

Real-time Anomaly Detection Project: Scripts Documentation

Bohemian Group

1 Requirements

Read all the Requirements before doing anything on the script file. IT'S MANDATORY!

- A Python version of 3.0 family (it's important, because the 2.0 family works in a different way)(<https://www.python.org/downloads/>);
- A Python IDE (e.g. PyCharm). Not mandatory, but highly recommended (<https://www.jetbrains.com/pycharm/download/#section=windows>, **Community version will be OK**);
- In order for the scripts to work, some additional libraries must be installed. The list of the libraries with their versions are stored in file **requirements.txt**. Put the file in the project directory and launch the following command from PyCharm terminal:

```
pip install -r "path-to-file\requirements.txt"
```

For example:

```
pip install -r C:\Users\user\PycharmProjects\prova\requirements.txt
```

PyCharm terminal looks like this, once you created your project (see below for more details):



Figure 1: PyCharm terminal

Three scripts have been wrote to solve three different problems:

1. **Overspending_Underspending.py**: helps you understand, starting from the database, if, for a given product, the corresponding budget and campaign duration, and a desired data, either if you're in **Underspending** or **Overspending** situation, or if you are in a good situation;
2. **Optional.py**: produces a .xlsx file (a file compatible with Excel or Libre Calculator - for Linux users) with some interesting columns which you can use to understand the behaviour of your campaigns for a given product on different channels. The **campoCalcolato1**, **campoCalcolato2**, **campoCalcolato3**, **campoCalcolato4** correspond to the following ratios:

- $\text{campoCalcolato1} \rightarrow \frac{\sum_i \text{commoncosts}_i}{\sum_i \text{commonimpressions}_i}$
- $\text{campoCalcolato2} \rightarrow \frac{\sum_i \text{commoncosts}_i}{\sum_i \text{commonclicks}_i}$
- $\text{campoCalcolato3} \rightarrow \frac{\sum_i \text{commoncosts}_i}{\sum_i \text{conversions}_i}$
- $\text{campoCalcolato4} \rightarrow \frac{\sum_i \text{commonclicks}_i}{\sum_i \text{commonimpressions}_i}$ (i.e. the CTR, expressed in percentage)

The ratios are calculated for each product, for each source and for each campaign;

3. **Quartiles.py**: helps you understand the behaviour of your advertising project by looking at the CTR ratio and confronting it with the quartiles values of the last data for each product on each campaign;

2 Usage

We'll take into account that you've already installed **PyCharm** on your computer.

First of all, open PyCharm and create a new project. When the window for project creation pops up, click on the little arrow next to **Project Interpreter: New Virtualenv environment**. Some fields will appear, and you can select where to save your project folder. I suggest you to save it on the Desktop, so it will be more easy for you to access to it.

Now that you've created your project, a window just like this will appear:

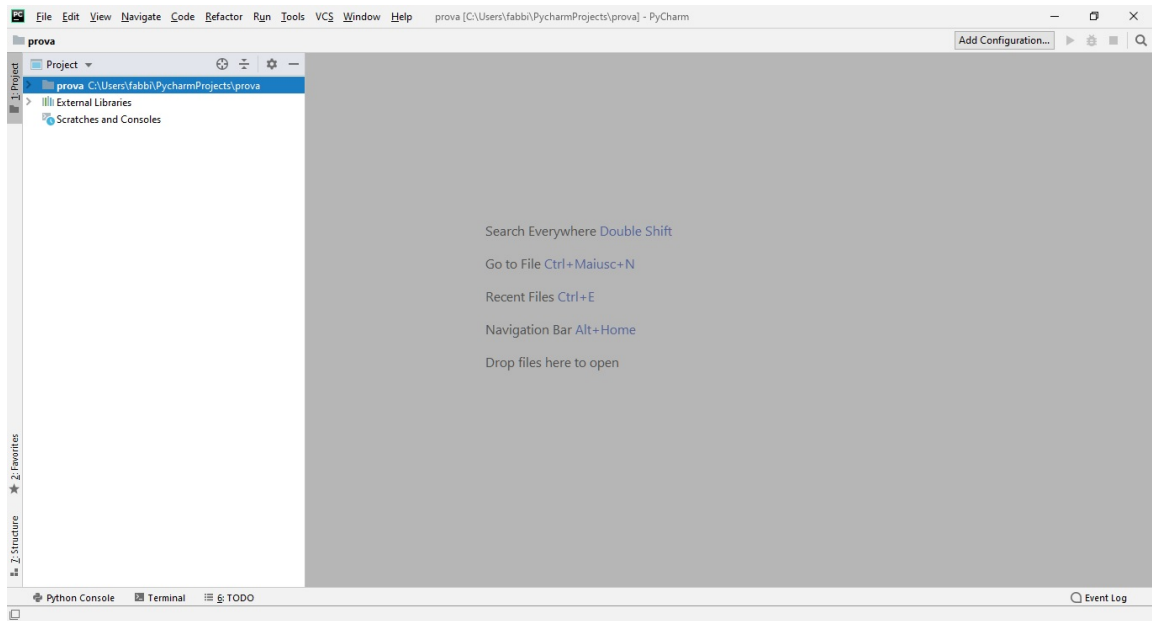


Figure 2: Window for a new PyCharm project

Before doing anything on PyCharm, we need to move inside the project directory all the files we need: **Overspending_Underspending.py**, **Quartiles.py**, **Optional.py**, **functions.py**, **destinatari.txt** and folder **resources**. **functions.py** is a file containing useful functions shared by the scripts, and it's necessary to avoid errors during execution. All the files (including the **requirements.txt**) are contained in the **cleancode** folder you can find on Github. Put the folder inside your project directory, inside the folder named **venv**. Then, from **cleancode** take the scripts files, the **requirements.txt**, the **destinatari.txt** and the **resources** folder and just copy them inside the **venv** folder. In the following image you can see how the file will be organized inside the project directory at the end:

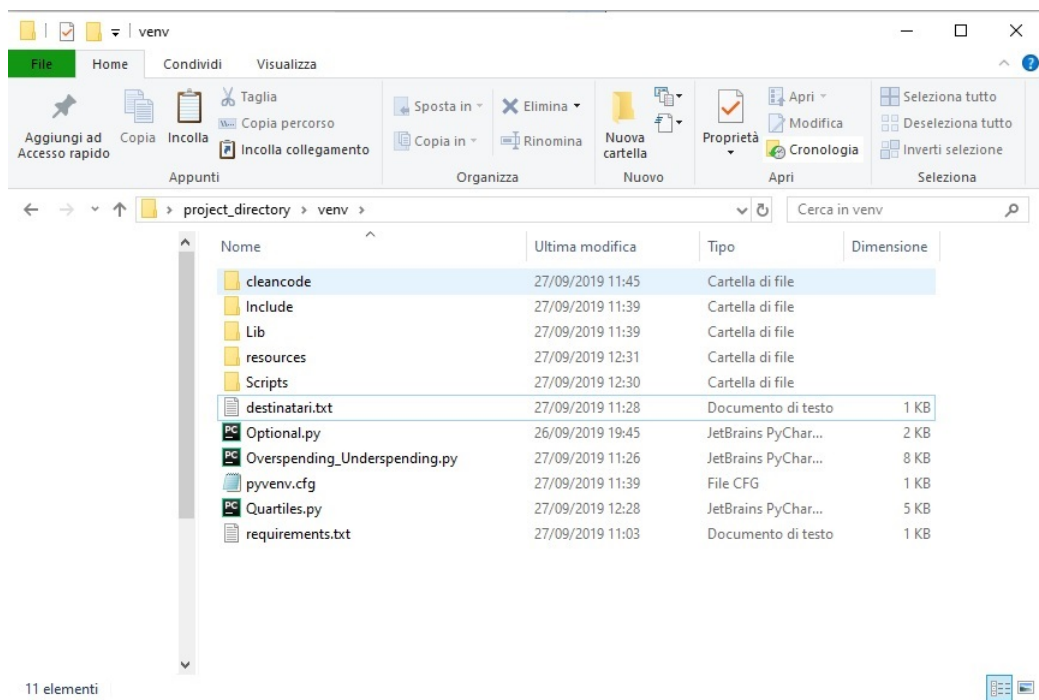


Figure 3: Screenshot showing what you should have inside the project directory

Back to PyCharm, click on the little arrow next to the project name (in my case, I named the project **dummy_project**). Click again on the little arrow next to **venv**. Double click **Overspending_Underspending.py** to open it. Now you can check script's code. To run it, first click to **Add configuration....** This window will appear:

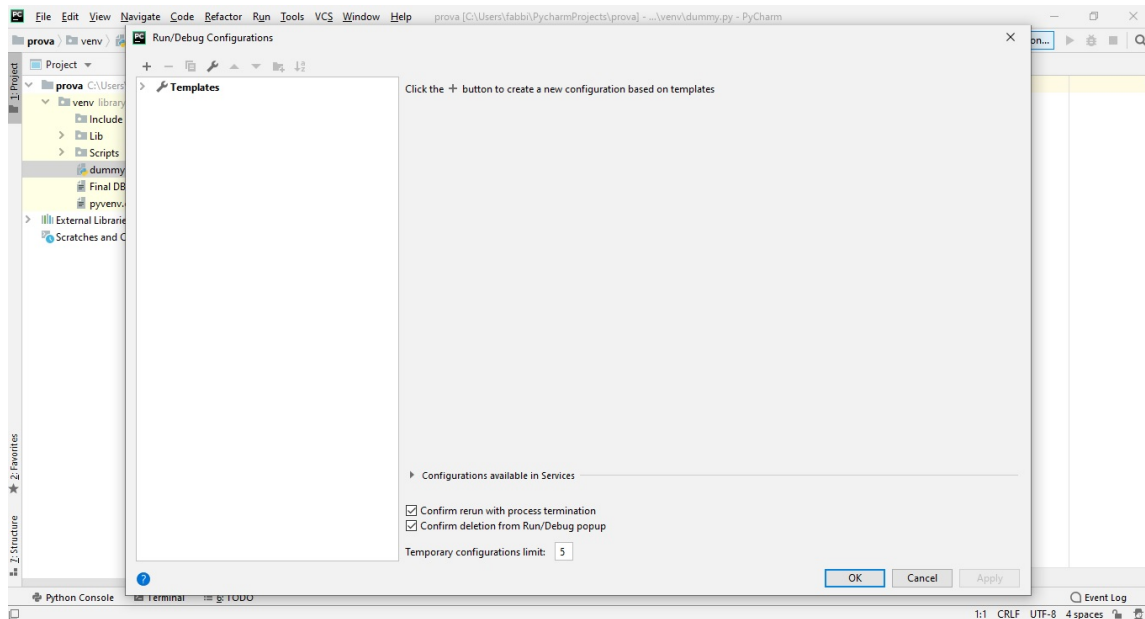


Figure 4: Configuration window

Click on the +, then select **Python**. This window will appear:

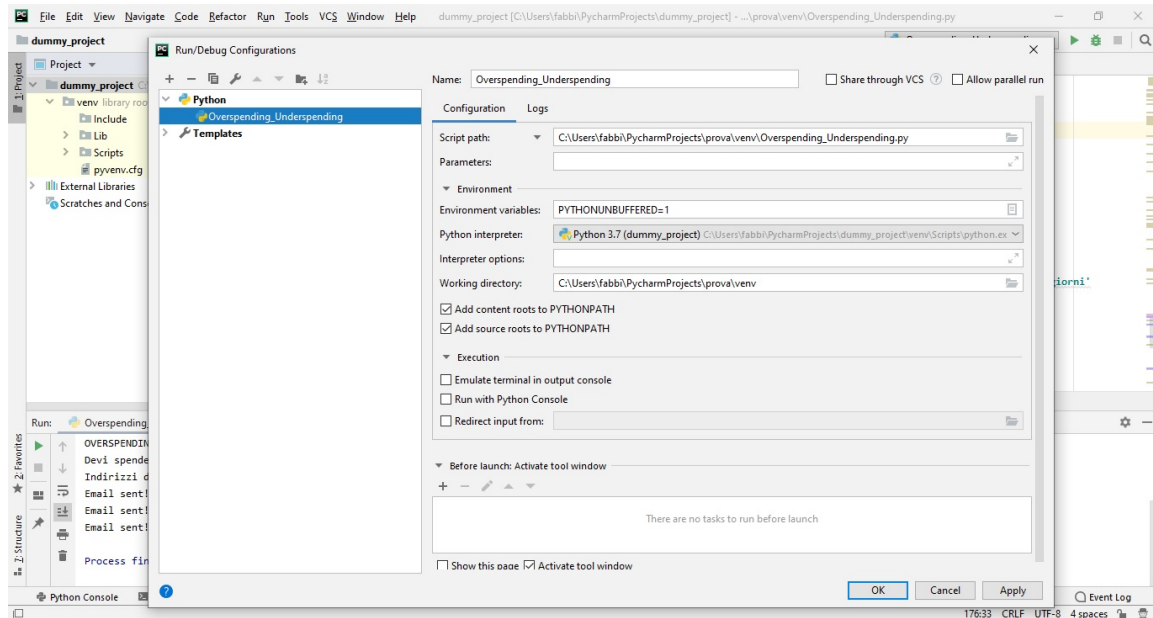


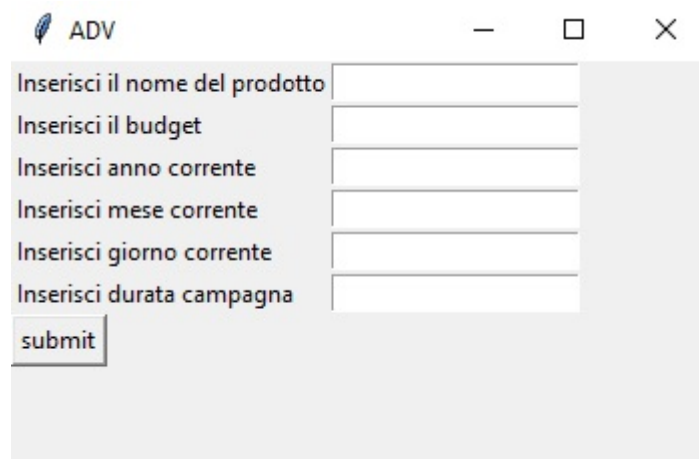
Figure 5: Script configuration settings

If you have nothing written inside Script Path, then use the folder icon inside to check for the path location of your script. Otherwise, just click

OK. Now you can run the script by clicking at the little green arrow next to the configuration button you clicked before, which is now changed into the file name! To run a different script, repeat the procedure by clicking on the configuration button and selecting **Edit Configurations....** Then, when the window pops up, select the path for the different file.

3 Usage for file **Overspending_Underspending.py** file

When the program starts, it loads the whole database from the Google Cloud Platform and manipulate it. Then, this window will appear, asking you to write some data for the analysis:



Label	Input Field
Inserisci il nome del prodotto	<input type="text"/>
Inserisci il budget	<input type="text"/>
Inserisci anno corrente	<input type="text"/>
Inserisci mese corrente	<input type="text"/>
Inserisci giorno corrente	<input type="text"/>
Inserisci durata campagna	<input type="text"/>

Figure 6: Window to enter the desired data

After you entered your data, click on the **submit** button. If your data are correct then the window will close and the program will print your results. If you're in a situation of Overspending or Underspending, the program will sent an email to the clients, which are stored inside a file named **destinatari.txt**, in column. You can change one client email by changing one email inside of file. Also you can add new client email to the file, but keep the column fashion. The email sent will notice to clients the negative results and what they have to do to arrive to the end of the campaign in a good situation.

4 Usage for Optional.py

This second script is used to produce a new DataFrame which contains the four **campoCalcolato** columns which we described before. This DataFrame is then written on a xlsx file, named **Opzionale.xlsx**. The script doesn't take any input values, it just works on the preexisting DataFrame. The only thing to do is to launch the script and it will generate the file automatically in the same directory where the script is saved. Then, use Excel to analyze the results.

5 Usage for Quartiles.py

The last script works in the following way. When you launch it, it first creates a new DataFrame with the quartiles columns, which are **primo_quartile** and **terzo_quartile**. These are used to do the analysis. The new DataFrame is then stored in a new .xlsx file named "**Quartili.xlsx**" which is stored inside the same directory where the script is. Then, the program asks for a product and a campaign, with the following window:

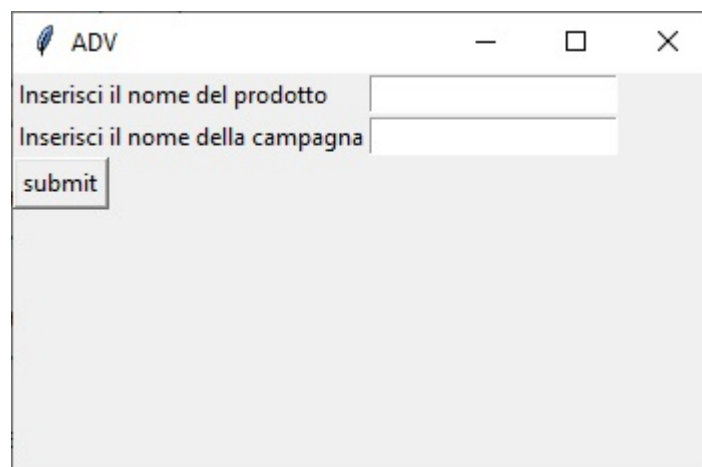


Figure 7: Window to enter the desired data

Once you entered the two desired inputs, click on submit. Then, it will tell you the results of the analysis on the quartiles. If the results are negative, it also sends a mail to the clients reporting them the general result. The clients emails are stored inside the file **destinatari.txt**, in column. You can add manually the clients email, but keep them in a column fashion.