Data Science and Machine Learning Lab
Lab 02 - Pandas Introduction

Politecnico di Torino

Intro

The main objective of this laboratory is to put into practice what you have learned on the Pandas data analysis library. This laboratory will let you carry out a preparatory data exploration analysis on two real-world datasets, to practice with the Pandas data analysis library and its advanced functionalities. If you have any doubts about the Pandas APIs, you can check the course material or the official documentation.

1 Preliminary steps

1.1 Datasets

1.1.1 New York Point Of Interest

The *New York Point Of Interest* dataset collects a sub-sample of the point of interests (POI) available in the city of New York. It contains the coordinates of each point of interest and the category to which it belongs to, specifying the *type* of the POI, for each POI *category*. There are four possible categories of POIs: *amenity*, *shop*, *public_transport* and *highway*.
The fields available in the dataset are:

- `@id`: a unique id for each point of interest.
- `@lat`: latitude coordinate of the POI in decimal degrees.
- `@lon`: longitude coordinate of the POI in decimal degrees.
- `amenity`: if the POI category is *amenity* its type is reported in this field.
- `name`: name of the amenity.
- `shop`: if the POI category is *shop* its type is reported in this field.
- `public_transport`: if the POI category is *public transport* its type is reported in this field.
- `highway`: if the POI category is *highway* its type is reported in this field.

To identify which are the POIs belonging only to the *New York City municipality*, a further file is provided. The *New York City municipality POIs* metadata file is composed by a single column containing the Ids of the POI of interest of the municipality.
Then, a map of the New York municipality is provided as well as a .PNG file. For this map, the latitude boundaries are (40.5024225, 40.9139069), the longitude boundaries are (-74.2527909, -73.7001124). You will use this map as a background for various data visualizations.

The main dataset is provided in TSV (Tab Separated Values) format. This format is very similar to the CSV one, the only difference is the separator used to split the fields in a row: the separator is a TAB character. You can download a zip containing the three files at:

`https://github.com/dbdmg/data-science-lab/raw/master/datasets/NYC_POIs.zip`

1.1.2 Flight Delay Data

This dataset is made available by the Bureau of Transportation Statistics of the United States Department of Transportation.
Measuring the performance of flight carriers (e.g. American Airlines, EasyJet) is extremely important for the transportation department and, for this reason, all the information related to each flight are constantly monitored and collected in huge databases by the Department of Transportation. To the aim of this laboratory, just a small set of information has been extracted.
The dataset contains the Carrier On-Time Performance information collected from 01-01-2017 until 31-01-2017 for all the flights in the United States. Each row represents a flight in a specific day.
Some of the most useful fields in the dataset are:

- `FL_DATE`: day of the flight in format YYYY-mm-dd.
- `TAIL_NUM`: aircraft registration number, unique to a single aircraft.
- `UNIQUE_CARRIER`: flight carrier id.
- `FL_NUM`: number of the flight.
- `ORIGIN`: departure airport code.
- `DEST`: destination airport code.
- `CRS_DEP_TIME`: scheduled departure time (local time: HHMM) shown in the carriers' Computerized Reservations Systems (CRS)
- `DEP_TIME`: actual departure time (local time: HHMM)
- `DEP_DELAY`: overall delay at departure. Difference in minutes (floating point number) between scheduled and actual departure time. Early departures set to 0.
- `CRS_ARR_TIME`: scheduled arrival time (local time: HHMM) shown in the carriers' Computerized Reservations Systems (CRS)
- `ARR_TIME`: actual arrival time (local time: HHMM)
- `ARR_DELAY`: overall delay. Difference in minutes (floating point number) between scheduled and actual arrival time. Early arrivals show negative numbers.
- `CARRIER_DELAY`: delay in minutes (floating point number) caused by the carrier.
- `WEATHER_DELAY`: delay in minutes (floating point number) caused by the weather.
- `NAS_DELAY`: delay in minutes (floating point number) caused by the National Air System (NAS).
- `SECURITY_DELAY`: delay in minutes (floating point number) caused by the security.
- `LATE_AIRCRAFT_DELAY`: delay in minutes (floating point number) caused by the aircraft.

There are some other fields in this dataset. You can explore them, understand what they represents and whether they are significant or not for your analysis. Also, you can navigate the web page where the data has been collected from.

You can download the dataset at:

`https://github.com/dbdmg/data-science-lab/raw/master/datasets/US_FlightDelayData.zip`

2 Exercises

In this laboratory you have to practice with the Pandas data analysis library. Pandas is a very powerful library that provides high-performance, easy-to-use data structures and data analysis tools, allowing to simplify many of the tasks of the data scientist.

Info: All the exercises in this laboratory have to be solved using the Pandas library APIs whenever it is possible. Please consider that Pandas provides even plotting APIs, so you can try to solve plotting requests without the use of matplotlib. Further details about the Pandas plotting functionalities are available in the course material or in the `DataFrame.plotting` documentation

2.1 Data exploration of Point Of Interest

In this exercise, you will perform a simple data exploration task on the *New York Point Of Interest* dataset exploiting the Pandas library.

