Laboratory 2

In this laboratory we will introduce the IRIS dataset. We will see how we can load and visualize the dataset, and compute some statistics of the data.

The IRIS Dataset

The dataset was introduced in Fisher, R.A. “The use of multiple measurements in taxonomic problems”, Annual Eugenics, 7, Part II, 179-188 (1936). The dataset contains information on 150 samples (instances) of iris flowers belonging to 3 different families (classes): iris setosa, iris versicolor and iris virginica. There are 50 samples for each class. For each sample, the dataset provides 4 attributes: sepal length (cm), sepal width (cm), petal length (cm), petal width (cm).

Loading the dataset

We want to load the dataset into a numpy 2-dimensional array of shape (4×150). Each row of the data matrix will correspond to an attribute, whereas each column will represent a sample. We will also construct a 1-dimensional array of class labels. Iris setosa will be indicated with value 0, iris versicolor with value 1 and iris virginica with value 2.

The dataset is provided as a comma-separated values (csv) file, where each line represents a data point. Each line contains the four attributes and the family name, for example:

```
5.1,3.5,1.4,0.2,Iris-setosa
```

The dataset can be found in `Solution/iris.csv`. Write a `load` function that, given the dataset filename, returns the (4×150) data matrix, and the corresponding 1-dimensional array of size 150 containing the class label. Suggestion: for each line, create a 4×1 vector with the attribute values. Store all vectors in a list, and then concatenate the vectors.

Visualizing the dataset

We want to visualize the distribution of the different attributes for the different classes.

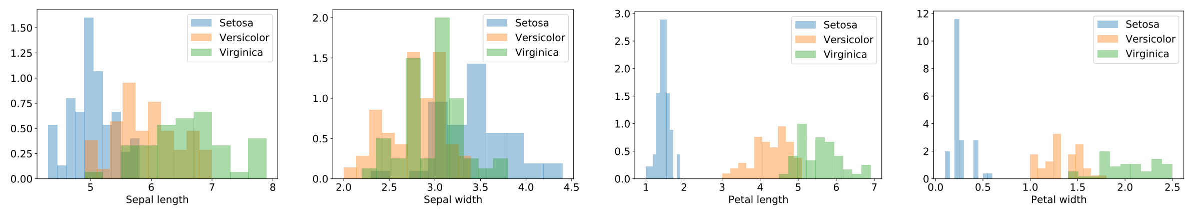
- For each attribute, plot the corresponding histogram for each class. You can use `matplotlib.pyplot.hist`, providing a 1-dimensional numpy array with the target data. Try with different number of bins.

  You can normalize the histogram setting the parameter `density = True`. To create a new figure, you can use `matplotlib.pyplot.figure`. To visualize the current figure(s), use `matplotlib.pyplot.show`.

  NOTE: `matplotlib.pyplot.show` shows the current image(s) in different windows, and blocks the execution until the windows are closed.

