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 gigantic amounts of data: Pandas.

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, it is recommended to follow the PEP 8 standards, though it is not mandatory. 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.
- If you are a student from 42, you can access our Discord server on 42 student's associations portal and ask your questions to your peers in the dedicated Bootcamp channel.
- You can learn more about 42 Artificial Intelligence by visiting our website.
- If you find any issue or mistake 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.
- We are constantly looking to improve these bootcamps, and your feedbacks are essential for us to do so !
  You can tell us more about your experience with this module by filling this form.
  Thank you in advance and good luck for this bootcamp !

Contents

Chapter II

Exercise 00

	Exercise : 00
FileLoader	
Turn-in directory : *ex*00/	
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(self, 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(self, 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")
# Output
Loading dataset of dimensions 32561 x 15


loader.display(data, 12)
# Output
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 : *ex*01/	
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 for a given year.

Instructions

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

Write a function `youngest_fellah` that takes two arguments:

- a pandas.DataFrame which contains the dataset
- an Olympic year.

The function returns a dictionary containing the age of the youngest woman and man who took part in the Olympics on that year. The names of the dictionary's keys are up to you, but they must be explicit and 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 youngest_fellah
youngest_fellah(data, 2004)
# Output
{’f’: 13.0, ’m’: 14.0}
```

Chapter IV

Exercise 02

	Exercise : 02
ProportionBySport	
Turn-in directory : *ex*02/	
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 gender.

Instructions

This exercise uses the dataset `athlete_events.csv`.

Write a function `proportion_by_sport` 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 female participants in the 2016 Olympics?"

> Here and later on, if needed, drop duplicate 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 individual.

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 proportion_by_sport
proportion_by_sport(data, 2004, 'Tennis', 'F')
# Output
0.019302325581395347
```

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

Chapter V

Exercise 03

	Exercise : 03
HowManyMedals	
Turn-in directory : *ex*03/	
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 types of medals for each year during which the participant won medals.

Instructions

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

Write a function `how_many_medals` that takes two arguments:

- a pandas.DataFrame which contains the dataset,
- a participant's 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 years of the Olympic games.

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.

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 how_many_medals
how_many_medals(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 : *ex*04/	
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.

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 : *ex*05/	
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 types of medals for each competition where a given country delegation earned medals.

Instructions

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

Write a function `how_many_medals_by_country` 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 types of medals 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 keys are 'G', 'S', 'B' corresponding to the types of medals won.
Duplicated medals per team games should be handled and not counted twice.

Hint: You may find this list to be of some use.

```
  team_sports = [’Basketball’, ’Football’,  ’Tug-Of-War’, ’Badminton’, ’Sailing’,
                ’Handball’, ’Water Polo’, ’Hockey’, ’Rowing’, ’Bobsleigh’, ’Softball’,
                ’Volleyball’, ’Synchronized Swimming’, ’Baseball’, ’Rugby Sevens’,
                ’Rugby’, ’Lacrosse’, ’Polo’]
```

Examples

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


from HowManyMedalsByCountry import how_many_medals_by_country
how_many_medals_by_country(data, ’Martian Federation’)
# Output
{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.

The `athlete_events.csv` dataset might not always take these posterior changes into account.

Chapter VIII

Exercise 06

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

Objective

The goal of this exercise is to introduce you to plotting methods using different libraries like 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 takes two arguments:

- a pandas.DataFrame which contains the dataset,
- a list of features names.

`What is a feature?`

- `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.

