42 ARTIFICIAL INTELLIGENCE

Python & ML - Module 04

Pandas

Summary: Today you will learn how to use a Python library that will allow you to manipulate dataframes.

Chapter I

Common Instructions

- The version of Python recommended to use is 3.7, you can check the version of Python with the following command: `python -V`
- The norm: during this bootcamp you will follow the PEP 8 standards. You can install pycodestyle which is a tool to check your Python code.
- The function `eval` is never allowed.
- The exercises are ordered from the easiest to the hardest.
- Your exercises are going to be evaluated by someone else, so make sure that your variable names and function names are appropriate and civil.
- Your manual is the internet.
- You can also ask questions in the `#bootcamps` channel in the 42AI or 42born2code.
- If you find any issue or mistakes in the subject please create an issue on 42AI repository on Github.
- We encourage you to create test programs for your project even though this work **won't have to be submitted and won't be graded**. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be run after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Contents

Chapter II

Exercise 00

	Exercise : 00
FileLoader	
Turn-in directory : $ex00/$	
Files to turn in : `FileLoader.py`	
Forbidden functions : `None`	

Objective

The goal of this exercise is to create a Fileloader class containing a load and a display method.

Instructions

Write a class named `FileLoader` which implements the following methods:

- `load(path)`: takes as an argument the file path of the dataset to load, displays a message specifying the dimensions of the dataset (e.g. 340 x 500) and returns the dataset loaded as a pandas.DataFrame,
- `display(df, n)`: takes a pandas.DataFrame and an integer as arguments, displays the first n rows of the dataset if n is positive, or the last n rows if n is negative.

`FileLoader` object should not raise any exceptions (wrong path, file does not exist, parameters different than a string ...).

Examples

```
from FileLoader import FileLoader
loader = FileLoader()
data = loader.load("../data/adult_data.csv")
Loading dataset of dimensions 32561 x 15
loader.display(data, 12)
age         workclass  fnlwgt  ... hours-per-week  native-country salary
0    39         State-gov   77516  ...             40   United-States  <=50K
1    50  Self-emp-not-inc   83311  ...             13   United-States  <=50K
2    38           Private  215646  ...             40   United-States  <=50K
3    53           Private  234721  ...             40   United-States  <=50K
4    28           Private  338409  ...             40            Cuba  <=50K
5    37           Private  284582  ...             40   United-States  <=50K
6    49           Private  160187  ...             16         Jamaica  <=50K
7    52  Self-emp-not-inc  209642  ...             45   United-States   >50K
8    31           Private   45781  ...             50   United-States   >50K
9    42           Private  159449  ...             40   United-States   >50K
10   37           Private  280464  ...             80   United-States   >50K
11   30         State-gov  141297  ...             40           India   >50K

[12 rows x 15 columns]
```

NB: Your terminal may display more columns if the window is wider.

Chapter III

Exercise 01

	Exercise : 01
YoungestFellah	
Turn-in directory : $ex01/$	
Files to turn in : `FileLoader.py, YoungestFellah.py`	
Forbidden functions : `None`	

Objective

The goal of this exercise is to create a function that will return a dictionary containing the age of the youngest woman and the youngest man who took part in the Olympics a given year.

Instructions

This exercise uses the following dataset: `athlete_events.csv`.

Write a function `youngestFellah` that takes two arguments as describe by the function signature:

```
def youngfellah(df, year):
        """
        Get the name of the youngest woman and man for the given year.
        Args:
                df: pandas.DataFrame object containing the dataset.
                year: integer corresponding to a year.
        Returns:
                dct: dictionary with 2 keys for female and male athlete.
        """
```

The function returns a dictionary containing the age of the youngest woman and man who took part in the Olympics on that year. Otherwise the same dictionary with 'nan' is returned. The name of the dictionary's keys is up to you, but it must be self-explanatory.

Examples

```
from FileLoader import FileLoader
loader = FileLoader()
data = loader.load('../data/athlete_events.csv')
# Output
Loading dataset of dimensions 271116 x 15

from YoungestFellah import youngestfellah
youngestfellah(data, 2004)
# Output
{'f': 13.0, 'm': 14.0}

youngestfellah(data, 1991)
# Output
{'f': 'nan', 'm': 'nan'}
```

Chapter IV

Exercise 02

	Exercise : 02
ProportionBySport	
Turn-in directory : $ex02/$	
Files to turn in : `FileLoader.py, ProportionBySport.py`	
Forbidden functions : `None`	

Objective

The goal of this exercise is to create a function displaying the proportion of participants who played a given sport, among the participants of a given genders.

