Using Notebooks for Data Exploration

In this lab we will be exploring the Iris dataset from British statistician and biologist Ronald Fisher (read more here (http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.names)). This dataset is widely used to demonstrate Machine Learning classification techniques. The dataset contains measurements of three different species of Iris flowers collected by Ronald Fisher. The variables include the sepal length, the sepal width, the petal length, and the petal width. Please refer to the image below for a visual reference. The image file is provided courtesy of Wikimedia Commons, by Danielle Langlois under the Creative Commons Attribution-Share Alike 3.0 Unported license (https://commons.wikimedia.org/wiki/File:Iris_versicolor_3.jpg (https://commons.wikimedia.org/wiki/File:Iris_versicolor_3.jpg))

To accomplish our task of understanding the data, we will need to use some Python libraries. Some of the packages we will use today include Pandas (https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673), Scipy (https://www.scipy.org/getting-started.html), Numpy (https://docs.scipy.org/doc/numpy-1.15.0/user/whatisnumpy.html), and MatPlotLib (https://matplotlib.org/). If you want to learn more about how these different packages work with examples, click here (http://cs231n.github.io/python-numpy-tutorial/).

In [2]:
```
#
# first, we need to import a bunch of Python libraries
#

# pandas is a very powerful data management package
import pandas as pd
from pandas.plotting import scatter_matrix

# these are some very standard math/science Python libraries
import scipy as sc
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

# Scikit-learn is a widely used machine learning package
import sklearn as sk
from sklearn.datasets import load_iris

```

In [3]:
```
#
# We will now load a copy of the built-in iris dataset
#

#
# this will create a Scikit learn Bunches dataset structure for us
#
dataset = load_iris()
```

In [6]:
```
#
# For starters, let's look at the way our dataset is constructed
# Let's look at the column headers, or "keys"
#
print('Keys of the iris dataset')
for k in dataset.keys():
    print('key = %s' % (k))
```

```
Keys of the iris dataset
key = data
key = target
key = target_names
key = DESCR
key = feature_names
key = filename
```

In [7]:
```
#
# using the type() method, we can double check that
# our variable dataset is a data type called a
# scikit-learn Bunch, which is kind of like a Panda's data frame
#
type(dataset)
```

Out[7]:
```
sklearn.utils.Bunch
```

In [16]:
```
#
# Let's take a peek at the actual numeric data
#
# note the way we access the dataset.  We specify which attribute
# we want ("data"), then specify which rows we want (here 1:5)
#
dataset['data'][1:5]
```

Out[16]:
```
array([[4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

In [15]:
```
#
# Let's look at some of the other parts of the data
#
# the "feature_names" corresponds to the column headers for the numeric

dataset['feature_names']

```

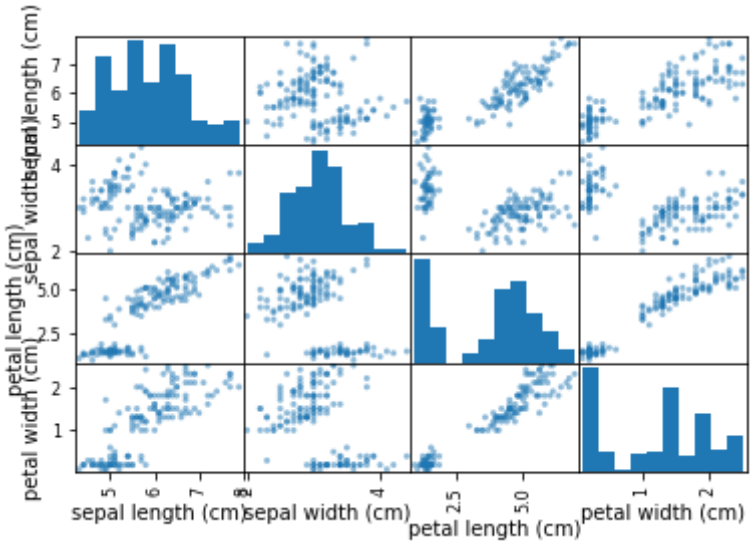
Out[15]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [72]:
```
#
# Let's look at some of the other parts of the data
#
# the "target_names" corresponds to the known species for each row

