

Bootcamp

Python



Day04 - Pandas

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

Notions of the day

Pandas ! And Bamboos !

General rules

- Use the Pandas Library.
- The version of python 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 Pep8 standards <https://www.python.org/dev/peps/pep-0008/>
- The function eval is never allowed.
- The exercices are ordered from the easiest to the hardest.
- Your exercices are going to be evaluated by someone else so make sure that variables and functions names are appropriated.
- Your man is internet.
- You can also ask question in the dedicated channel in Slack: 42-ai.slack.com.
- If you find any issue or mistakes in the subject please create an issue on our dedicated repository on Github: https://github.com/42-AI/bootcamp_python/issues.

Helper

```
pip install pandas
```

For this day you will use the dataset athlete_events.csv provided at the root in the data folder.

Exercise 00 - FileLoader

Exercise 01 - YoungestFellah

Exercise 02 - ProportionBySport

Exercise 03 - HowManyMedals

Exercise 04 - SpatioTemporalData

Exercise 05 - HowManyMedalsByCountry

Exercise 06 - MyPlotLib

Exercise 07 - Komparator

Exercise 00 - FileLoader

Turnin directory :	ex00
Files to turn in :	FileLoader.py
Allowed Libraries :	Pandas
Remarks :	Be as lazy as possible...

Write a class named **FileLoader** which implement the following methods:

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

```
>>> 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        State-gov 77516 ...          40 United-States <=50K
1        Self-emp-not-inc 83311 ...       13 United-States <=50K
2         Private 215646 ...          40 United-States <=50K
3         Private 234721 ...          40 United-States <=50K
4         Private 338409 ...          40          Cuba <=50K
5         Private 284582 ...          40 United-States <=50K
6         Private 160187 ...          16        Jamaica <=50K
7        Self-emp-not-inc 209642 ...         45 United-States >50K
8         Private 45781 ...          50 United-States >50K
9         Private 159449 ...          40 United-States >50K
10        Private 280464 ...          80 United-States >50K
11        State-gov 141297 ...          40        India >50K
[12 rows x 15 columns]
```

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

Exercise 01 - YoungestFellah

Turnin directory :	ex01
Files to turn in :	FileLoader.py, YoungestFellah.py
Allowed Libraries :	Pandas
Remarks :	n/a

This exercise uses the following dataset: athlete_events.csv

Write a function **youngestFellah** which 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 name of the dictionary's keys is up to you, but it must be self-explanatory.

```
>>> from FileLoader import FileLoader
>>> loader = FileLoader()
>>> data = loader.load('../data/athlete_events.csv')
Loading dataset of dimensions 271116 x 15
>>> from YoungestFellah import youngestFellah
>>> youngestFellah(data, 2004)
{'f': 13.0, 'm': 14.0}
```

Exercise 02 - ProportionBySport

Turnin directory :	ex02
Files to turn in :	FileLoader.py, ProportionBySport.py
Allowed libraries :	Pandas
Remarks :	n/a

This exercise uses the dataset athlete_events.csv

Write a function `__proportionBySport__` which 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?"

Hint: here and further, if needed, drop duplicated sportsmen to count only unique ones.

```
>>> from FileLoader import FileLoader
>>> loader = FileLoader()
>>> data = loader.load('../data/athlete_events.csv')
Loading dataset of dimensions 271116 x 15
>>> from ProportionBySport import proportionBySport
>>> proportionBySport(data, 2004, 'Tennis', 'F')
0.01935634328358209
```

We assume that we are always using good arguments.

Exercise 3 - HowManyMedals

Turnin direcctory :	ex03
Files to turn in :	FileLoader.py, HowManyMedals.py
Allowed Libraries :	Pandas
Remarks :	n/a

This exercise uses the following dataset: athlete_events.csv

Write a function **howManyMedals** which 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 key are 'G', 'S', 'B' corresponding to the type of medals won (gold, silver, bronze).

```
>>> from FileLoader import FileLoader
>>> loader = FileLoader()
>>> data = loader.load('../data/athlete_events.csv')
Loading dataset of dimensions 271116 x 15
>>> from HowManyMedals import howManyMedals
>>> howManyMedals(data, 'Kjetil Andr Aamodt')
{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}}
```

Exercise 04 - SpatioTemporalData

Turnin directory :	ex04
Files to turn in :	FileLoader.py, SpatioTemporalData.py
Allowed Libraries :	Pandas
Remarks :	n/a

This exercise uses the dataset athlete_events.csv

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

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

```
>>> from FileLoader import FileLoader
>>> loader = FileLoader()
>>> data = loader.load('../data/athlete_events.csv')
Loading dataset of dimensions 271116 x 15
>>> from SpatioTemporalData import SpatioTemporalData
>>> sp = SpatioTemporalData(data)
>>> sp.where(1896)
['Athina']
>>> sp.where(2016)
['Rio de Janeiro']
>>> sp.when('Athina')
[2004, 1906, 1896]
>>> sp.when('Paris')
[1900, 1924]
```

