PYTHON FOR MACHINE LEARNING EXAM 1

COMPUTER PART, TRY 2

10. September 2019, Helsinki, Finland

How to start the exam: To avoid submitting .ipynb files, first run the Run_me.sh your_name_here
That bash script that creates the solution files for you. Make sure that the file is executable. If you use Windows, then you should find another way.
After you finish a problem, you should make sure that you have copied your solution only to this file, and nowhere else.

How to package your solution:

Before you submit, make sure each function has been properly documented.
Your whole submission has to be zipped to a .ZIP archive, named computer_exam_1_your_name, together with the relevant data and anything needed to run your submissions.
The archive has to be sent to me directly on SLACK when you end the exam. Don't share your submission with anyone else.

NB!
Important:
All the solutions in the computer exam have to be presented as a Python script files with .py ending. The tasks will ask you to also present a screenshot of your program running from the Command Line (Terminal) to help you remember that.

SUBMISSION:

YOU CAN CHOOSE TO DO 4 PROBLEMS OUT OF 5.

What does that mean?

Better do 4 problems well that 5 problems half-way -- i.e. if one problem seems too hard, then don't spend all the time on this problem, but do these 4 that you understand better.

You should not run __main__ inside Jupyter notebook, because this doesn't convert the Jupyter Notebook into Python script. Refer to this REFERENCE to convert IPYNB to Python executable.

BEFORE YOU START THE EXAM

Please review what is a Python script and remember that for that, you definitely need to have the file properly formatted, according to PEP8 rules, and saved in *.py format.

PROBLEM 1 : Pandas : Pipe and Pivot Table

PART 1: PIPE

- Read in the Titanic Dataset from the “data” folder.
- Define two methods that can be applied on Pandas DataFrame and use the .pipe operator to apply these methods in such a way to find the relative frequencies of each PClass value in the Titanic Dataset.
- The first method should be called “select_column” and it should accept a Pandas DataFrame and a column_specifier that can be either string or integer; the second method should be called **find_ratio(series),**
  where the first method is a selector method and the other method is to find relative frequencies for a Pandas Series object.

- The Program has to be an executable Python script that prints out two statements:
  - 1. The relative PClass frequencies in the Titanic Dataset
  - True if there is a way to select the PClass column in the Pandas DataFrame both via column_specifier being integer or string value; **ANd if these selections give identical outcome.**

  NOTE : The goal of the program is to:
  - Understand how the Pandas .pipe operator works
  - Prove that there are 2 identical ways of selecting a Pandas DataFrame column, **and that they give identical result.**

PS: Pandas Code often works without .pipe, but the idea of Pipe is to learn clean code formatting.

PART 2 : PIVOT TABLE

In this part, your goal is using the Titanic data set, reproduce the following Pivot Table:

class	First	Second	Third
sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

Question: What is this pivot table actually showing?

PROBLEM 2 : Time Series in BUSINESS Context

1. Create a DatetimeIndex that contains each business day of 2018 and generate a uniformly distributed numbers for each DateTimeIndex value. Save the result in a Series.
2. Call the column of random numbers “normalized_losses” and create a new dataframe for this series.
3. Find the average and median normalized loss in 3 ways:
    a. Over all Mondays and Fridays of 2019
    b. Over all weeks

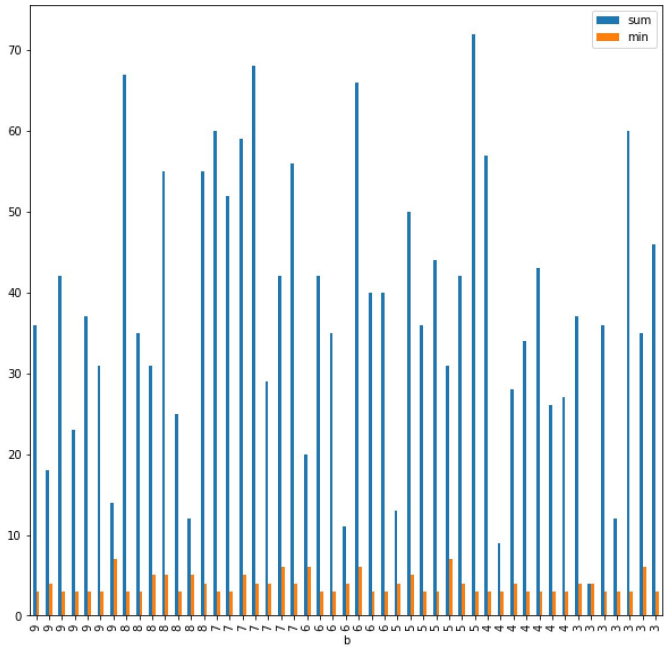
4. Find the month of 2018 with the highest minimum normalized loss. Mote : You may use the ‘resample’ and/or TimeGrouper.