dataset['target_names']
```

Out[72]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [71]:
```
#
# now let's visualize the data set
# we will first create a Pandas data frame using the DataFrame() method
# then we'll use the scatter_matrix() method to make a graphic
#
pdata = pd.DataFrame( dataset['data'], columns=dataset.feature_names )
pd.plotting.scatter_matrix( pdata )

```

Out[71]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000029211CD35C0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029212E1C5C0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029212E42828>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029212E6DA90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029212E97CF8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029212EBDF60>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029212EF1208>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029212F1A4A8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029212F1A4E0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029212F6A940>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029212F94BA8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029212FBEE10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029212FF00B8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002921301B320>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029213041588>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000292112BFEB8>]],
      dtype=object)
```


sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)

In [54]:
```
#
# let's do some basic stats
# the data set has 50 observations per flower type
# so we can look at blocks of 50 rows at a time
#
for i in range(0,3):
    lo = 50*i
    hi = lo + 50
    name = dataset.target_names[i]
    print('data for flower type %s' % (name))
    print(pdata[lo:hi].describe())

```

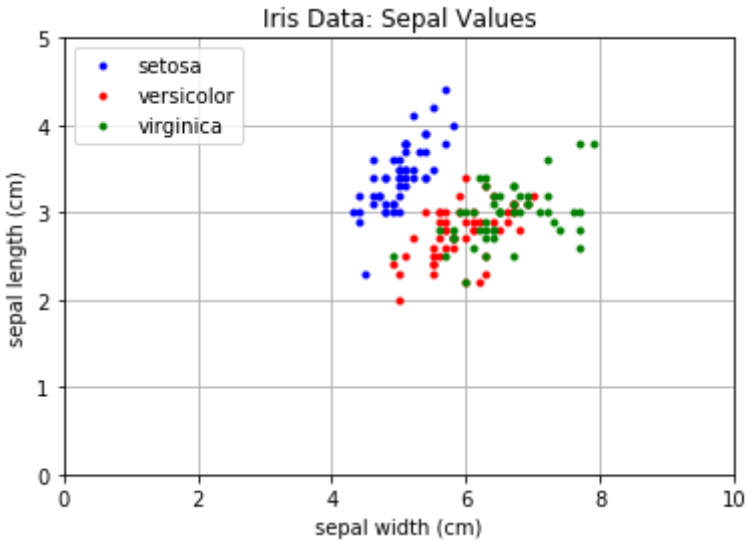
```
data for flower type setosa
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count           50.00000         50.000000          50.000000
mean             5.00600          3.428000           1.462000
std              0.35249          0.379064           0.173664
min              4.30000          2.300000           1.000000
25%              4.80000          3.200000           1.400000
50%              5.00000          3.400000           1.500000
75%              5.20000          3.675000           1.575000
max              5.80000          4.400000           1.900000

       petal width (cm)
count         50.000000
mean           0.246000
std            0.105386
min            0.100000
25%            0.200000
50%            0.200000
75%            0.300000
max            0.600000
data for flower type versicolor
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count           50.000000         50.000000          50.000000
mean             5.936000          2.770000           4.260000
std              0.516171          0.313798           0.469911
min              4.900000          2.000000           3.000000
25%              5.600000          2.525000           4.000000
50%              5.900000          2.800000           4.350000
75%              6.300000          3.000000           4.600000
max              7.000000          3.400000           5.100000

       petal width (cm)
count         50.000000
mean           1.326000
std            0.197753
min            1.000000
25%            1.200000
50%            1.300000
75%            1.500000
max            1.800000
data for flower type virginica
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count           50.00000         50.000000          50.000000
mean             6.58800          2.974000           5.552000
```

std	0.63588	0.322497	0.551895
min	4.90000	2.200000	4.500000
25%	6.22500	2.800000	5.100000
50%	6.50000	3.000000	5.550000
75%	6.90000	3.175000	5.875000
max	7.90000	3.800000	6.900000

	petal width (cm)
count	50.00000
mean	2.02600
std	0.27465
min	1.40000
25%	1.80000
50%	2.00000
75%	2.30000
max	2.50000

