Assignment

Homework0

Arystan Tatishev

A CS5785 Homework Assignment

CORNELL TECH

May 20, 2024

Problem 1

1 IRIS FLOWERS

In 1935, Edgar Anderson went to his favourite pasture and recorded the length and width of the sepals and petals on several flowers in the field. For whatever reason, this dataset became one of the oldest and most well-known "sanity-check" datasets around, being cited by countless papers. This class continues this time-honored tradition by using Iris Flowers to sanity-check your Python environment and plotting libraries.

1. Find and download the Iris Flowers dataset from the UC Irvine Machine Learning datasets archive at http://archive.ics.uci.edu/ml/datasets.php Hint: The iris.names file describes the structure of the dataset. How many features/attributes are there per sample? How many different species are there, and how many samples of each species did Anderson record?

2. Figure out how to parse the dataset you downloaded. Load the samples into an $N \times p$ array, where N is the number of samples and p is the number of attributes per sample. Aditionally, create a N-dimensional vector containing each sample's label (species).

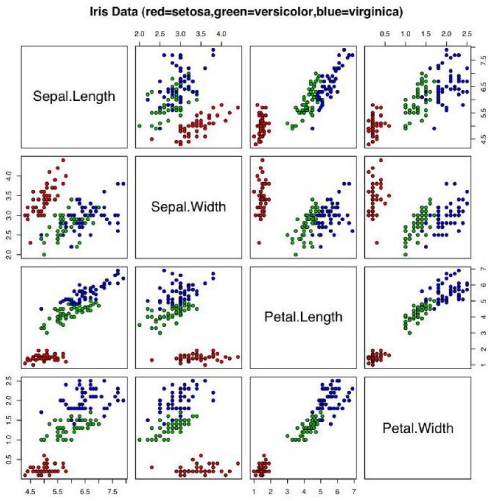
   Hint: Python has a built-in CSV parser in the csv library, or you can use the "string" .split (. .) method.

   Hint 2: Here is some code that prints each line in a file:

```
for line in open("/path/to/filename.txt"):
    print "Line contains: "+line
```

3. To visualize this dataset, we would have to build a p-dimensional scatterplot. Unfortunately, we only have 2D displays so we must reduce the dataset's dimensionality. The easiest way to view the set is to plot two attributes of the data against one another and repeat for each pair of attributes.

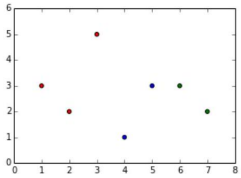
   Create every possible scatterplot from all pairs of two attributes. (For example, one scatterplot would graph petal length vs sepal width, another would graph petal length vs. sepal length, and so on). Within each scatterplot, the color of each dot should correspond with the sample species. Ideally, we're looking for something like this figure from Wikipedia:



But your results do not have to be this ornate. Presenting six separate figures in your report is certainly fine. Be sure to include the source code for all plots!

Hint: This is one way to draw a scatterplot. Use whatever works for you.

```
from matplotlib import pyplot as plt
import numpy
xs      = numpy.array([1, 2, 3, 4, 5, 6, 7])
ys      = numpy.array([3, 2, 5, 1, 3, 3, 2])
colors = ["r","r","r","b","b","g","g"]
plt.scatter(xs, ys, c=colors)
plt.savefig("plot.png")
```



Hint: If you would like plots to appear right inside of your Jupyter Notebook, restart the kernel and evaluate the following before running anything else: % matplotlib inline

Good luck!

Solution.

1. Q. How many features/attributes are there per sample?

   A. There are 4 attributes per sample:

      1. sepal length in cm
      2. sepal width in cm
      3. petal length in cm
      4. petal width in cm

