Coding for Data Science - Python Exercise Sessions

Contacts: sergio.picascia@unimi.it

Exam Modalities

1. **REGULAR**: take the exam in a regular exam session. You will be asked to complete different exercises in Python.
2. **PROJECT**: develop a project in Python, according to the requirements. Prepare a 10 minutes presentation for the discussion of your project.

Calendar

- Sep 23rd, 16:30-18:00 - *Introduction*
- Sep 30th, 16:30-18:00 - *Project Setup*
- Oct 7th, 16:30-18:00 - *Input/Output*
- Oct 14th, 16:30-18:00 - *Data Manipulation*
- Oct 21st, 16:30-18:00 - *Scientific Computing*
- Oct 28th, 16:30-18:00 - *Data Visualization*
- Nov 4th, 16:30-18:00 - *Dashboard*
- Nov 11th, 16:30-18:00 - *Project Finalisation*
- End of November (TBD) - *Evaluation*

NOTE: this is a tentative outline and and may be subject to changes.

Project Requirements

Each project should present the following characteristics:

1. usage of *GitHub*
2. correct modularisation
3. import and output of data
4. usage of a data manipulation library (e.g. *pandas*)
5. usage of a scientific computing library (e.g. *numPy*)
6. usage of a data visualization library (e.g. *matplotlib*)
7. BONUS - usage of a web app creation framework (e.g. *streamlit*)

Project Proposals

Choose among one of the following projects or propose your own.

1. Quiz

The purpose of the project is to automatically create multiple-choice film-related quizzes, exploiting data from IMDb. Each quiz must consist of one question and four possible answers, only one of which must be correct. The program that generates the quizzes must also implement a criterion to generate the possible answers proportional to the difficulty of the desired quiz. The program must also allow a human player to answer the quiz and must measure his/her performance with an overall score that takes into account the difficulty of each individual question. IMDb data can be acquired through specific APIs (such as Cinemagoer) or existing datasets, such as IMDb Dataset.

2. Hotels

The dataset chosen for the project contains the following information:

- `hotels.xlsx` : number of vacant rooms and unit cost of each room for 400 hotels;
- `guests.xlsx` : discount fraction for 4000 potential customers;
- `preferences.xlsx` : order of hotel preference for each customer.

The program must calculate the allocation of customers at hotels, considering the number of available rooms, the fact that each customer occupies exactly one room, and that each stay lasts only one night. The price paid by the customer is the unit price of the room discounted by the fraction of the discount to which the customer is entitled.

The program must implement four different allocation strategies:

- random: customers are randomly distributed to the rooms until the seats or customers are exhausted;
- customer preference: customers are served in order of reservation (the customer

number indicates the order) and are allocated to the hotel based on their preference, until the seats or customers are exhausted;
- price: the places in the hotel are distributed in order of price, starting with the cheapest hotel and following in order of reservation and preference until the places or customers are exhausted;
- availability: places in hotels are distributed in order of room availability, starting with the most roomy hotel and subordinately in order of reservation and preference until places or clients are exhausted.

Finally, the program must present and display a report of the result obtained, showing for each strategy the number of customers accommodated, the number of rooms occupied, the number of different hotels occupied, the total volume of business (total earnings of each hotel), and the degree of customer satisfaction (calculated according to the location of the hotel assigned to them with respect to their preferences).

3. Board Games

The BoardGameGeek (BGG) website provides various data and statistics on the enthusiastic board game hobby. BGG users comment on games and optionally assign them a liking score between 0 and 10.

The dataset contains millions of board game reviews. The purpose of the project is to produce a ranking of games in order of user liking. It should be noted that in order to obtain an adequate ranking it is necessary to consider not only the votes, but also the fact that different games may be associated with very different numbers of comments and thus votes from individual users. For a discussion of this point see, for example, How Not To Sort By Average Rating.

The project should propose its own strategy for sorting the games, also arguing its appropriateness in relation to the official BGG ranking, available online.

The project should also produce graphs comparing the proposed ranking with the ranking obtained only through the average rating of each game, showing the different distribution of scores and the differences found.

4. Weather

The dataset reports the temperature recorded in major cities around the world since 1750. Using this data, the project will need to provide an effective graphical visualization of the change in temperatures over time, highlighting the cities where the largest temperature ranges were recorded during different historical periods. For visualization of the data on a map, see geopandas.

The program will also suggest, depending on the period considered, the best route to follow for a traveler who intends to move from Beijing to Los Angeles by moving step by step to the warmest city among the 3 closest to him.

5. Animals

The dataset provides data about different animal species in order to classify them into 7 different classes, namely mammals, birds, reptiles, fish, amphibians, insects and invertebrates. Following an unsupervised approach, that is, without observing the class of each animal species, the project should aim at comparing the different species and group them using different clustering algorithms.

Comparing the result of each algorithm, show which clustering algorithm better approximates the classes provided by the dataset.

It is therefore required not only to define a methodology for comparing the clustering results with the expected classification, but also to briefly describe the distinguishing characteristics of each species cluster produced by the algorithm under evaluation.

6. Travel

Consider the dataset describing some of the world's major cities. Assume that it is always possible to travel from each city to the 3 nearest cities and that such travel takes 2 hours to the nearest city, 4 hours to the second nearest city, and 8 hours to the third nearest city. In addition, the trip takes an additional 2 hours if the destination city is in another country than the starting city and an additional 2 hours if the destination city has more than 200,000 inhabitants.

