Lab 2: Introduction to NumPy, Pandas and Matplotlib[1] (Week 3)

1 – Objectives

With this work the student should be able to use NumPy, start using the Pandas library as a data structure to store his Data, and the Matplotlib to show some results in graphical format.

2 – Package installation

Python comes with an inbuilt package management system, pip. Pip can install, update, or delete any official package. You can install packages via the command line by entering:

python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose

3 – NumPy

NumPy (https://numpy.org/) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. A NumPy tutorial is available at https://numpy.org/devdocs/user/quickstart.html

Some examples of how to use NumPy:
Array creation

```
>>> import numpy as np
>>> x = np.array([1, 2, 3])
>>> x
array([1, 2, 3])
>>> y = np.arange(10)  # like Python's range, but returns an array
>>> y
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Basic operations

```
>>> b = np.linspace(0, 2, 4)  # create an array with four equally
spaced points starting with 0 and ending with 2.
>>> c = a - b
>>> c
array([ 1.        ,  1.33333333,  1.66666667,  4.        ])
>>> a**2
array([ 1,  4,  9, 36])
```

[1] Rui Neves, João Paulo Carvalho

Universal functions

```
>>> a = np.linspace(-np.pi, np.pi, 100)
>>> b = np.sin(a)
>>> c = np.cos(a)
```

Linear algebra

```
>>> from numpy.random import rand
>>> from numpy.linalg import solve, inv
>>> a = np.array([[1, 2, 3], [3, 4, 6.7], [5, 9.0, 5]])
>>> a.transpose()
array([[ 1. ,  3. ,  5. ],
       [ 2. ,  4. ,  9. ],
       [ 3. ,  6.7,  5. ]])
>>> inv(a)
array([[-2.27683616,  0.96045198,  0.07909605],
       [ 1.04519774, -0.56497175,  0.1299435 ],
       [ 0.39548023,  0.05649718, -0.11299435]])
>>> b =  np.array([3, 2, 1])
>>> solve(a, b)  # solve the equation ax = b
array([-4.83050847,  2.13559322,  1.18644068])
>>> c = rand(3, 3) * 20  # create a 3x3 random matrix of values within
[0,1] scaled by 20
>>> c
array([[  3.98732789,   2.47702609,   4.71167924],
       [  9.24410671,   5.5240412 ,  10.6468792 ],
       [ 10.38136661,   8.44968437,  15.17639591]])
>>> np.dot(a, c)  # matrix multiplication
array([[  53.61964114,   38.8741616 ,   71.53462537],
       [ 118.4935668 ,   86.14012835,  158.40440712],
       [ 155.04043289,  104.3499231 ,  195.26228855]])
>>> a @ c # Starting with Python 3.5 and NumPy 1.10
array([[  53.61964114,   38.8741616 ,   71.53462537],
       [ 118.4935668 ,   86.14012835,  158.40440712],
       [ 155.04043289,  104.3499231 ,  195.26228855]])
```

4 – The Pandas Library

The Pandas library enables the Python programmer to have a place to store his data. Pandas stores the data in a spreadsheet like format (as you would do in Excel), with each column used to store one data feature.

Examples of Pandas tutorials:

https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introductionfor-beginners/

https://data36.com/pandas-tutorial-1-basics-reading-data-files-dataframes-dataselection/

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Sometimes you can use Numpy arrays for simpler things and have the core of your data stored in a Pandas DataFrame.

Index	Inflation	GDP	People	Cars
0	1.8	1.4	10 M	3 M
1	1.7	1.8	10.3 M	3.1 M
2	1.3	2.2	10.4 M	3.2 M

Figure 1. A Pandas DataFrame

A pandas Series is a pair (index, data).

Index	Inflation
0	1.8
1	1.7
2	1.3

Figure 2. A Pandas series

The simpler way to create a Panda Dataframe is to read a .csv file where you store your Data. First you need to import the library. Then read the file.

Exercise 1: *Open file "AAPL_yah.csv", create a DataFrame, sort it by column "Close" and then store the data in a new file called "ChangeData.csv"*

```
import numpy as np
import pandas as pd

df = pd.read_csv('AAPL_yah.csv')

df1 = df.sort_values(by=['Close'])

df1.to_csv('ChangedData.csv')
#open the file to see what happened
#notice that the index was also written

#now control how you write to the file
df1.to_csv('ChangedData1.csv', decimal=',', sep=';', index=False)
```

Pandas has many functions, before starting doing your own functions find out if it is already implemented:

https://pandas.pydata.org/pandas-docs/stable/reference/frame.html

Pandas is computationally very efficient when compared to standard Python. Check it out yourself:

Exercise 2: *Do the cumulative sum of one column. For that use the function "cumsum" and store the data in a new column (just one line of code…).*

Now try to do that using a "for" cycle. To see how much time you take to execute your code, use the following instructions:

```
import time
start_time = time.clock()

df['newCol'] = df['Volume'].cumsum()

print ("time used in cumsum -->", time.clock() - start_time, "seconds")
```

The library numpy has a very usefull function called "where":

numpy.where(condition[, x, y])

This function can be used to perform some operation in all off the numpy array, and also be used in conjunction with Pandas.

Exercise 3: *Try the following code:*

```
a = np.arange(10)
print(a)
#a = array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
b = np.where(a < 5, a, 10*a)
print(b)
#array([ 0,  1,  2,  3,  4, 50, 60, 70, 80, 90])

df['signal']= np.where(df['Close'] > df['Open'] , 1.0, 0.0)
```

Now is time to explore the pandas library and see what other functions are available. See what you can do with the functions "loc", "iloc", df[1:3], df[:4], df[4:]. There are many functions, explore and try them.

5 – The Matplotlib Library

The matplotlib is a library that enables you to show results in graphical form. The best way to learn how it works is to see some examples and change them. The matplot lib tutorial link is presented next:

https://matplotlib.org/3.1.1/tutorials/index.html

https://matplotlib.org/3.1.1/tutorials/introductory/pyplot.html

https://matplotlib.org/3.1.1/gallery/index.html

Exercise 4: *Try the following examples:*

a) Plot some points and make a y label. You use the "plot" function to say what you are going to plot. The "show" function shows the drawing to the user.

```
# First example, my first plot
plt.plot([1, 2, 3, 4])
plt.ylabel('some numbers')
plt.show()
```

b) Use the axis function to control which points are used in the x and y axis.

```
# second example, lets control the axis
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```

c) The third example is a little more complex:

```
# third example, lets have two plot in one figure
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure()
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```

6 – Application

Now that you are more familiar with pandas and matplotlib, it is time to make exercise your skills by getting some data, processing it using Pandas, and visualize the output with matplotlib.

Exercise 4: *The file DCOILBRENTEU.csv shows the evolution of the oil price through time.*

a) Normalize (min-max) and standardize (Z-score) the price (using the methods you learned in the classes) and visualize it. Does it make sense to standardize this type of data?

b) Smooth the normalized price by computing a moving average with a 30-day period. Visualize the result.