Exercise 05 - HowManyMedalsByCountry

Turnin directory :	ex05
Files to turn in :	FileLoader.py, HowManyMedalsByCountry.py
Allowed Libraries :	Pandas
Remarks :	n/a

This exercise uses the following dataset: athlete_events.csv

Write a function **howManyMedalsByCountry** which takes two arguments:
* a pandas.DataFrame which contains the dataset
* a country name

The function return a dictionary of dictionaries giving the number and type of medal for each competition where the country got 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.

```
>>> 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...

Exercise 06 - MyPlotLib

Turnin directory :	ex06
Files to turn in :	MyPlotLib.py
Allowed Libraries :	Pandas, Matplotlib, Seaborn, Scipy
Remarks :	The less work you do, the better! You don't necessarily need all those libraries to complete the exercise.

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:

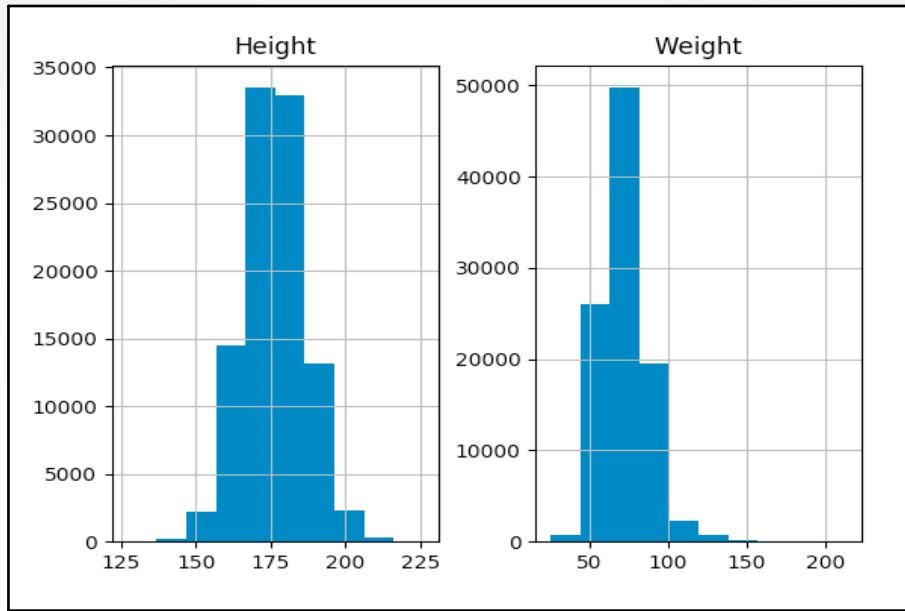
- * a pandas.DataFrame which contains the dataset
- * a list of feature names

Hint: What is a feature?
<https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114>

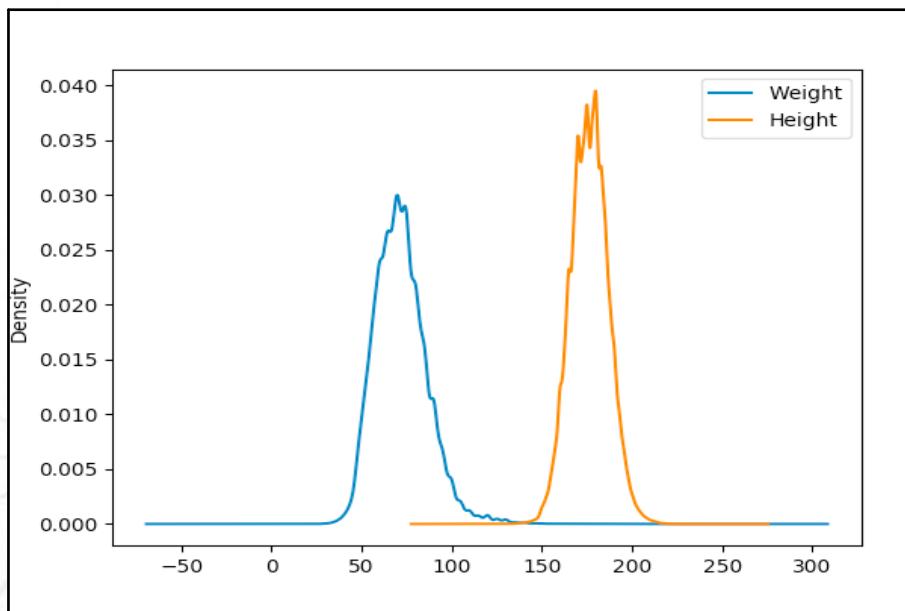
- **histogram(data, features)** : plots one histogram for each numerical feature in the list
- **density(data, features)** : plots the density curve of each numerical feature in the list
- **pair_plot(data, features)** : plots a matrix of subplots (also called scatter plot matrix). On each subplot shows 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:

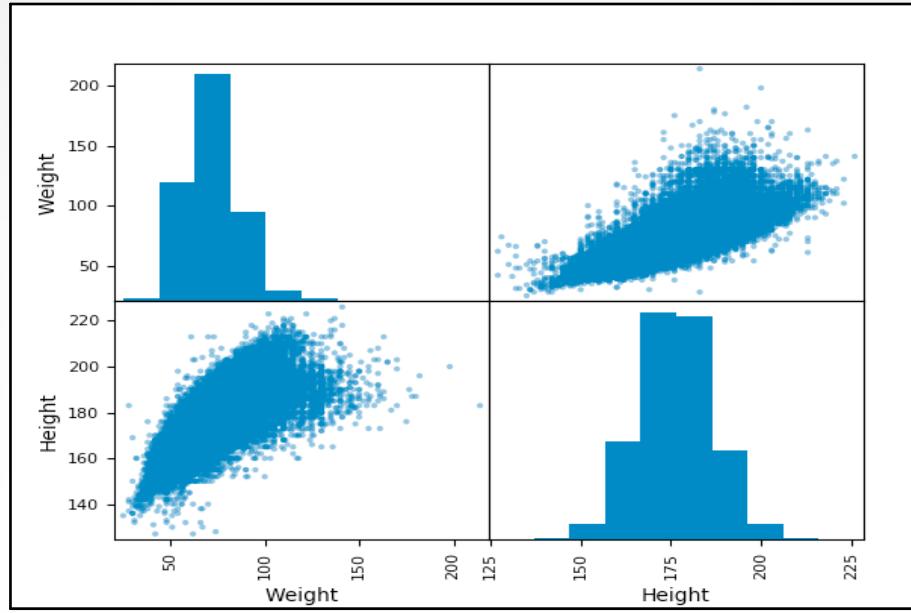
- histogram:



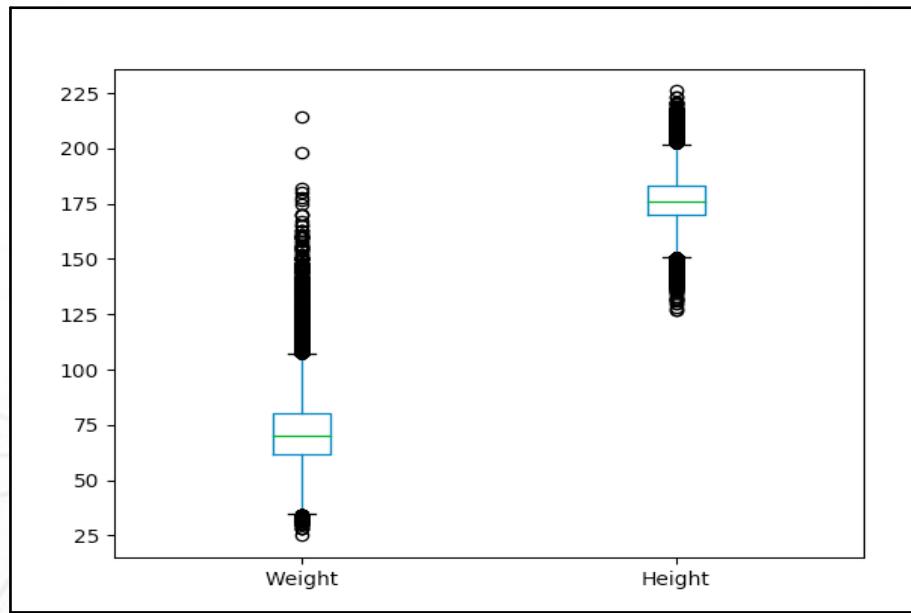
- density:



- pair_plot:



- box_plot:



Exercise 07 - Komparator

Turnin directory :	ex07
Files to turn in :	Komparator.py, MyPlotLib.py (optional)
Allowed Libraries :	Pandas, Matplotlib, Seaborn, Scipy
Remarks :	The less work you do, the better! You don't necessarily need all those libraries to complete the exercise.

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 series of box plots to compare the distribution of the numerical variable in each possible value of the categorical variable. There should be as many box plots as there are possible values of the categorical variable. For example, with Sex and Height, we would compare the distribution of height between men and women with two box plots.
- **density(categorical_var, numerical_var)** : displays the density of the numerical variable, with a different curve for the subpopulation which belongs to each categorical variable.
- **compare_histograms(categorical_var, numerical_var)** : displays separate histograms of the numerical variable for each category represented in the categorical variable. As a bonus, you can make it display overlapping histograms of different colors.

BONUS: Your functions can also accept a list of numerical variables, and output comparison plots for each feature against different