Examples

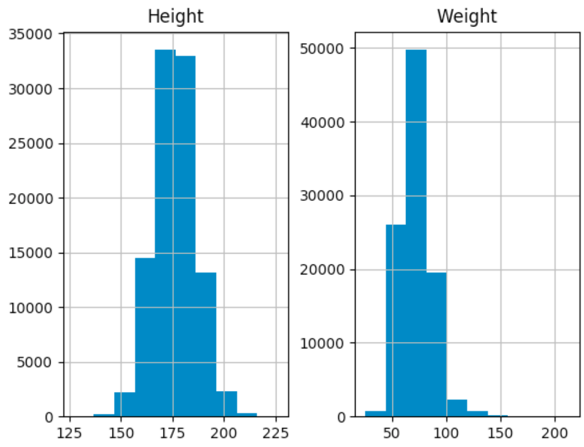

Figure VIII.1: histogram

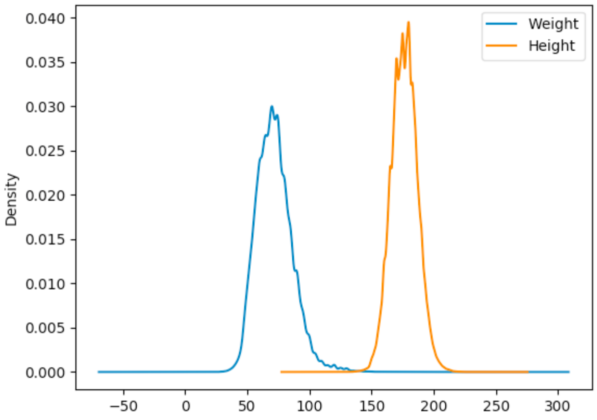

Figure VIII.2: density

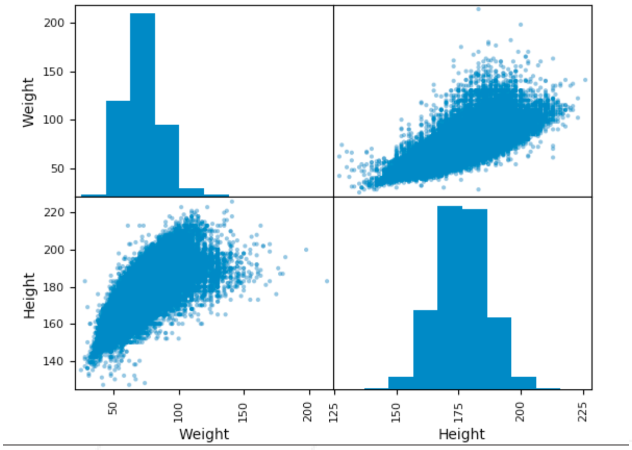

Figure VIII.3: pair plot

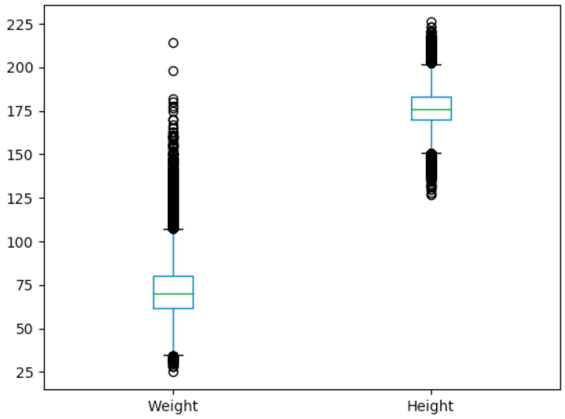

Figure VIII.4: box plot

Chapter IX

Exercise 07

	Exercise : 07
Komparator	
Turn-in directory : *ex*07/	
Files to turn in : `Komparator.py, MyPlotLib.py (optional)`	
Forbidden functions : `None`	

Objective

The goal of this 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(self, categorical_var, numerical_var)`: displays a series of box plots 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 box plots as categories. For example, with Sex and Height, we would compare the height distributions of men vs. women with two box plots.

- `density(self, 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(self, 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.

BONUS: Your functions can also accept a list of numerical variables (instead of just one), and output a comparison plot for each variable in the list.

Contact

You can contact 42AI by email: contact@42ai.fr

Thank you for attending 42AI's Python Bootcamp module04 !

Acknowledgements

The Python bootcamp is the result of a collective work, for which we would like to thank:

- Maxime Choulika (cmaxime),
- Pierre Peigné (ppeigne, pierre@42ai.fr),
- Matthieu David (mdavid, matthieu@42ai.fr),
- Quentin Feuillade–Montixi (qfeuilla, quentin@42ai.fr)
- Mathieu Perez (maperez, mathieu.perez@42ai.fr)

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

- Louis Develle (ldevelle, louis@42ai.fr)
- Augustin Lopez (aulopez)
- Luc Lenotre (llenotre)
- Owen Roberts (oroberts)
- Thomas Flahault (thflahau)
- Amric Trudel (amric@42ai.fr)
- Baptiste Lefeuvre (blefeuvr@student.42.fr)
- Mathilde Boivin (mboivin@student.42.fr)
- Tristan Duquesne (tduquesn@student.42.fr)

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

- All prior participants who took a moment to provide their feedbacks, and help us improve these bootcamps !