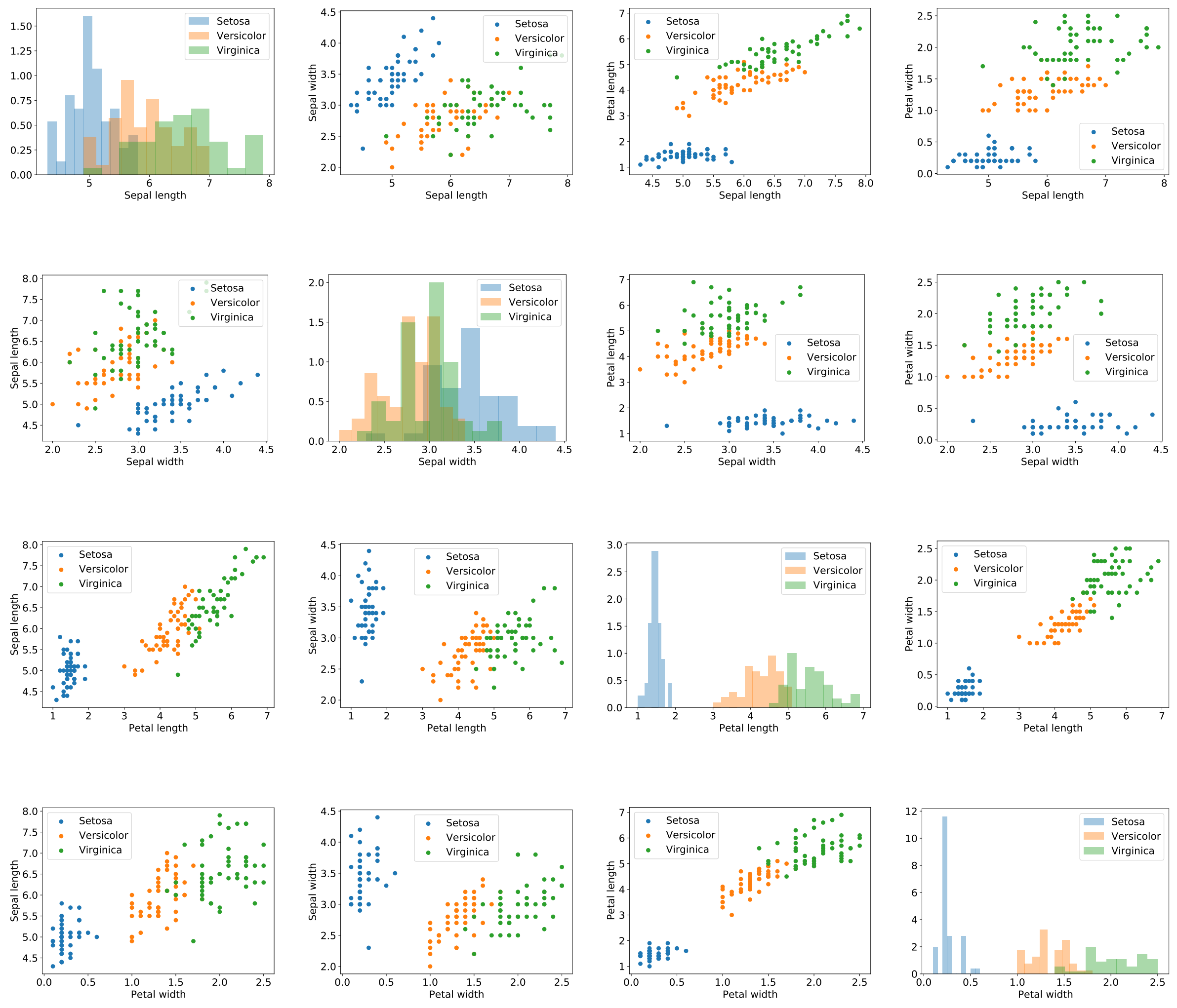
  Suggestion: start extracting from the data matrix the parts corresponding to the different classes, then proceed with the plots. If `D` is the data matrix, and `L` is the label vector (1-dimensional), we can build a mask for each class and use it to filter the columns of `D`:

  ```
  M0 = (L == 0)
  D0 = D[:, M0] # or equivalently D0 = D[:, L==0]
  ```



We can observe that there is large overlap for the first two attributes, whereas values for the third and fourth attributes of iris-setosa are well separated from those of the other two flower families

- We now consider pairs of values. Visualize the scatter plots of the different attribute pairs for each class.



Statistics

We now look at how to compute simple statistics and transformations of the data.

Data mean

We can compute the dataset mean using a `for` loop:

```
mu = 0
for i in range(D.shape[1]):
    mu = mu + D[:, i:i+1]

mu = mu / float(D.shape[1])
```

The `for` loop is, in general, slow. `Numpy` allows computing the mean of an array through the method `.mean`. The method allows specifying an axis — for 2-D arrays, `axis = 0` allows computing the mean over rows, whereas `axis = 1` allows computing the means over columns. We can thus compute the dataset mean as

```
mu = D.mean(1)
```

Pay attention to the shape of `mu`: it's a 1-D array.
We now exploit broadcasting to *center the data*, i.e. to remove the mean from all points:

```
DC = D - D.mean(1).reshape((D.shape[0], 1))
```

Notice that we want the mean to be a column vector, so we first reshape it.
Suggestion: we will often have to reshape 1-D vectors as column or row vectors. Write a function `mcol` and a function `mrow` that implements the reshaping

You can try plotting again the centered data.