Python Tracker - Short Documentation

AD

November 2020

Abstract

We propose a simple solution to follow the progress of students working on python lab sessions within partially pre-completed Jupyter notebooks.

For the organizer of the lab session, the amount of work is meant to be as limited as possible: adding 4 lines in the first cell, and then one line for each question on which the organizer wants to observe a feedback. This line is typically `send(my_result, question_number)`. For the student, the package is totally transparent.

1 Introduction

Machine learning lectures are nowadays typically combined with labs sessions to have a hands on approach on the problems and discover the computational tools and packages available.

To limit the burden of writing code from scratch, it has become a standard practice to use "Jupyter Notebooks"[1]. Notebooks allow to combine text, pre-filled coding cells, and cells to fill. This is very convenient for teaching purposes, to allow the student to combine e.g., mathematical description of the problems and code. Moreover, it allows students to only focus on the most crucial parts of the code.

Following the progress of students during lab sessions is typically difficult, even when students are in the same room as the teacher, and possibly even more with remote learning. We looked for solutions to improve interactivity, follow students progress,

A solution is to automatically correct the code given by the students by running simple tests on a few random inputs and checking if the output is as predicted (see e.g., nbgrader). However, we did not find a simple solution to rapidly get an overview of students progress with very little work for the organizer.

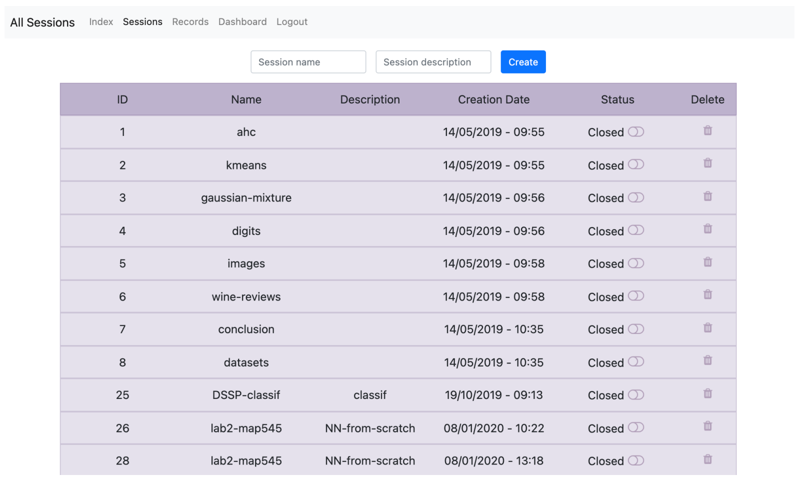
Here, we use the `requests` package to directly send to a server results obtained by the students. We call each of this result a "record". Results stored on the server can then be visualized through a dashboard (which allows to see how many students have performed each question) and the list of records, that can easily be filtered by question or student.

2 The interface

The online interface requires to create an account on https://courdier.pythonanywhere.com/. Once the account is created, 4 Tabs can be accessed.

1. **Index:** this contains documentation.
2. **Sessions:** this allows to create a session, and open or close the session (to prevent any future submission to be added after the session is finished). See Figure 1.
3. **Records:** allows to visualize all the records sent by users. The list can be filtered by session, question number, or user name. See Figure 3, 4, and 5.
4. **Dashboard**: Allows to visualize the global progress of the students. See Figure 2.

[1]https://jupyter.org/



ID	Name	Description	Creation Date	Status	Delete
1	ahc		14/05/2019 - 09:55	Closed	
2	kmeans		14/05/2019 - 09:55	Closed	
3	gaussian-mixture		14/05/2019 - 09:56	Closed	
4	digits		14/05/2019 - 09:56	Closed	
5	images		14/05/2019 - 09:58	Closed	
6	wine-reviews		14/05/2019 - 09:58	Closed	
7	conclusion		14/05/2019 - 10:35	Closed	
8	datasets		14/05/2019 - 10:35	Closed	
25	DSSP-classif	classif	19/10/2019 - 09:13	Closed	
26	lab2-map545	NN-from-scratch	08/01/2020 - 10:22	Closed	
28	lab2-map545	NN-from-scratch	08/01/2020 - 13:18	Closed	

Figure 1: Sessions Tab: create new sessions, open, close or delete existing ones.

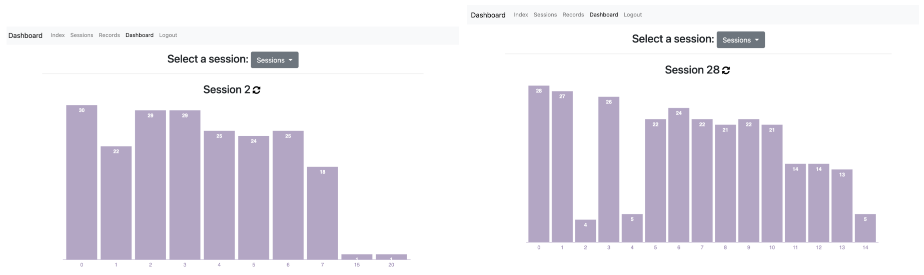


Figure 2: Dashboard visualization: track the number of students that have completed each question.