5. Consider now the statistics over the quarters (1-2-3-4) of the entire year. Find the quarter for which the 25%-quantile of the normalized loss is the highest.

Submission :

The program has to implemented as executable Python script and you should also submit a screenshot of the program running from the Command Line (Terminal).
Define a new method for each of the points (e.g. def step1, def step2 etc).
When the script is run, all the methods that work should run one by one and print the necessary results to the console.
The methods that don’t work as needed should be commented out, where you can write the explanations why it doesn’t work in the comments as well.

PROBLEM 4. PANDAS OPERATIONS



Given a Pandas Dataframe of random numbers
bardata =

pd.DataFrame(np.random.randint(3,10,size=(300,4)),columns=['a', 'b', 'c', 'd'])

What is the meaning of the following table on the right? +Reproduce it.

		a	
		sum	min
b	c		
3	3	81	3
	4	65	3
	5	39	4
	6	12	3
	7	29	3
	8	18	4
	9	56	3
4	3	44	3
	4	42	3
	5	40	3
	6	35	5
	7	19	3
	8	47	3
	9	28	3
5	3	61	4

- **What is the meaning of the graph, in your own words?**
- **+ Reproduce it**.
- HINT : The bars in the plot are representing the a-variable certain aggregation values (sum, min).

Code Formatting Requirement: The code has to be formatted vertically, not horizontally, meaning that the formatting should emphasize data pipeline formatting (please refer to the df.pipe examples).

Submission : The program has to implemented as executable Python script and you should also submit a screenshot of the program running from the Command Line (Terminal).

PROBLEM 4 : UNDERSTANDING FUNCTION APPLICATION

Given

```
import pandas as pd
import numpy as np
import functools
from functools import partial
def conjunction(*conditions):
    return functools.reduce(np.logical_and, conditions)

def disjunction(*conditions):
    return functools.reduce(np.logical_or, conditions)


##################### GENERATING NECESSARY INPUTS #####

df = pd.DataFrame(np.random.randint(0,100,size=(100, 4)), columns=list('ABCD'))



######################## PANDAS + FUNCTIONAL STARTS  ###############

def dictionary_apply_kwargs(dicti_, **kwargs):
```

```
    """ Applies a given function to all dictionary items (values)"""
    s = dict()
    fun1 = kwargs.get('step1')
    arg1 = kwargs.get('arg1')
    for k,v in dicti_.items():
        if v is not None:
            s[k] = fun1(v,arg1)
    return s

def data_transformer(data=df,conditions,**kwargs):
    all_true = data[data.apply(conjunction(*conditions))]
    any_true =  data[data.apply(disjunction(*conditions))]
    dict_ = {'any': any_true, 'all': all_true,"step2":None}
    if kwargs is not None:
        step1 = kwargs.get("step1")
        step2 = kwargs.get("step2")
    dictionary_apply_kwargs(dicti_=dict_,**kwargs)

    return dict_

############################ RUNTIME ############################
arg_one =  50
kwargs = {'step1':pd.DataFrame.sub,'arg1':arg_one}
from_step_1 = data_transformer(**kwargs)
```

What to do:

Define 2 conditions:

- Filter out all values that are greater than 3, from the C Column. Call this step1.
- Filter out all values smaller than 50, from the A column. Call this step2.

