Cairo University
Faculty of Computers and Artificial Intelligence
Machine Learning Course

Lab 2: Data Analysis with Python

Lab Outline:

- Installing the required packages
- Overview of Pandas, NumPy, Matplotlib and Seaborn

Installing Packages:

You can use **pip** to install packages. Check that you have a working Python with **pip** installed by running the following commands and making sure that the output looks similar:

```
C:> py --version
Python 3.N.N
C:> py -m pip --version
pip X.Y.Z from ... (python 3.N.N)
```

Then, use **pip** as follows:

```
pip install pandas
pip install scikit-learn
pip install matplotlib
pip install seaborn
```

Overview of Some Data Analysis Packages:

Pandas:

Pandas is an open source powerful Python library used for data analysis and manipulation.

Pandas provides two types of classes for handling data:

- **Series:** a one-dimensional labeled array holding data of any type such as integers, strings, Python objects etc.
- **DataFrame:** a two-dimensional data structure that holds data like a two-dimensional array or a table with rows and columns.

Import Pandas by writing:

```
import pandas
```

Let's create a series and a dataframe:

```
my_data = [5.1, 4.9, 4.7]
my_series = pandas.Series(my_data)
print(my_series)
```

```
0    5.1
1    4.9
2    4.7
dtype: float64
```

```
my_data = {'Col1': [5.1, 4.9, 4.7], 'Col2': [3.5, 3.0, 3.2]}
my_df = pandas.DataFrame(my_data)
print(my_df)
```

```
   Col1  Col2
0   5.1   3.5
1   4.9   3.0
2   4.7   3.2
```

We are now going to use Pandas to load and manipulate the iris flowers dataset. This dataset contains 150 observations of iris flowers. There are four columns of measurements of the flowers in centimeters (the length and the width of the sepals and petals). The fifth column is the species of the flower observed. All observed flowers belong to one of three species.

We will load the iris data from the CSV file as follows:

```
# url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

data = pandas.read_csv("iris.data", names=['sepal-length', 'sepal-width',
'petal-length', 'petal-width', 'class'])

print(data)
print(type(data))
```

```
   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
..            ...          ...           ...          ...             ...
```

```
145            6.7           3.0            5.2           2.3  Iris-virginica
146            6.3           2.5            5.0           1.9  Iris-virginica
147            6.5           3.0            5.2           2.0  Iris-virginica
148            6.2           3.4            5.4           2.3  Iris-virginica
149            5.9           3.0            5.1           1.8  Iris-virginica

[150 rows x 5 columns]
<class 'pandas.core.frame.DataFrame'>
```

Note: The local path of the file can be replaced by the URL.

Explore the data:

```
print(len(data)) # data.shape returns both dimensions
print(data.head()) # default 5 rows
print(data.tail(3))
print(data[120:122])
```

```
150
   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
     sepal-length  sepal-width  petal-length  petal-width           class
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica
     sepal-length  sepal-width  petal-length  petal-width           class
120           6.9          3.2           5.7          2.3  Iris-virginica
121           5.6          2.8           4.9          2.0  Iris-virginica
```

To select certain data, we can use **loc** or **iloc**. The **loc** function requires row labels whereas the **iloc** function requires indexes (i.e. integer positions).

```
print(data.iloc[0]) # prints the first row
print(data.iloc[:, 0]) # prints the first column
print(data.iloc[0, 0]) # prints the first cell
print(data.iloc[20:101, 0:2])
print(data.loc[:, 'class'])
print(data.loc[20:101, 'sepal-width']) # why does it work?
# print(data.iloc[:, 'class']) produces an error
# print(data.loc[0,0]) produces an error
```

```
print(data.values) # prints numpy representation, data.to_numpy() can be used
print(data.T) # transpose
```

```
print(data.columns)
print(data.index)
```

```
Index(['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class'], dtype='object')
RangeIndex(start=0, stop=150, step=1
```

