Data Science - Group Delivery Guide
August 2020 - The Bridge

INSTRUCTOR: Gabriel Vázquez Torres
gabriel@thebridgeschool.es

TEACHER: Clara Piniella Martinez
clara.piniella@thebridgeschool.es

Delivery explanation

This group delivery aims to practice different concepts about EDA and APIs. Also, the project will be presented.

Groups

There are four groups:

- **Group A**: Natalio, Tomás, Elsa y Rosario
- **Group B:** Cristina, Jose María y Jalex
- **Group C:** David, Valeria, Roxanna y Alba
- **Group D**: Mónica, Filipa y Jose Luis

Countries assigned

- **Group A**: Argentina, Russia, Colombia, Chile and Spain
- **Group B:** India, Peru, US, Francia and Spain
- **Group C:** Mexico, Holland, Brazil, Iran and Spain
- **Group D**: Portugal, Venezuela, Turkey, UK and Spain

Requirements

The next requirements are mandatories:

1. Each group must choose a person that will do the presentation.
2. Each group must choose another (not the same) person that will write the documentation.
3. All the participants must write code tasks. For this, in each function, apart from that the function must be well documented, it must contain the name of the student or students that have done the exercise. It must appear with the syntax "@alias_of_github".
4. It is mandatory that each group uses the software *trello* (or other related) to manage the tasks in different status: TODO, DOING and DONE. BACKLOG and REVIEW are optionals.
5. The delivery must be sent before 31/08/2020 at 23:59.
6. The delivery must be sent in a *.zip* file by email with this structure:
    a. A folder **src/** that contains all the source code.
    b. A folder **documentation/** that contains all the documents related to documentation.
    c. A folder **resources/** that contains other useful content (images,...)
    d. A folder **src/utils/** that contains all the modules used by the *main* file.
    e. A file **src/main.ipynb** that contains all the functionality. This file must only contain imports, pandas, matplotlib, requests,... and calls to your **src/utils/*** modules.
    f. There are, at least, four modules inside **src/utils/** :

i. “folders_tb.py” that contains the generic functionality related to open, create, read and write files.

ii. “visualization_tb.py” that contains the generic functionality related to pandas, matplotlib, seaborn and other libraries focus on visualizations.

iii. “mining_data_tb.py” that contains the generic functionality related to collect data, clean data and others (wrangling methods such us working with multiples jsons)

iv. “apis_tb.py” that contains the generic functionality related to working with APIs.

g. A file **src/api/server.py** that contains the functionality that starts the Flask API. There are two GET functions:

i. One that must allow to receive an ***group_id*** and will return a json with one attribute called “token” with the S value (explained below). This function must only return the S value in a json if the ***group_id*** received is equal to N (explained below). Otherwise, it must return a string with a message of error.

ii. Another that must allow you to receive a ***token_id*** value and, if ***token_id*** is equal to S, return the json that contains the information of the dataframe (df). Otherwise, return a string with a message of error.

1. N is the letter of your group concatenated with the sum of the ages of the participants. Examples: “A103”, “C86”.

2. S is the letter of your group concatenated with the multiplication of the DNIs of the participants of the group. Example: “B85918591859851”, “D635154795182”

7. The json returned (df) is different by group:

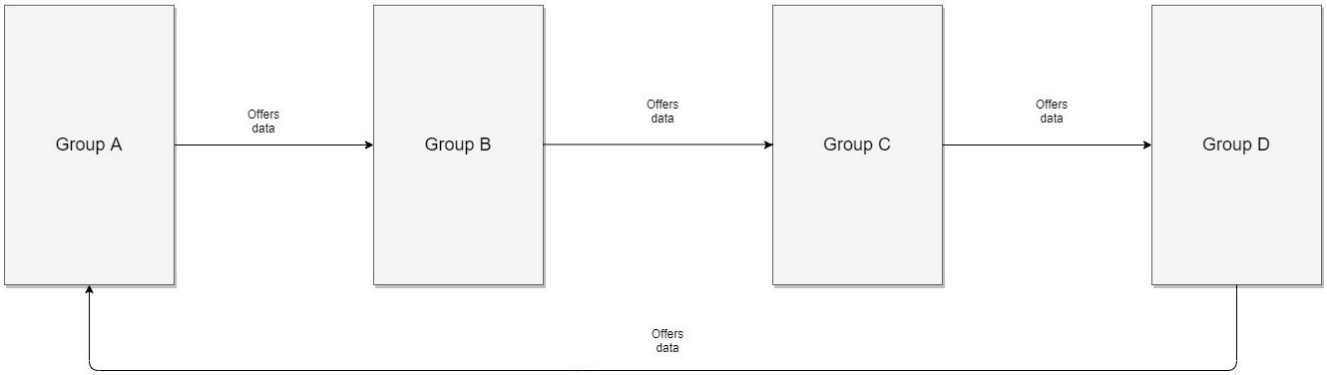
   a. **Group A**: it must return a json with one key *"c_averages"* that represents the mean of the "new_cases" of all of your countries.