   Q. How many different species are there, and how many samples of each species did Anderson record?

   A. There are 3 species, each having 50 samples:

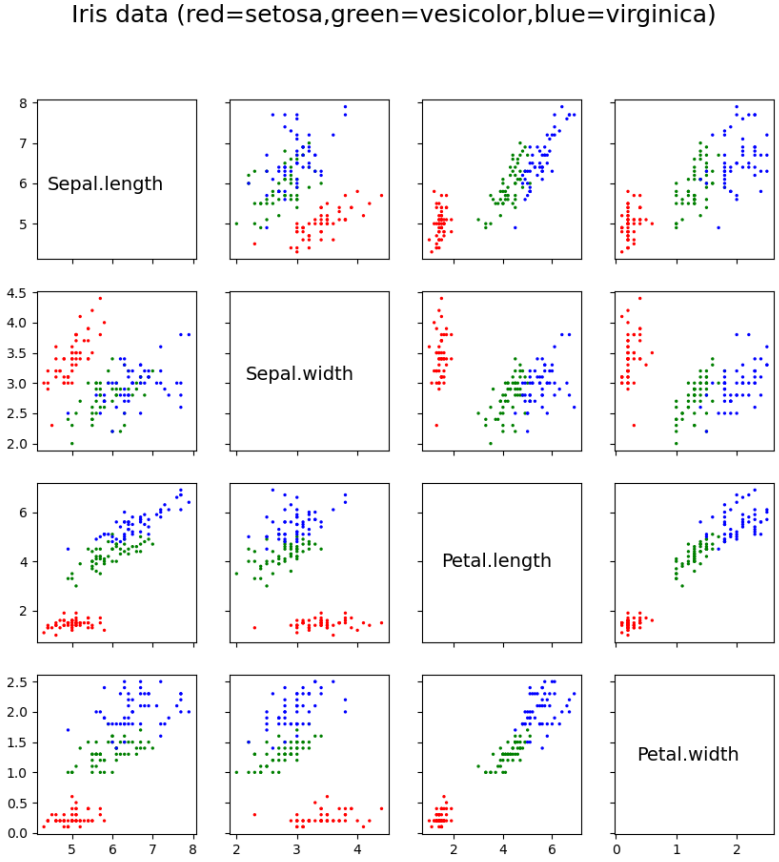
      - Iris Setosa
      - Iris Versicolour
      - Iris Virginica

2. I decided to parse the data by first figuring out how many attributes there are per line. Then I initialized a numpy array with zeros. Every new row in the array is a new sample and each row is a parameter. I also made a numpy array with zeros for every new row samples, so that I can replace each zero with a parameter. Then I add this row to the main dataset array by using vstack. I also stored each samples specie name in a list.

```
import numpy as np

with open("./iris/iris.data") as f:
    first_line = f.readline().strip('\n')
    row_length = len(first_line.split(",")) - 1
raw_data = open("./iris/iris.data")
full_data = np.zeros((1, row_length))
sample_labels = []
for i, line in enumerate(raw_data):
    if i == 150:
        break
    entry_array = np.zeros((1, row_length))
    for idx, parameter in enumerate(line.split(",")):
        if idx == row_length:
            sample_labels.append(parameter.rstrip())
        else:
            entry_array[0, idx] = float(parameter)
    full_data = np.vstack((full_data, entry_array))
full_data = np.delete(full_data, 0, 0)
raw_data.close()
```

3. I mimicked the plot that was shown as an example. I created a 4 by 4 subplot grid with the axis names going across the diagonals. I had issues with centering the axis names, because for some reason my axes had different start and end bounds. Then, by going one by one with the subplots, I filled in the data using two for-loops. The output of the graph was identical to the one in the homework problem.




```
from matplotlib import pyplot as plt

colors = []
for label in sample_labels:
    if label == "Iris-setosa":
        colors.append("r")
    elif label == "Iris-versicolor":
        colors.append("g")
    else:
        colors.append("b")

fig, ax = plt.subplots(4, 4, subplot_kw=dict(box_aspect=1), sharex='col',
     sharey='row', figsize=((10,10)))
fig.suptitle('Iris data (red=setosa,green=vesicolor,blue=virginica)',
    fontsize=18)
for i in range(4):
    for j in range(4):
        if i == j:
            if i == 0:
                name = "Sepal.length"
            elif i == 1:
                name = "Sepal.width"
            elif i == 2:
```

```
                name = "Petal.length"
            else:
                name = "Petal.width"
            ax[i, j].text(full_data[:, j].mean(), full_data[:, j].mean(),
     name,
                      fontsize=14, ha='center')
        else:
            ax[i, j].scatter(full_data[:, j], full_data[:, i], c = colors
    , s = 2)

plt.savefig ("Iris data.png")
```