Instructions

This exercise uses the dataset `athlete_events.csv`.

Write a function `proportionBySport` that takes four arguments:

- a pandas.DataFrame of the dataset,
- an olympic year,
- a sport,
- a gender.

The function returns a float corresponding to the proportion (percentage) of participants who played the given sport among the participants of the given gender.

The function answers questions like the following: "What was the percentage of female basketball players among all the female participants of the 2016 Olympics?"

Of course, the function should not raise any exceptions.

Here and further, if needed, drop duplicated sports people to count only unique ones. Beware to call the dropping function at the right moment and with the right parameters, in order not to omit any individuals.

Examples

```
from FileLoader import FileLoader
loader = FileLoader()
data = loader.load('../data/athlete_events.csv')
# Output
Loading dataset of dimensions 271116 x 15

from ProportionBySport import proportionBySport
proportionBySport(data, 2004, 'Tennis', 'F')
# Output
0.01935634328358209
```

We assume that we are always using appropriate arguments as input, and thus do not need to handle input errors.

Chapter V

Exercise 03

	Exercise : 03
HowManyMedals	
Turn-in directory : $ex03/$	
Files to turn in : `FileLoader.py, HowManyMedals.py`	
Forbidden functions : `None`	

Objective

The goal of this exercise is to implement a function that will return a dictionary of dictionaries giving the number and type of medals for each year during which the participant won medals.

Instructions

This exercise uses the following dataset: `athlete_events.csv`.

Write a function `howManyMedals` that takes two arguments:

- a pandas.DataFrame which contains the dataset,
- a participant name.

The function returns a dictionary of dictionaries giving the number and type of medals for each year during which the participant won medals. The keys of the main dictionary are the Olympic games years. In each year's dictionary, the keys are 'G', 'S', 'B' corresponding to the type of medals won (gold, silver, bronze). The innermost values correspond to the number of medals of a given type won for a given year.

The function should not raise any exceptions.

Examples

```
from FileLoader import FileLoader
loader = FileLoader()
data = loader.load('../data/athlete_events.csv')
# Output
Loading dataset of dimensions 271116 x 15

from HowManyMedals import howManyMedals
howManyMedals(data, 'Kjetil Andr Aamodt')
# Output
{1992: {'G': 1, 'S': 0, 'B': 1},
 1994: {'G': 0, 'S': 2, 'B': 1},
 1998: {'G': 0, 'S': 0, 'B': 0},
 2002: {'G': 2, 'S': 0, 'B': 0},
 2006: {'G': 1, 'S': 0, 'B': 0}}
```

Chapter VI

Exercise 04

	Exercise : 04
SpatioTemporalData	
Turn-in directory : $ex04/$	
Files to turn in : `FileLoader.py, SpatioTemporalData.py`	
Forbidden functions : `None`	

Objective

The goal of this exercise is to implement a class called `SpatioTemporalData` that takes a dataset (pandas.DataFrame) as argument in its constructor and implements two methods.

Instructions

This exercise uses the dataset `athlete_events.csv`.

Write a class called `SpatioTemporalData` that takes a dataset (pandas.DataFrame) as argument in its constructor and implements the following methods:

- `when(location)`: takes a location as an argument and returns a list containing the years where games were held in the given location,
- `where(date)`: takes a date as an argument and returns the location where the Olympics took place in the given year.

No exceptions should be risen by `where` and `when`.

Examples

```
from FileLoader import FileLoader
loader = FileLoader()
data = loader.load('../data/athlete_events.csv')
# Output
Loading dataset of dimensions 271116 x 15

from SpatioTemporalData import SpatioTemporalData
sp = SpatioTemporalData(data)
sp.where(1896)
# Output
['Athina']

sp.where(2016)
# Output
['Rio de Janeiro']

sp.when('Athina')
# Output
[2004, 1906, 1896]

sp.when('Paris')
# Output
[1900, 1924]
```

Chapter VII

Exercise 05

	Exercise : 05
HowManyMedalsByCountry	
Turn-in directory : $ex05/$	
Files to turn in : `FileLoader.py, HowManyMedalsByCountry.py`	
Forbidden functions : `None`	

Objective

The goal of this exercise is to write a function that returns a dictionary of dictionaries giving the number and type of medal for each competition where the country delegation earned medals.