   b. **Group B**: it must return a json with one key *"d_averages"* that represents the mean of the "new_deaths" of all of your countries.

   c. **Group C**: it must return a json with one key *"t_c_averages"* that represents the mean of the "total_cases" of all of your countries.

   d. **Group D**: it must return a json with one key *"t_d_averages"* that represents the mean of the "total_deaths" of all of your countries.

8. The groups are related as in the Figure 1:



Presentation

All groups must do a presentation about its project. The presenter of the group will use a presentation file to explain all the steps of the workflow with graphs.

The duration of the presentation won't be longer than 7 minutes so it is really important and necessary to explain the essential points of the work.

Evaluation criteria

For this delivery, there are different delivery options. Each group must choose what delivery they want to do. **C** is the minimum requirement for this delivery. There is a hierarchy in the options: **C→ B→ A→ A+***

It is not allowed to do:

- B without C
- A without B and C
- A+ without A, B and C

Option C

Apart from all requirements that are written in the **Requirements** section, there are the next mandatory exercises:

1. Document all steps. Structure your code to keep it cleaned using good practices.
2. Collect Coronavirus Data. It is mandatory that in each call, it collects the last updated data.
3. Determine and explain if the data is cleaned. If not, then clean it.
4. Create an API that returns a Json with the logic explained for your group. The flask server must be executed running the **src/api/server.py** file.
5. Get the jsons generated from your annexed group and plot it. First, try to connect to the private ip of your annexed group. If it is not possible because of physical issues, then simply use what they generate copying it. If your annexed group cannot give you the necessary json, then annotate it, use the json of another group. If there are no jsons from other groups, then use your json from your own API.
6. Show different tendencies for each column in your dataset. Show, vertically, the start date and end date of the alarm state in each plot. If there is no alarm state, then show only the start date.
7. Answer the questions:
    a. What position do your countries occupe respect to the number of total infected, total deaths and total recoveries?
    b. What can you conclude about your data study?
    c. Are there outliers or some rare data?

Option B

1. Draw, in different colors and vertically, the moments when the daily death curve increases and decreases.
2. Create with bars, lines, points and pie charts the daily deaths and infected.

Option A

1. Research to save each plot in local files. Save them on different folders for each country.
2. Use distribute modules for each functionality. The jupyter notebooks must not have any loop or function definitions. It only must have the initials imports and the call to necessary functions.
3. Answer the questions:
    a. Can we conclude that the alarm state has had an effect on the improvement of the daily infected rate? explain why (we know it is more complex of what you can explain with your data)
    b. How is the progression going each ten days?

Option A+

There are different A+. The groups can do the ones they want:

1. Use a different github repository adding all group participants with write permissions. Use that repository to manage the delivery code and resources.
2. How can you put your flask server with a public IP? realize that flask starts the server in a private net as default (localhost)
3. How can you put your flask server with a public URL?
4. There are more urls from where to collect Covid-19 data. Collect from one or more different urls and merge the new information by columns with the original. Try to find the country's populations.
5. In order to practice OOP and engineering/architecture concepts in computing, define all the functions inside classes and make the program functional using them. After that, use a program to create the class diagram.
6. Per country, which are the columns that are more related? find the correlation between columns with a *correlation matrix*.
7. Get the total deaths, new deaths, new cases and total cases for your countries using web scraping from this website.