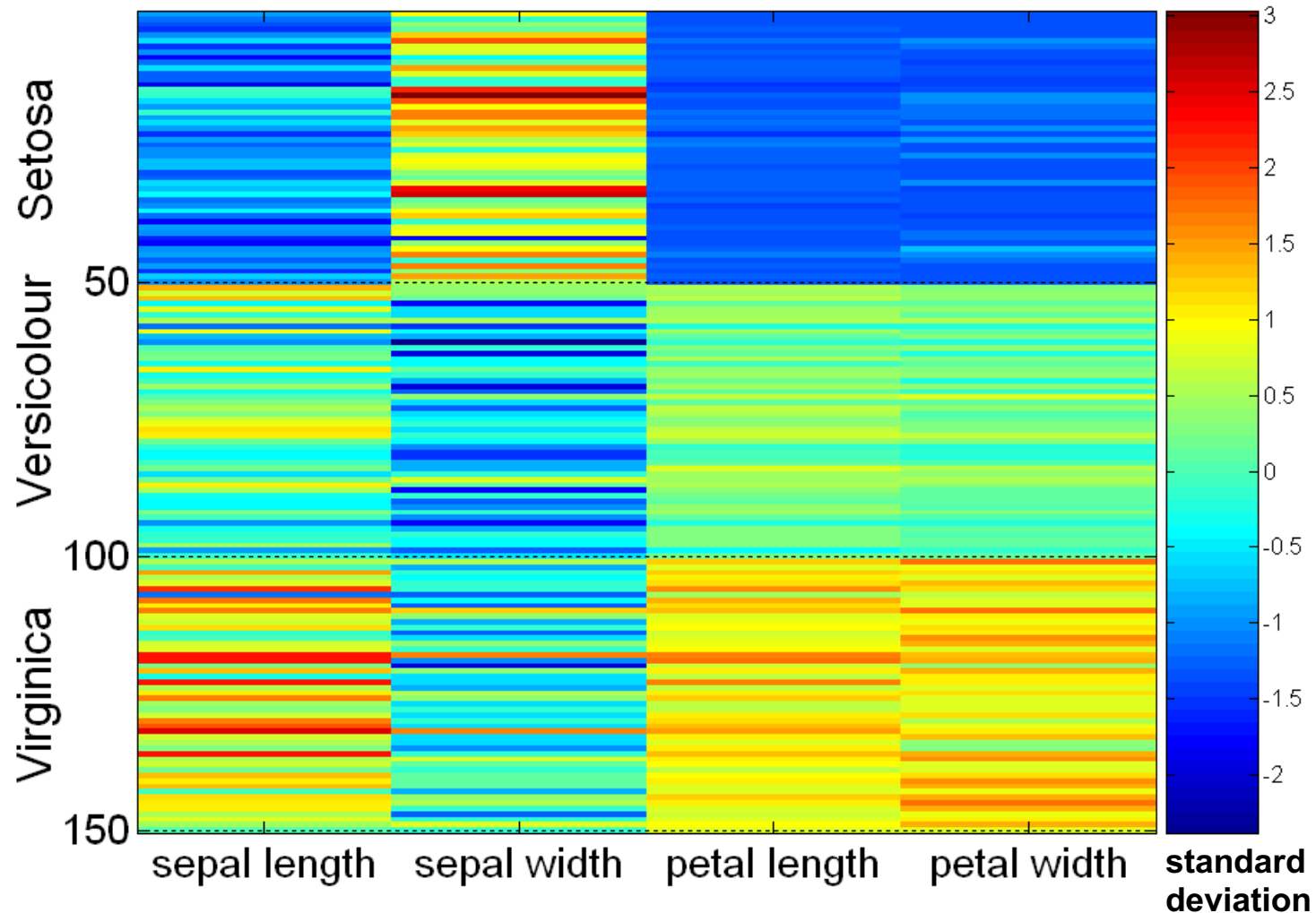
Contour Plot Example: SST Dec, 1998



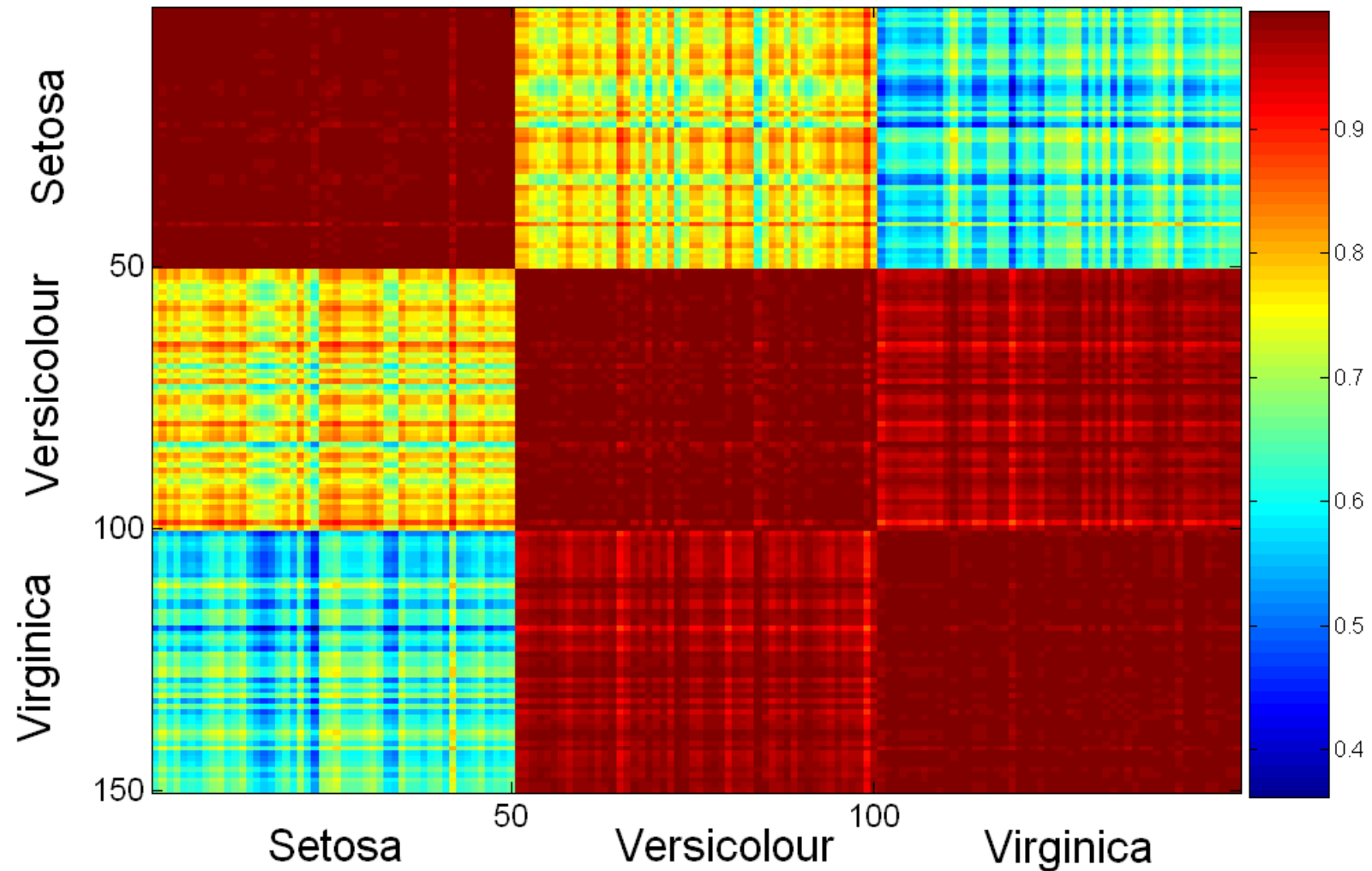
Visualization Techniques: Matrix Plots

- Matrix plots
 - Can plot the data matrix
 - This can be useful when objects are sorted according to class

Visualization of the Iris Data Matrix



Visualization of the Iris Correlation Matrix



Python Reference

● Matplotlib.pyplot.matshow

```
matplotlib.pyplot.matshow(A, fignum=None, **kwargs)
```

[\[source\]](#)

Display an array as a matrix in a new figure window.

The origin is set at the upper left hand corner and rows (first dimension of the array) are displayed horizontally. The aspect ratio of the figure window is that of the array, unless this would make an excessively short or narrow figure.

Tick labels for the axis are placed on top.

Parameters:

A : array-like(M, N)

The matrix to be displayed.

fignum : None or int or False

If *None*, create a new figure window with automatic numbering.

If a nonzero integer, draw into the figure with the given number (create it if it does not exist).

If 0, use the current axes (or create one if it does not exist).

Note

Because of how `Axes.matshow` tries to set the figure aspect ratio to be the one of the array, strange things may happen if you reuse an existing figure.