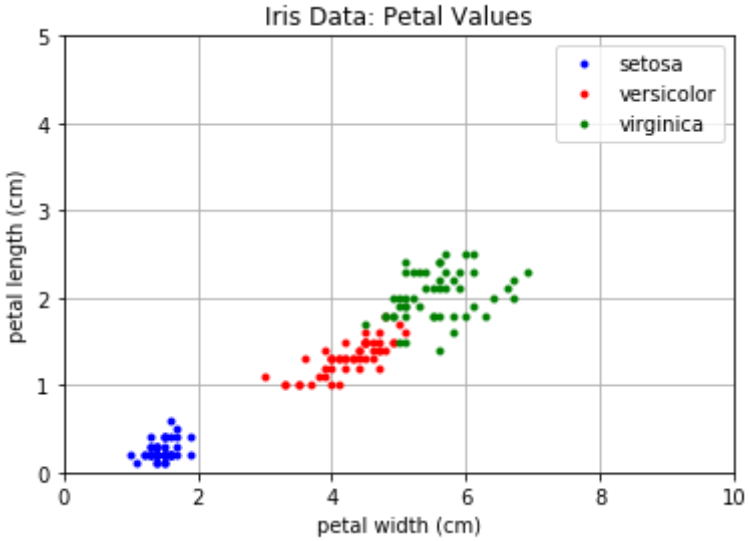
In [76]:

```
#
# let's do some basic visualizions
# the data set has 50 observations per flower type
# so we can look at blocks of 50 rows at a time
#

#
# make a list of blocks of data
#
blocks = []
for i in range(0,3):
    lo = 50*i
    hi = lo + 50
    block = pdata[lo:hi]
    blocks.append( block )

#
# first render the data as dots
#
plt.plot(blocks[0]['sepal length (cm)'], blocks[0]['sepal width (cm)'],
plt.plot(blocks[1]['sepal length (cm)'], blocks[1]['sepal width (cm)'],
plt.plot(blocks[2]['sepal length (cm)'], blocks[2]['sepal width (cm)'],

#
# set up plot embellishments
#
plt.ylim(0,5)
plt.xlim(0,10)
plt.title('Iris Data: Sepal Values')
plt.ylabel('sepal length (cm)')
plt.xlabel('sepal width (cm)')
plt.grid()
plt.legend(('setosa','versicolor','virginica'))
plt.show()

```



In [77]:

```
#
# let's do some basic visualizions
# the data set has 50 observations per flower type
# so we can look at blocks of 50 rows at a time
#

#
# make a list of blocks of data
#
blocks = []
for i in range(0,3):
    lo = 50*i
    hi = lo + 50
    block = pdata[lo:hi]
    blocks.append( block )

#
# first render the data as dots
#
plt.plot(blocks[0]['petal length (cm)'], blocks[0]['petal width (cm)'],
plt.plot(blocks[1]['petal length (cm)'], blocks[1]['petal width (cm)'],
plt.plot(blocks[2]['petal length (cm)'], blocks[2]['petal width (cm)'],

#
# set up plot embellishments
#
plt.ylim(0,5)
plt.xlim(0,10)
plt.title('Iris Data: Petal Values')
plt.ylabel('petal length (cm)')
plt.xlabel('petal width (cm)')
plt.grid()
plt.legend(('setosa','versicolor','virginica'))
plt.show()
```



In [ ]:

```
#
# now use the data to answer some questions
#
# 1. what is the mean petal size of each species of Iris
# 2. what is the mean sepal size of each species of Iris?
# 3. could you distinguish species using only sepal length and width? W
# 4. could you distinguish species using only petal length and width? W
```