Starting in London and always traveling east, is it possible to travel around the world by returning to London in 80 days? How long does this take at a minimum?

Running Example - Information Retrieval

CISI is a dataset for evaluating the performance of an Information Retrieval system. Specifically, CISI provides a set of text documents and queries, reporting the relevance of different documents for each query. The project aims at using such dataset to evaluate the performance of a document retrieval system based on natural language queries.

It is therefore required to construct three simple search engines: the first based on random responses to queries; the second based on the frequency of terms in documents; and the third based on the relevance measure of terms in documents calculated with Tfldf.

Produce a report that comparatively evaluates the three systems, providing an indication of the most relevant causes of error for each of the three alternatives.

Python Installation

- Python Website: https://www.python.org/downloads/
- Anaconda: https://www.anaconda.com/download/

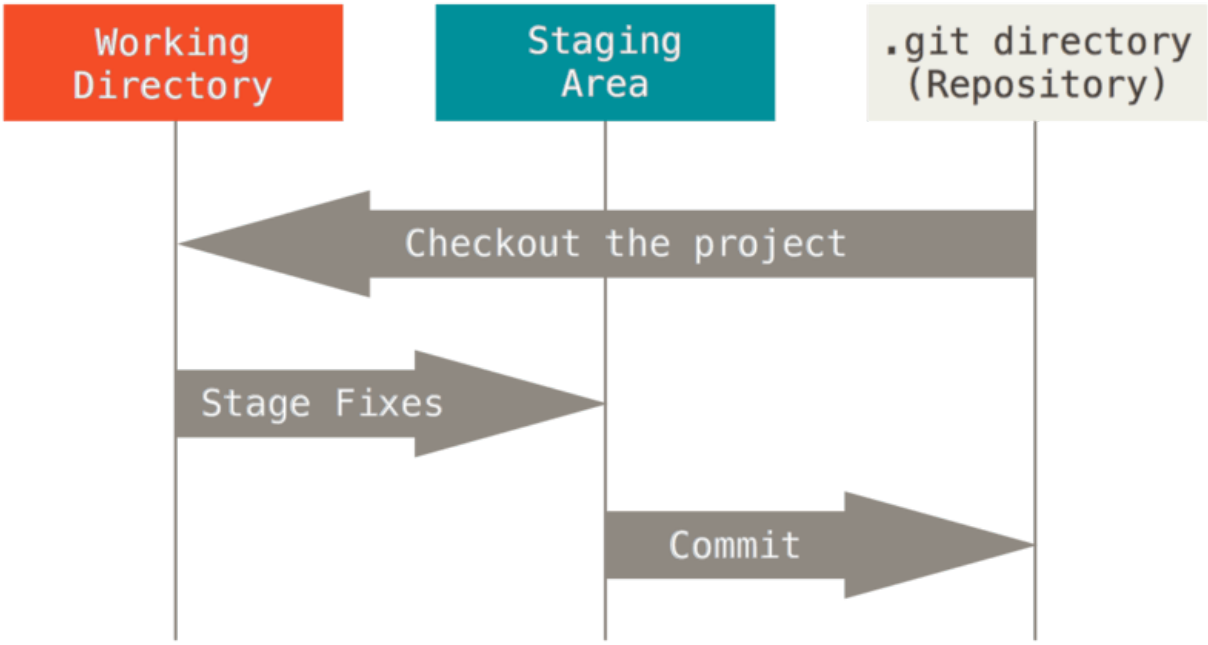
Version Control

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. A Version Control System (VCS) allows you to revert selected files back to a previous state, revert the entire project (**repository**) back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more. Using a VCS also generally means that if you screw things up or lose files, you can easily recover.

Git

Git is a Distributed Version Control Systems (DVCSs): in this case, clients don't just check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history. Thus, if any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data. With Git, every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot. To be efficient, if files have not changed, Git doesn't store the file again, just a link to the previous identical file it has already stored. Git thinks about its data more like a stream of snapshots.

Git has three main states that your files can reside in: modified, staged, and committed. *Modified* means that you have changed the file but have not committed it to your database yet. *Staged* means that you have marked a modified file in its current version to go into your next commit snapshot. *Committed* means that the data is safely stored in your local database. This leads us to the three main sections of a Git project: the working tree, the staging area, and the Git directory. The *working tree* is a single checkout of one version of the project; these files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify. The *staging area* is a file, generally contained in your Git directory, that stores information about what will go into your next commit. The *Git directory* is where Git stores the metadata and object database for your project; this is the most important part of Git, and it is what is copied when you clone a repository from another computer.



The basic Git workflow goes something like this:

1. *modify* files in the working tree;
2. selectively *stage* just those changes you want to be part of your next commit, which adds only those changes to the staging area;
3. *commit*, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory;
4. *push*, upload local repository content to a remote repository.

GitHub

GitHub is a for-profit company that offers a cloud-based Git repository hosting service. Essentially, it makes it a lot easier for individuals and teams to use Git for version control and collaboration. GitHub's interface is user-friendly enough so even novice coders can take advantage of Git. Without GitHub, using Git generally requires a bit more technical savvy and use of the command line.