Manipulate the data:

```
data.drop(columns=['class'])
print(data) # the original data is not affected
```

```
   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
..            ...          ...           ...          ...             ...
145           6.7          3.0           5.2          2.3  Iris-virginica
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica

[150 rows x 5 columns]
```

```
data2 = data.copy()
data2.drop(columns=['class'], inplace=True)
print(data2)
```

```
   sepal-length  sepal-width  petal-length  petal-width
0             5.1          3.5           1.4          0.2
1             4.9          3.0           1.4          0.2
2             4.7          3.2           1.3          0.2
3             4.6          3.1           1.5          0.2
4             5.0          3.6           1.4          0.2
..            ...          ...           ...          ...
145           6.7          3.0           5.2          2.3
146           6.3          2.5           5.0          1.9
147           6.5          3.0           5.2          2.0
148           6.2          3.4           5.4          2.3
149           5.9          3.0           5.1          1.8

[150 rows x 4 columns]
```

Note: We can use drop for dropping certain rows or columns by specifying the index/label of the row or column and the axis.

```
data.drop(1, axis=0) # drops the second row
data.drop('class', axis=1) # drops the last column
```

```
data['x0'] = [1]*150 # inserts a column of 1s at the end
data.insert(1, 'x1', [1]*150)
data.drop(columns=['x0', 'x1'], inplace=True)
```

```
print(data.sort_index(axis=1, ascending=True)) # axis = 0 to sort rows
print(data.sort_values(by='sepal-length'))
```

```
     class  petal-length  petal-width  sepal-length  sepal-width
0      Iris-setosa           1.4          0.2           5.1          3.5
1      Iris-setosa           1.4          0.2           4.9          3.0
2      Iris-setosa           1.3          0.2           4.7          3.2
3      Iris-setosa           1.5          0.2           4.6          3.1
4      Iris-setosa           1.4          0.2           5.0          3.6
..             ...           ...          ...           ...          ...
145  Iris-virginica          5.2          2.3           6.7          3.0
146  Iris-virginica          5.0          1.9           6.3          2.5
147  Iris-virginica          5.2          2.0           6.5          3.0
148  Iris-virginica          5.4          2.3           6.2          3.4
149  Iris-virginica          5.1          1.8           5.9          3.0

[150 rows x 5 columns]
     sepal-length  sepal-width  petal-length  petal-width           class
13            4.3          3.0           1.1          0.1     Iris-setosa
42            4.4          3.2           1.3          0.2     Iris-setosa
38            4.4          3.0           1.3          0.2     Iris-setosa
8             4.4          2.9           1.4          0.2     Iris-setosa
41            4.5          2.3           1.3          0.3     Iris-setosa
..            ...          ...           ...          ...             ...
122           7.7          2.8           6.7          2.0  Iris-virginica
118           7.7          2.6           6.9          2.3  Iris-virginica
117           7.7          3.8           6.7          2.2  Iris-virginica
135           7.7          3.0           6.1          2.3  Iris-virginica
131           7.9          3.8           6.4          2.0  Iris-virginica

[150 rows x 5 columns]
```

Calculate some statistics:

```
print(data.mean())
print(data.mean(1))
print(data.median())
```

```
sepal-length    5.843333
sepal-width     3.054000
petal-length    3.758667
petal-width     1.198667
dtype: float64
0      2.550
1      2.375
2      2.350
3      2.350
4      2.550
       ...
145    4.300
146    3.925
147    4.175
148    4.325
149    3.950
Length: 150, dtype: float64
sepal-length    5.80
sepal-width     3.00
petal-length    4.35
petal-width     1.30
dtype: float64
```

```
print(data.describe())
```

```
class           Iris-setosaIris-setosaIris-setosaIris-setosaIr...
dtype: object
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

```
print(data.isna().sum())
print(data['class'].str.contains('setosa').sum())
```

```
sepal-length    0
sepal-width     0
```

```
petal-length    0
petal-width     0
class           0
dtype: int64
50
```

```
print(data.groupby('class').size())
print(data['class'].value_counts())
```

```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
Iris-virginica     50
Iris-setosa        50
Iris-versicolor    50
Name: class, dtype: int64
```

Filter the data:

```
print(data['sepal-width'])
print(data['sepal-width'] > 4)
print(data[data.get('sepal-width') > 4]) # use & for multiple filters
```

```
0      3.5
1      3.0
2      3.2
3      3.1
4      3.6
      ...
145    3.0
146    2.5
147    3.0
148    3.4
149    3.0
Name: sepal-width, Length: 150, dtype: float64
0      False
1      False
2      False
3      False
4      False
       ...
145    False
146    False
147    False
148    False
149    False
Name: sepal-width, Length: 150, dtype: bool
```

```
    sepal-length  sepal-width  petal-length  petal-width        class
15           5.7          4.4           1.5          0.4  Iris-setosa
```

NumPy:

NumPy is a fundamental package for scientific computing in Python. It is a library that provides a multidimensional array object and an assortment of routines for fast operations on arrays (including mathematical, logical, shape manipulation, discrete Fourier transforms, basic linear algebra, statistical operations, etc.).

Here is a sample of NumPy's functions and usage:

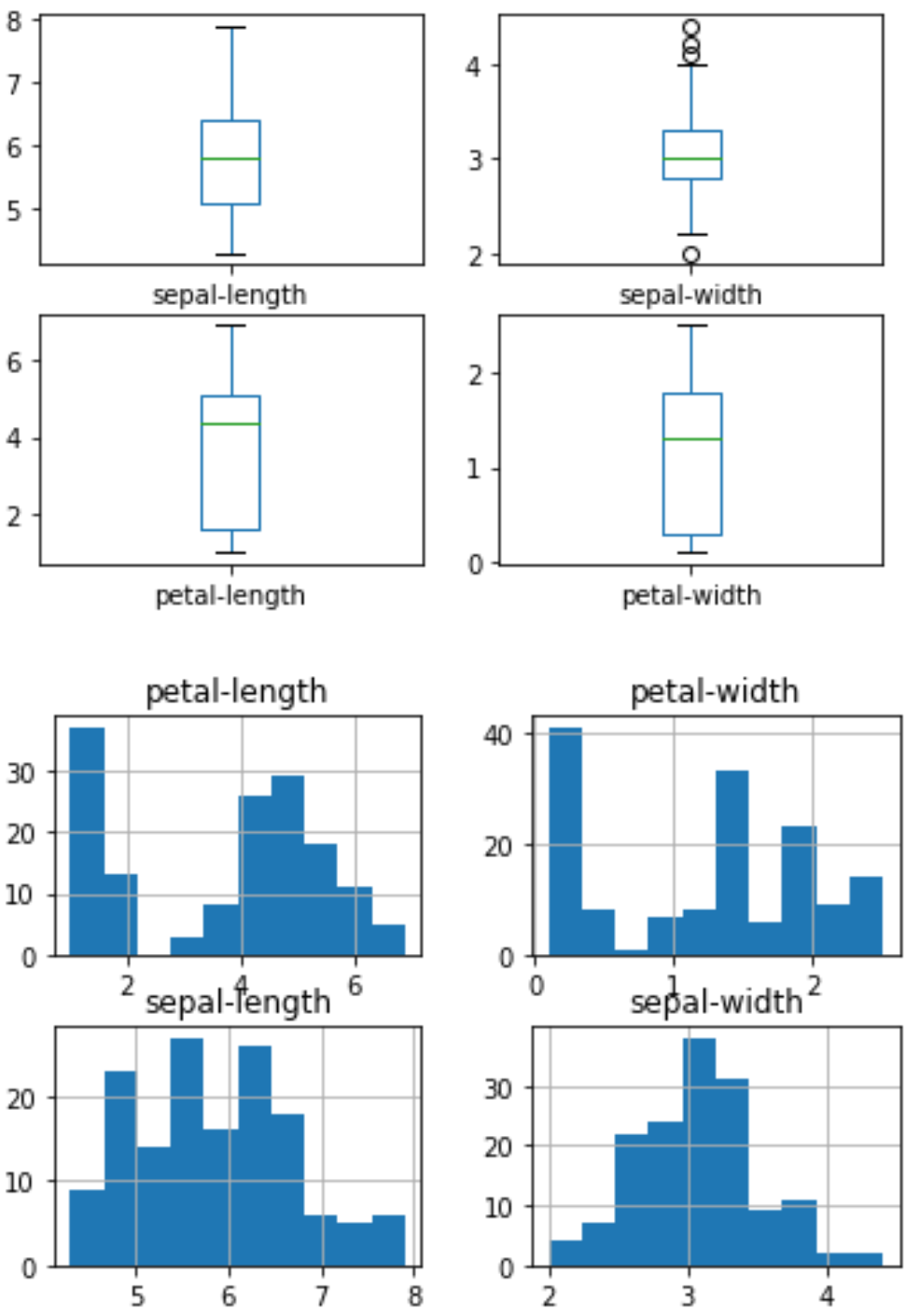
```
import numpy as np

