T9 - Big Data

T-DAT-901

Recommender

Bootstrap

1.1.2

EPITECH.

Survive the Titanic

Given a table of boat's passengers, named *titanic.csv*, you are about to find out what could have possibly happened in the early morning hours of 15 April 1912.

When dealing with big data, it is a key issue not only to solve the problem, but to solve it within the lowest possible time and expense of resources. Favor native, efficient methods over generic code.

Getting the dataframes

First, load the data from the csv file into a panda dataframe.

Beware that some optional arguments should be given explicitly in order to avoid unpleasant surprises later on (such as delimiter, encoding, header, …).

Now play around with this object. Display the first ten rows, and find out how many people's ages are missing.

Terminal

```
~/T-DAT-901> python info_age.py
age 1046 non-null 263 null
```

Extracting statistics

Extract data and compute statistics about survivors only: how many are they, what is their average age and average fare price? Then, do the same with the victims and compare the results.

Terminal

```
~/T-DAT-901> python survived_only.py
total: 500
age: 28.918
fare: 49.361
```

Almost all methods in Pandas do not modify the object they are called on. Instead, they return a copy of the table with the intended modification. This is a golden rule that prevents unwanted alteration of the data.

GROUPS ALONG CHARACTERISTICS

Find out the odds of surviving, depending on various variables.

You should avoid inefficient external loops.

Display the probability of surviving a Titanic disaster for all the possible values of the following variables: sex, pclass and age. Why is the result for age quite useless?

Transform your data in order to get interesting information about the ages: group them into wide categories (like *0-9*, *10-19*, …) before making statistics. This is called **coding** the information.

You could use loops, but this would certainly become inefficient on a larger dataset.

WELCOME TO THE JUNGLE OF TRAIN SYSTEM DATA

In real life, data is not usually given to you in a neat, single file, but you have to merge together different sources.You will find the data scattered among several files in the repository *data_sncf.zip*.

WALKING THROUGH THE DATA

Find out if one can catch a train from a given place (Paris - Gare de l'est). For that purpose, you might need two different tables. Identify these tables, load them into dataframes and play around with them.

Reading parameters should probably be different from the previous part. Also, you are getting closer to big data - but not yet.

```
Terminal
~/T-DAT-901> python info_trains_part_1.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209129 entries, 0 to 209128
Data columns (total 9 columns):
trip_id 209129 non-null object
arrival_time 209129 non-null object
...
```

MERGING INFORMATION

Merge your two tables in such a way that stations are now identified by their full name. Check out for the names, indices and columns of the resulting new dataframe.

Merging dataframes may lead to automatic re-indexing, even with a clean join. Watch out for your index numbers.

Once you have a clean table, keep only the trains that leave Paris - Gare de l'est, in the morning before 10:00. Display, as a dataframe, their ids and departure time.

```
Terminal
~/T-DAT-901> python info_trains_part_2.py
<class 'pandas.core.frame.DataFrame'>
trip_id departure_time
143417 OCESN839553F4704738345 08:42:00
143472 OCESN839553F2802838334 08:42:00
143499 OCESN839561F0800838467 05:42:00
143590 OCESN839551F3403438301 06:34:00
143599 OCESN839589F0500538521 07:42:00
... ... ...
149365 OCESN839130F0700738016 07:54:00
149372 OCESN839130F0200238015 07:54:00
149379 OCESN839134F0400438032 09:23:00
149386 OCESN839132F0300338026 08:54:00
150356 OCESN839573F2102138483 08:21:00
[70 rows x 2 columns]
```