Instructions

This exercise uses the following dataset: `athlete_events.csv`

Write a function `howManyMedalsByCountry` that takes two arguments:

- a pandas.DataFrame which contains the dataset
- a country name.

The function returns a dictionary of dictionaries giving the number and type of medal for each competition where the country delegation earned medals. The keys of the main dictionary are the Olympic games' years. In each year's dictionary, the key are 'G', 'S', 'B' corresponding to the type of medals won.

Duplicated medals per team games should be handled and not counted twice. Plus, function should not raise any exceptions.

Examples

```
>>> from FileLoader import FileLoader
>>> loader = FileLoader()
>>> data = loader.load('../data/athlete_events.csv')
Loading dataset of dimensions 271116 x 15
>>> from HowManyMedalsByCountry import howManyMedalsByCountry
>>> howManyMedalsByCountry(data, 'Martian Federation')
{2192: {'G': 17, 'S': 14, 'B': 23}, 2196: {'G': 8, 'S': 21, 'B': 19}, 2200: {'G': 26, 'S': 19, 'B': 7}}
```

You probably guessed by now that we gave up providing real examples...

If you want real examples, you can easily look online. Do beware that some medals might be awarded or removed years after the games are over, for example if a previous medallist was found to have cheated and is sanctioned. The `athlete_events.csv` dataset might not always take these posterior changes into account.

Chapter VIII

Exercise 06

	Exercise : 06
MyPlotLib	
Turn-in directory : $ex06/$	
Files to turn in : `MyPlotLib.py`	
Forbidden functions : `None`	

Objective

The goal the exercise is to introduce plotting methods among the different libraries Pandas, Matplotlib, Seaborn or Scipy.

Instructions

This exercise uses the following dataset: `athlete_events.csv`

Write a class called `MyPlotLib`. This class implements different plotting methods, each of which take two arguments:

- a pandas.DataFrame which contains the dataset,
- a list of feature names.
- `histogram(data, features)`: plots one histogram for each numerical feature in the list,
- `density(data, features)`: plots the density curve of each numerical feature in the list,
- `pair_plot(data, features)`: plots a matrix of subplots (also called scatter plot matrix). On each subplot shows a scatter plot of one numerical variable against another one. The main diagonal of this matrix shows simple histograms.
- `box_plot(data, features)`: displays a box plot for each numerical variable in the dataset.

As usual, the different methods should not raise any exceptions.