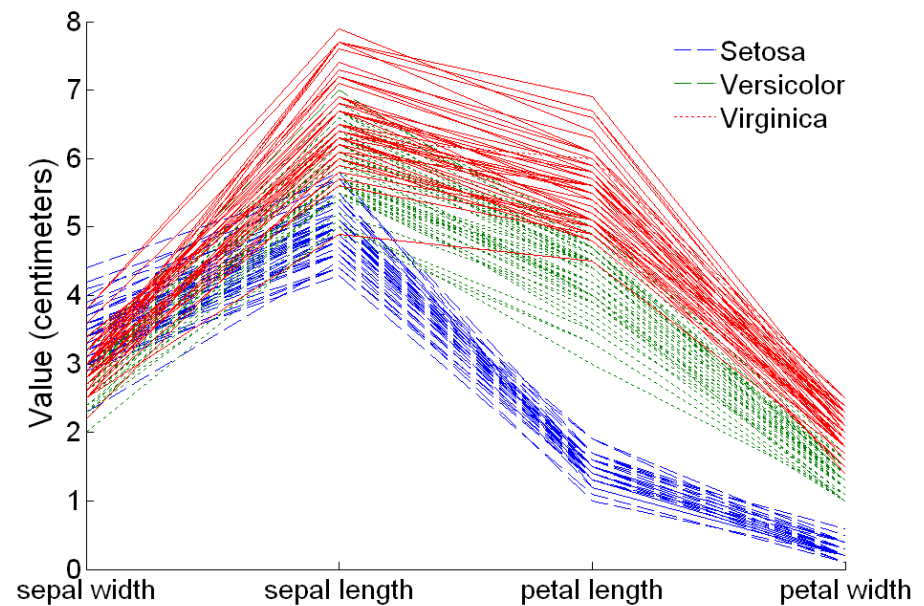
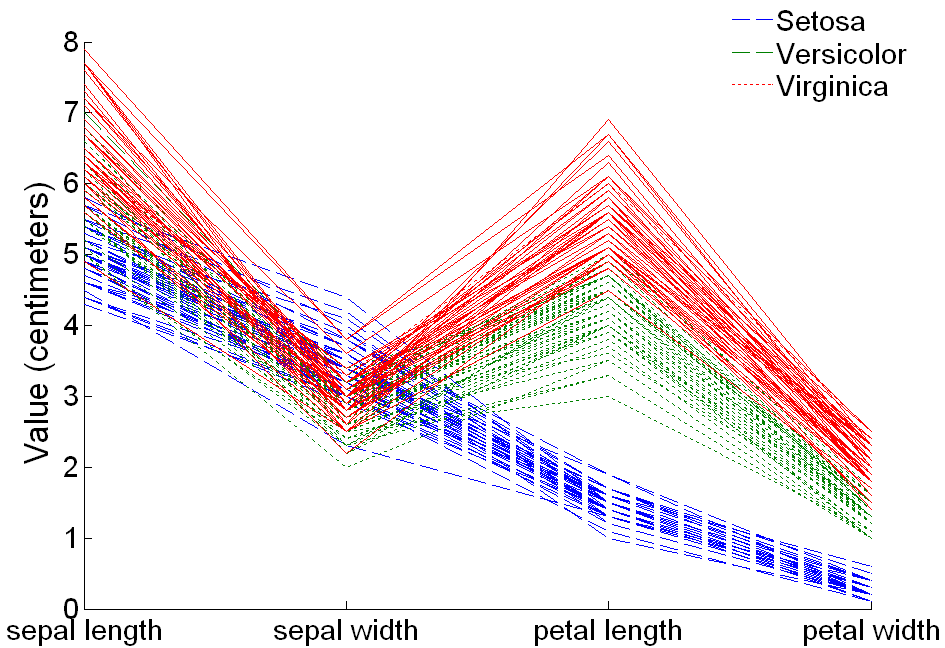
Returns:

image : `AxesImage`

Other Parameters:

****kwargs** : `imshow` arguments

Parallel Coordinates Plots for Iris Data



Visualization Techniques: Parallel Coordinates

● Parallel Coordinates

- Used to plot the attribute values of high-dimensional data
- Instead of using perpendicular axes, use a set of parallel axes
- The attribute values of each object are plotted as a point on each corresponding coordinate axis and the points are connected by a line
- Thus, each object is represented as a line
- Often, the lines representing a distinct class of objects group together, at least for some attributes
- Ordering of attributes is important in seeing such groupings

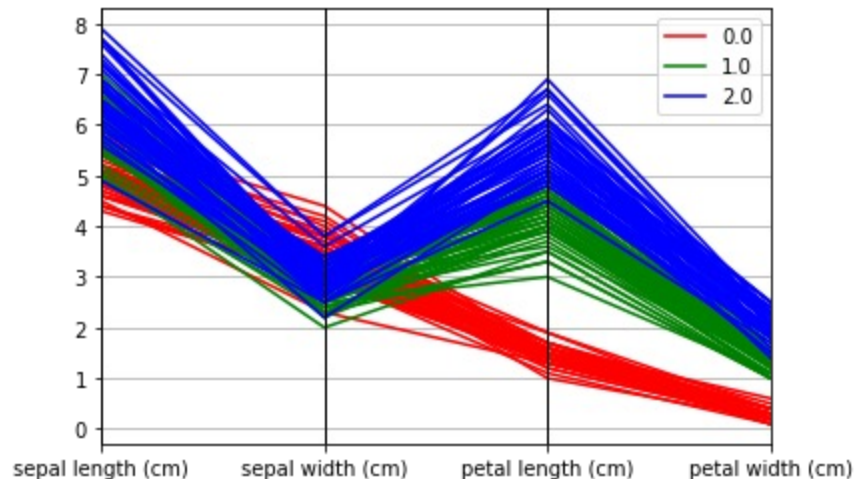
Python Reference

- pandas.plotting.parallel_coordinates

```
import pandas as pd

iris = load_iris()
iris_data = np.hstack((iris.data, iris.target.reshape(-1,1)))

iris_df = pd.DataFrame(data=iris_data, columns=iris.feature_names+ ["classes"])
iris_df.head()
pd.plotting.parallel_coordinates(iris_df, "classes", color=["red", "green", "blue"]);
```



Recap

- Statistics
- Visualization
 - Factors
 - techniques