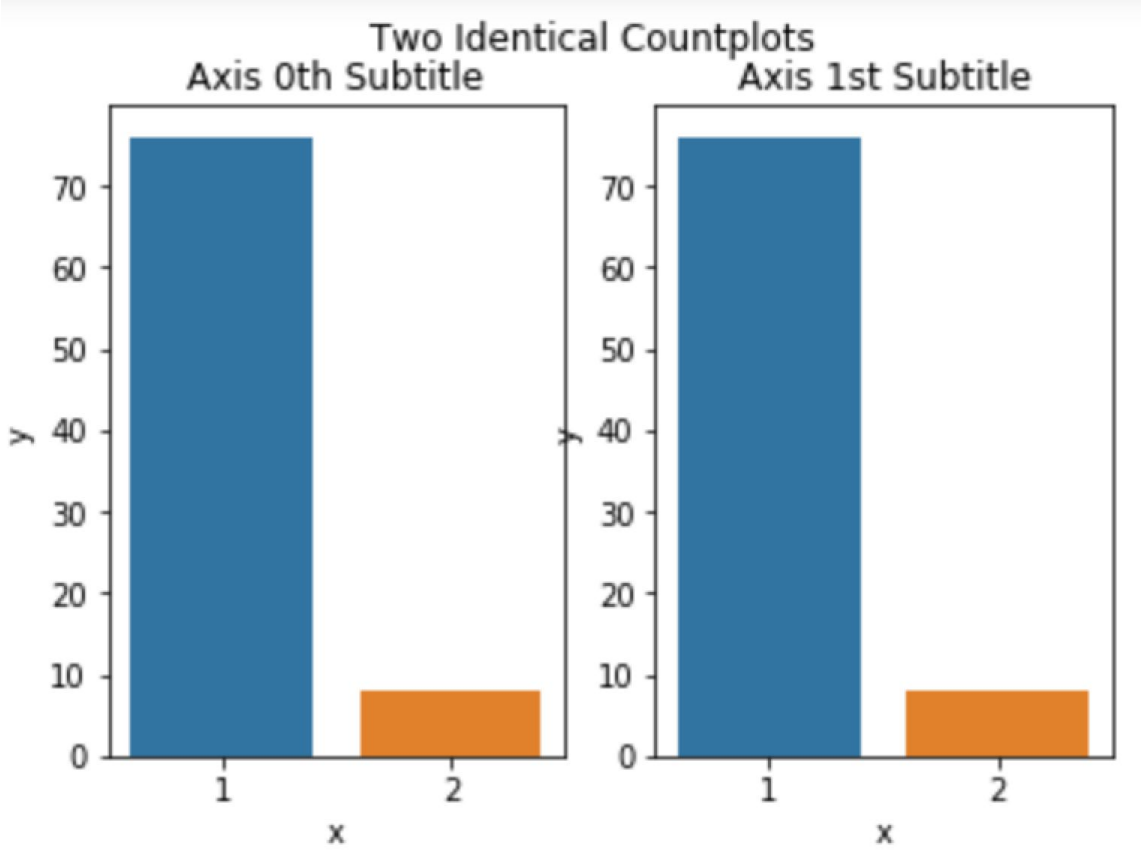
You should modify the existing functions and/or add any functionality to accomplish the following output :
The data_transformer in your code should return the following dictionary:
dict = {“step1”: data1,"step2":data2,”all” : data3}

Where
-step1, data1 is defined as the subset of the original dataframe that matched the step1 condition
-step2, data2 is defined as the subset of the original dataframe that matched the step2 condition

- all, data3 -- is defined as the subset of the original dataframe where both conditions are True.

How data3 should be obtained : these two functions (producing step1, and from step 1 producing step2) work together as a simple data pipeline, that is, apply the data_transformer function in such a way that both steps are applied consecutively (one after the other).
The program has to implemented as executable Python script and you should also submit a screenshot of the program running from the Command Line (Terminal).

PROBLEM 5. Reproducing Bernoulli Visualization in Seaborn.



Using a given Pickle file pandas_c.pickle, reproduce the binary variable graph below.
I.e.,
the goal is to generate a two-subplot plot in Seaborn using two different methods.**

HINT: Recall:

- **list methods**
- **Counter**
- **Seaborn countplot**
- **Seaborn barplot**

You may make use of the following helper function:

```
def flatten(items, seqtypes=(list, tuple)):
    for i, x in enumerate(items):
        while i < len(items) and isinstance(items[i], seqtypes):
            items[i:i + 1] = items[i]
    return items
```

The program has to implemented as executable Python script and you should also submit a screenshot of the program running from the Command Line (Terminal).

WHEN YOU HAVE FINISHED YOUR EXAM : PEP8 CHECK

Make sure that the file properly formatted, according to PEP8 rules, and saved in *.py format.

BONUS PROBLEM 6: If you have done your best in the main exam part, you do the following to get some extra points:

You can substitute of the 5 problems by answering the following questions:

-What file contents do you think look like this?

```
click==7.0
hvac==0.6.0
numpy==1.16.0
pandas==0.23.4
pytest==3.6.2
python-dateutil==2.7.3
redis==2.10.6
mlxtend==0.15.0.0
tabulate==0.8.2
scipy==1.2.0
scikit-learn==0.20.2
sklearn==0.0
tqdm==4.29.1
```

Remember what files we have analyzed when we were talking about virtual environments.

- Why does this file look like this?
- Your task : Filter out to leave in tqdm, sklearn, numpy and pandas and create a new virtual environment according to your preference, where only these 4 dependencies are installed at creation time.
- SUBMISSION : shell script containing all the commands to set up the virtual environment from that file as well as activate it. **File name: your_name_bonus.sh ,**together with a screenshot of execution on your computer.

Sample of a UNIX Shell Script:

```
#!/bin/bash

python3 -m pip install virtualenv

python3 -m virtualenv virtual source virtual/bin/activate

pip install <some libraries> python <filename.py>
```