x = np.array(data['sepal-width'])
y = data['petal-length'].to_numpy()
z = np.zeros(150)
print(x+y*(z+2))
print(np.stack((y,x), axis=1))

w = x.reshape(1,-1)
print(w.shape)
print(np.multiply(np.ones((150,1)), w))

matrix = np.array(data.drop('class', axis=1))
print(matrix.shape)
print(np.vstack((matrix, np.ones((1,4)))))
print(matrix.mean(), matrix.mean(0), matrix.mean(1))
print(matrix.flatten())
```

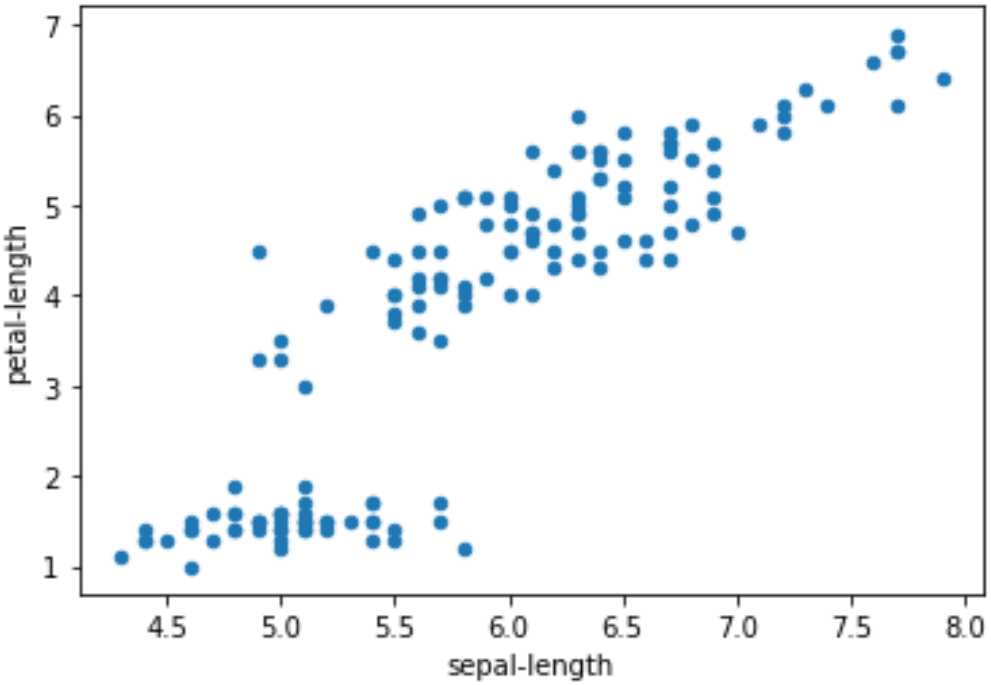
Matplotlib:

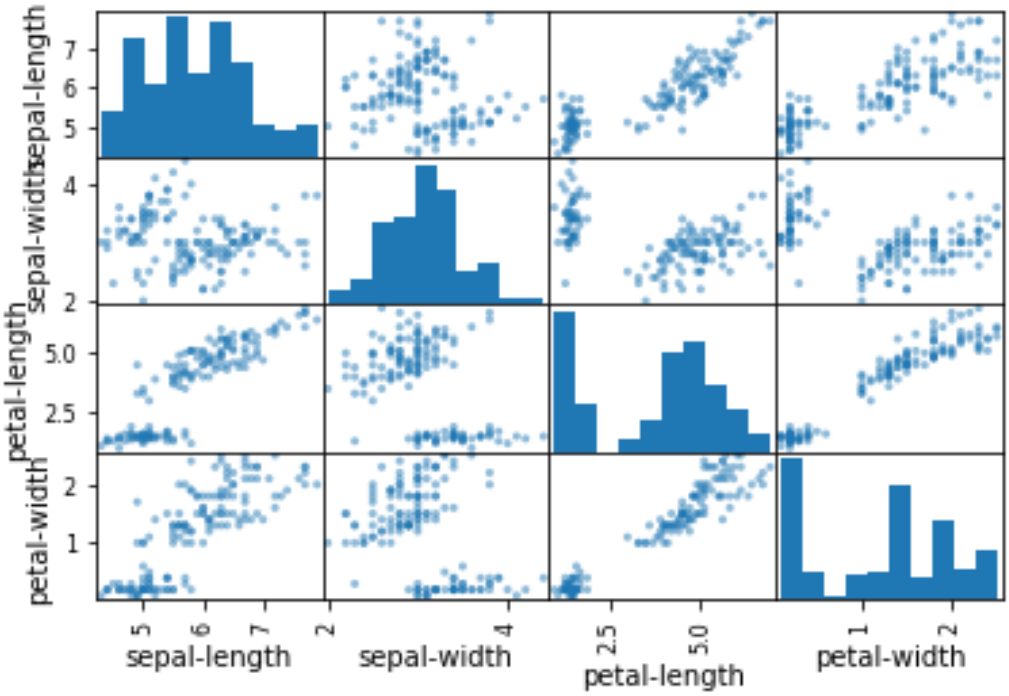
Matplotlib is a Python library that serves as a visualization utility. Here are some examples:

```
import matplotlib

data.plot(kind='box', subplots=True, layout=(2, 2), sharex=False,
sharey=False)
matplotlib.pyplot.show()

data.hist()
matplotlib.pyplot.show()

data.plot(kind='scatter', x='sepal-length', y='petal-length')
```

```
matplotlib.pyplot.show()

pandas.plotting.scatter_matrix(data)
matplotlib.pyplot.show()
```




petal-length
sepal-length
7
6
5
4
3
2
1
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

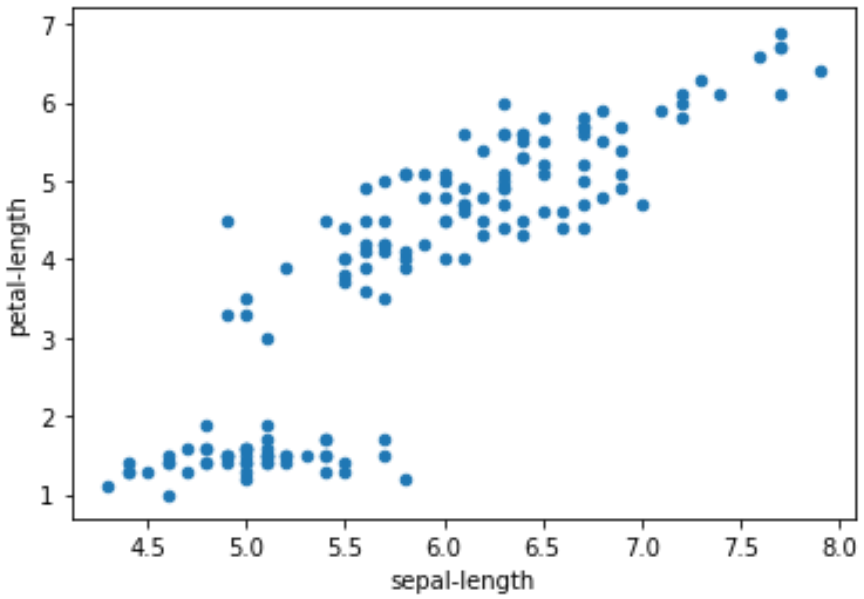

sepal-length
sepal-width
petal-length
petal-width

Seaborn:

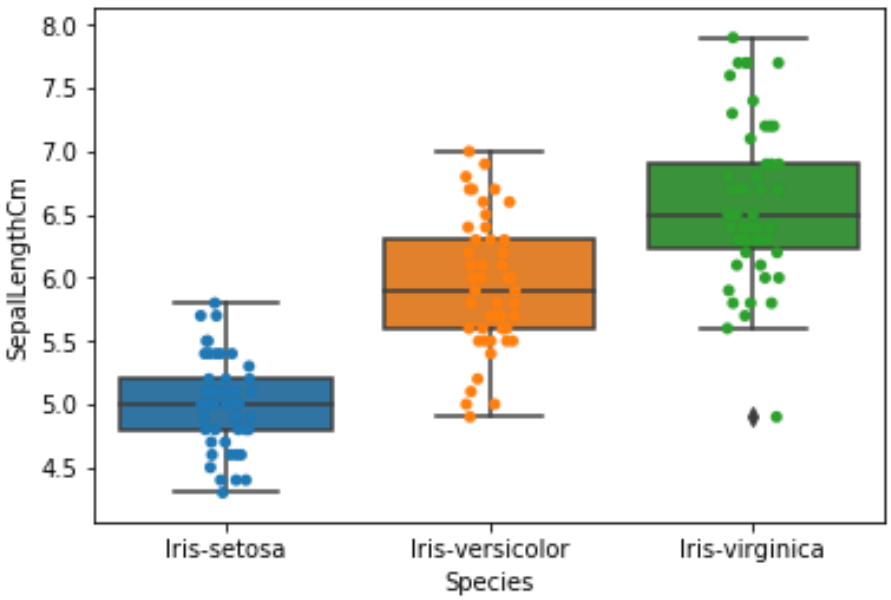
Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Here is a sample of Seaborn's plots:

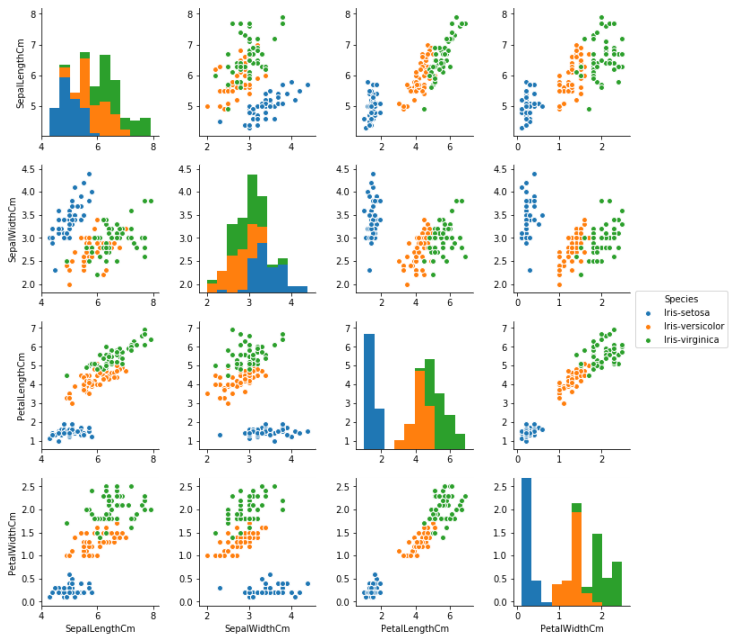
```
import seaborn as sns

irisData = data.copy()
sns.jointplot(x='sepal-length', y='petal-length', data=irisData, size=5)
```




```
combined = sns.boxplot(x='class', y='sepal-length', data=irisData)

combined = sns.stripplot(x='class', y='sepal-length', data=irisData,
jitter=True, edgecolor='gray')
```




```
sns.pairplot(irisData, hue='class')
```



```
sns.heatmap(irisData.corr(),annot=True,cmap='summer')
```