Lab of Applied Computational Intelligence

IST

2024/2025

Introduction to Pandas and Matplotlib

Guide 2

13 September 2024

(Week 1)

1 – Objetives

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

2 – The Panda Library

The Pandas library enables the Python programmer to have a place to store his data. Pandas Stores the data like a spreadsheet, like you would use in Excel, with each column used to store your data feature.

Examples of pandas tutorials:

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

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

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 Panda 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 just a pair (index, data), we will be using mainly the Dataframe.

Index	Inflation
0	1
1	1.7
2	1.3

Figure 2. A Pandas Series

The simpler way to create a Panda Dataframe is just read a csv file where you store your Data. First you need to import the library. Then read the file. For example, sort the dataframe by the 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 for example in the following link:

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

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

You can do the cumulative sum of one column. For that just use the function "cumsum" and store the data in a new column in just one line of code.

Now try to do that with a for cycle, see who is faster. To see how much time do you take to execute your code use the following code. You can use process_time() or perf_counter() to compute relative times.

```
import time
start_time1 = time.process_time()
start_time2 = time.perf_counter()

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

print ("time used in cumsum -->", time.process_time() - start_time1, "seconds")
print ("time used in cumsum -->", time.perf_counter() - start_time2, "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 conjuction with pandas.

```
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 see what other function there are available. See what you can do with the function “loc”, “iloc”, df[1:3], df[:4], df[4:]. There are many functions explore and try them.

3 – The Matplotlib Library

The matplotlib is a library that enables you to show some results in graphical form. The best way to learn is to see some examples and try to change them. The matplot lib tutorial link is presented next, where you have an introduction and then some examples:

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

The first example is very simple, just plot some points and make a y label. You use the plot function to say who you are going to plot, but only after the show function the drawing will be seen by the user.

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

In the second example use the axis function to control which point 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()
```

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()
```

4 – Execution of the lab work

Now that you are more familiar with pandas and matplotlib is time to make some example application. Get same data, process it, and visualize the output with matplotlib.

The file DCOILBRENTEUv2.csv shows the evolution of the oil price through time. a) Normalize (min-max) and standardize (Z-score) the price and visualize it. b) Smooth the normalized price by computing a moving average with a 50-day period. Visualize the result.

Min-Max normalization: newx = x-min/(max-min)

z-score standardization z = (x-μ)/σ=(x-mean)/(standard deviation)