1. Load the *New York Point Of Interest* dataset exploiting Pandas APIs. Load the *NY Municipality POIs ID* metadata as well and filter out from the *New York Point Of Interest* data the records that do not belong to the New York municipality.
   - Which columns have been parsed?
   - Which is the type of the data inferred by Pandas?
2. For each column in the loaded dataset count the number of missing values.
   - What did you expect?
   - What can you infer from this first analysis?
3. Now, analyze the distribution of the POI types for each POI category. Point of interest categories are `amenity`, `shop`, `public_transport` and `highway`. For each of them, plot a histogram showing the distribution of the types of POI. Note that, for certain categories, due to the high number of types, bars and labels could not fit adequately in the figure. Hence, fix a threshold (a percentage one is better) and plot only the most frequent types.

   To simplify the subsequent analysis, use the retained top frequent POI types also for the following exercises.
4. Show the points of interest on the New York map for a given category (e.g. amenity). To do so, you have to define a new function (or a new class) that, given the name of the POI category is able to show a scatter plot of the locations of the POI types, onto the New York municipality map, with a different color for each category.
   - Are you able to identify areas in which the concentration of a specific POI type is higher then others?
   - How can you better characterize the POI distributions?

   Info: to display an image on an Axes object, you can use the matplotlib imshow fuction. Then, you can specify the Axes object as the plotting axes in pandas methods, using the `ax` argument.
5. Discretize the POIs by geographical position. Define a new function (or class) that is able to split the geographical position of the POIs using a grid. Once defined the grid over the New York municipality, the function has to assign each POI to the cell to which it belongs to. The result should be a Pandas DataFrame with a new column containing the cell_id for each POI.
6. Now you have to identify how many times a POI type is contained in each cell, for each category. Create a new DataFrame containing the `cell_id` as index, the POI types as columns and the count of the occurrences of each type, in each cell, as values.

7. For the categories *amenities* and *shop* identify if there exist a correlation between the location of different POI types. Compute the pairwise correlation between the columns of the cell-type DataFrame and show it through a heatmap chart.

    - Are you able to identify which are the POI types more correlated between each other? What does it mean?
    - Which are the more correlated tuples? Do they belong to the same category?

2.2 Data exploration and queries on Flight Delay Data

In this exercise, you will carry out some data exploration on the *Flight Delay* dataset. Just as before, you will use the Pandas library to answer different queries, exploiting crucial pandas concepts on indices, slicing types, pivoting and more.

1. Load the Flight Delay dataset exploiting Pandas APIs.

   > **Info:** take a closer look at the `parse_dates` argument of the `read_csv` method. Whenever your data come with timestamps of dates, working with Python's datetime structures becomes extremely useful.

2. Use the `info()` and `describe()` methods to analyze how your records are distributed. Before continuing, try to answer the following questions:
   - which type does each column have?
   - are there any missing values?
   - how many unique carriers are present?
   - how many unique airports are present?
   - from which time interval data were collected?
3. Filter out all canceled flights.
4. Use any pandas method and functionality to answer the following queries:
   - how many flights had each carrier operated?
   - for each carrier, compute the mean delay considering all possible reasons (due to the carrier, weather, etc.)
5. Add two new columns to your DataFrame:
   - `weekday`: it is the day of the week expressed as an integer number. Check out Pandas dayofweek attribute.
   - `delaydelta`: it is the difference between the arrival delay and the departure one.
6. Choose one of the visualization tools that you know and inspect the arrival delay as a function of the day of the week. Can you find any correlation?
7. Consider the weekend days only, compute, for each carrier, the mean arrival delay. Now consider the working days and compute, for each carrier, the mean arrival delay.
   Then, compare the delays in working days and in weekends for each company.
   - Are you able to identify companies that are delayed only in weekends or only in working days? Why?
8. Create a Pandas DataFrame with a multi-index composed of the columns: `tunique_carrier, origin, dest, fl_date`.
9. For each flight operated by American Airlines (AA) and Delta Airlines (DL), taken off from the Los Angeles International Airport (LAX) and for each date, display the departure time and delay.
10. For each flight that flew in the first week of the month, with LAX as destination airport, compute the mean arrival delay.
11. Generate a pivot table containing the number of departed flights for each carrier and for each day of the week and show it.
    Compute now the pairwise correlation between the carriers and show it on a heatmap.
    - What does this correlation matrix represent?

- Can you find any carrier with different flight plans?

12. Generate a pivot table containing the average arrival delay, for each carrier and for each day of the week and show it.
    Compute now the pairwise correlation between the carriers and show it on a heatmap.
    - What does this correlation matrix represent?
    - Can you find any carrier with different delay behaviors?
13. Using a pivot table, for the carriers HA, DL, AA and AS compute the average `deltadelay` for each day of the week. Then, display the results on a line plot, having a line per carrier and the weekday on the x-axis.