Examples

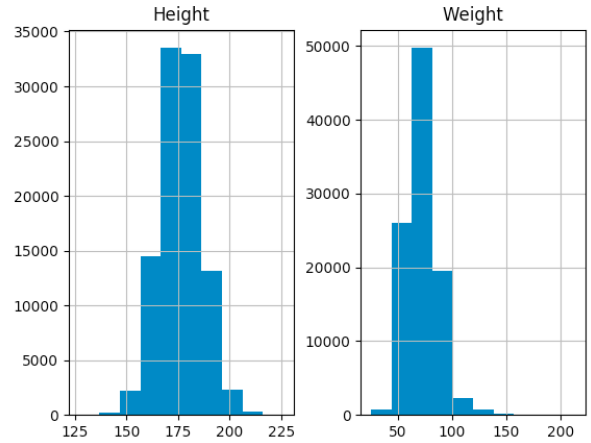

Figure VIII.1: histogram

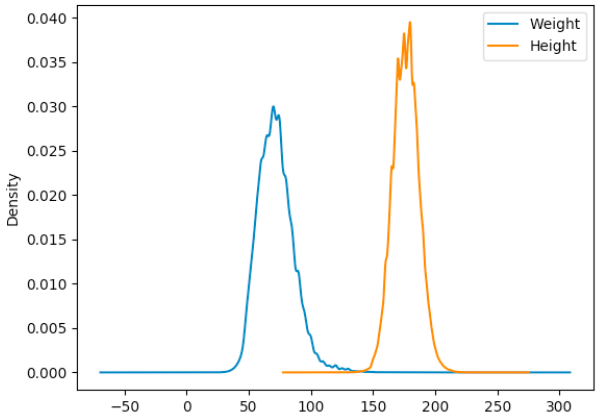

Figure VIII.2: density

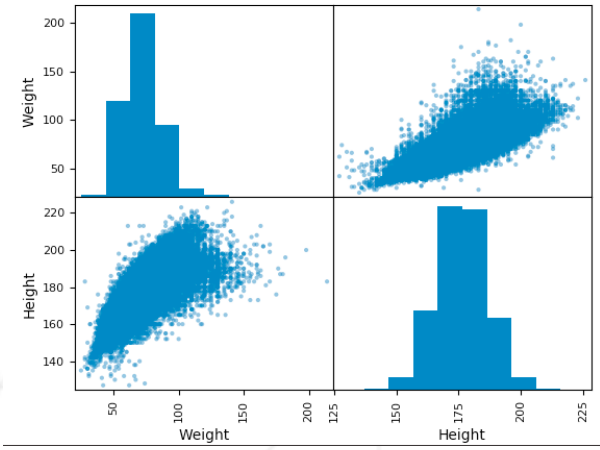

Figure VIII.3: pair plot

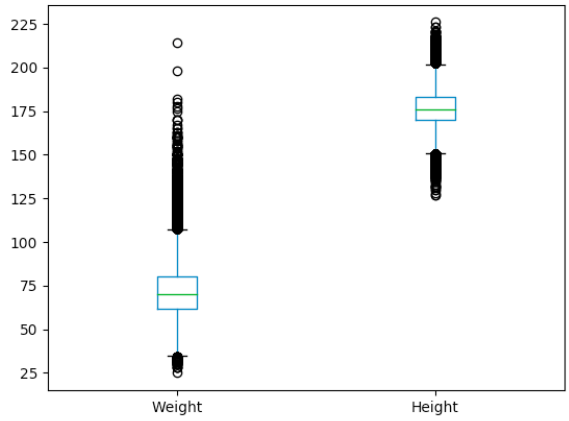

Figure VIII.4: box plot

Chapter IX

Exercise 07

	Exercise : 07
Komparator	
Turn-in directory : $ex07/$	
Files to turn in : `Komparator.py, MyPlotLib.py`	
Forbidden functions : `None`	

Objective

The goal the exercise is to introduce plotting methods among the different libraries Pandas, Matplotlib, Seaborn or Scipy.

Instructions

This exercise uses the following dataset: `athlete_events.csv`.

Write a class called `Komparator` whose constructor takes as an argument a pandas.DataFrame which contains the dataset. The class must implement the following methods, which take as input two variable names:

- `compare_box_plots(categorical_var, numerical_var)`: displays a box plot with several boxes to compare how the distribution of the numerical variable changes if we only consider the subpopulation which belongs to each category. There should be as many boxes as categories. For example, with Sex and Height, we would compare the height distributions of men vs. women with two boxes on the same graph,
- `density(categorical_var, numerical_var)`: displays the density of the numerical variable. Each subpopulation should be represented by a separate curve on the graph,
- `compare_histograms(categorical_var, numerical_var)`: plots the numerical variable in a separate histogram for each category. As an extra, you can use overlapping histograms with a color code (but no extra point will be granted!).

No execptions should be risen by any methods.

Contact

You can contact 42AI association by email: contact@42ai.fr
You can join the association on 42AI slack and/or posutale to one of the association teams.

Acknowledgements

The modules Python & ML is the result of a collective work, we would like to thanks:

- Maxime Choulika (cmaxime),
- Pierre Peigné (ppeigne),
- Matthieu David (mdavid).

who supervised the creation, the enhancement and this present transcription.

- Amric Trudel (amric@42ai.fr)
- Baptiste Lefeuvre (blefeuvr@student.42.fr)
- Mathilde Boivin (mboivin@student.42.fr)
- Tristan Duquesne (tduquesn@student.42.fr)
- Quentin Feuillade Montixi (qfeuilla@student.42.fr)

for your investment for the creation and development of these modules.

- Barthélémy Leveque (bleveque@student.42.fr)
- Remy Oster (roster@student.42.fr)
- Quentin Bragard (qbragard@student.42.fr)
- Marie Dufourq (madufour@student.42.fr)
- Adrien Vardon (advardon@student.42.fr)

who betatest the first version of the modules of Machine Learning.