2.1 Type of data that can be sent into a record

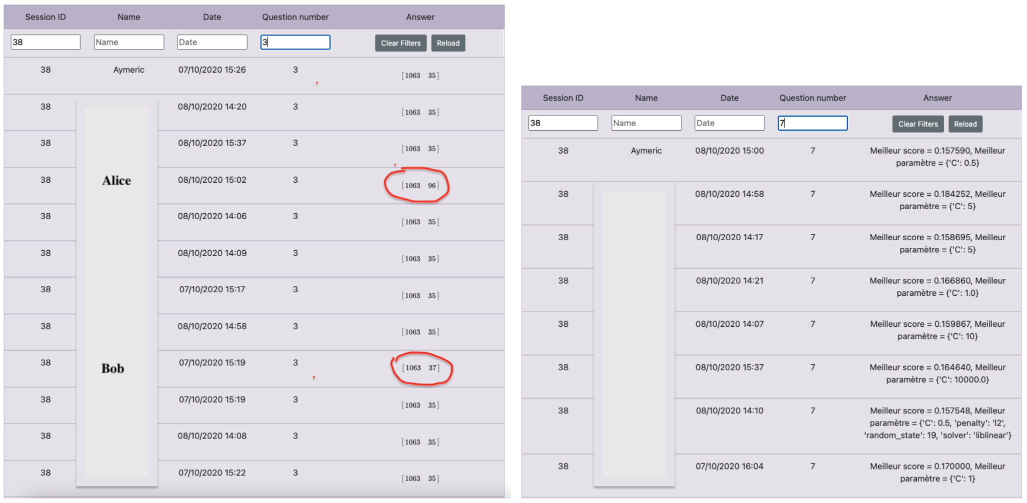
The following data can be sent:

- Strings (e.g., deterministic “lab started”, or to be completed “the number of parameters is ...”)
- np.arrays, integers, floats
- plots
- code of a function
- other types can easily be added by patching the send function.

2.2 Possible use cases

Remark that designing/choosing efficient quantities to observe requires a bit of exploration/exploitation.

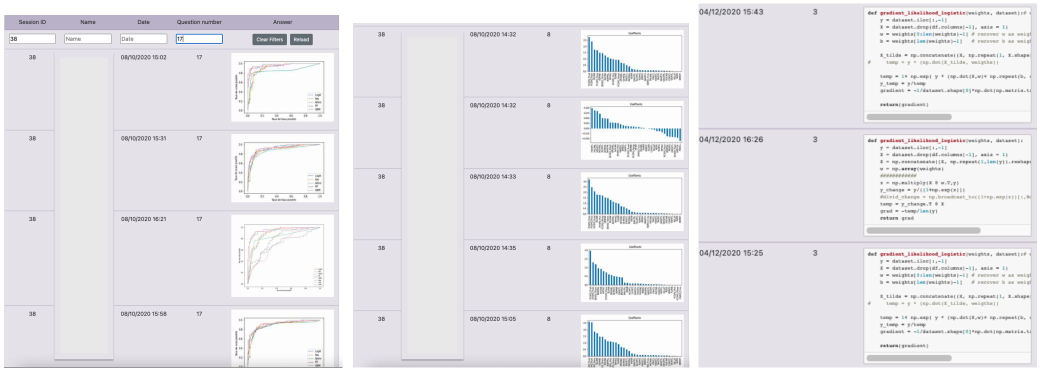
- **Incentive for the student to connect.** Tracking the number of questions solved by each student is a good incentive for the student to dive in the lab session.
- **Regulating the pace:** I have often be surprised by how long it takes for the students to start the lab. Having a permanent access to how many students have say started, reached and succeeded in a specific question is very helpful. The Dashboard is used in that case.



Session ID	Name	Date	Question number	Answer
38	Name	Date	3	Clear Filters Reload
38	Aymeric	07/10/2020 15:26	3	[1063 35]
38		08/10/2020 14:20	3	[1063 35]
38		08/10/2020 15:37	3	[1063 35]
38	Alice	08/10/2020 15:02	3	[1063 96]
38		08/10/2020 14:06	3	[1063 35]
38		08/10/2020 14:09	3	[1063 35]
38		07/10/2020 15:17	3	[1063 35]
38		08/10/2020 14:58	3	[1063 35]
38	Bob	07/10/2020 15:19	3	[1063 37]
38		07/10/2020 15:19	3	[1063 35]
38		08/10/2020 14:08	3	[1063 35]
38		07/10/2020 15:22	3	[1063 35]

Session ID	Name	Date	Question number	Answer
38	Name	Date	7	Clear Filters Reload
38	Aymeric	08/10/2020 15:00	7	Meilleur score = 0.157590, Meilleur paramètre = {'C': 0.5}
38		08/10/2020 14:58	7	Meilleur score = 0.184252, Meilleur paramètre = {'C': 5}
38		08/10/2020 14:17	7	Meilleur score = 0.158695, Meilleur paramètre = {'C': 5}
38		08/10/2020 14:21	7	Meilleur score = 0.166860, Meilleur paramètre = {'C': 1.0}
38		08/10/2020 14:07	7	Meilleur score = 0.159867, Meilleur paramètre = {'C': 10}
38		08/10/2020 15:37	7	Meilleur score = 0.164640, Meilleur paramètre = {'C': 10000.0}
38		08/10/2020 14:10	7	Meilleur score = 0.157548, Meilleur paramètre = {'C': 0.5, 'penalty': 'l2', 'random_state': 19, 'solver': 'liblinear'}
38		07/10/2020 16:04	7	Meilleur score = 0.170000, Meilleur paramètre = {'C': 1}

Figure 3: Examples of records. For example, on the left, the vector was supposed to contain the number of observation and features. Observing the results allows to give feedback to Alice and Bob, who do not have the correct record. On the right, the results of a model training are given. Asking the students to compare the results allows to comment on the variability of different random seeds and the robustness of the approach.



Session ID	Name	Date	Question number	Answer
38	Name	Date	17	Clear Filters Reload
38		08/10/2020 15:02	17	
38		08/10/2020 15:31	17	
38		08/10/2020 16:21	17	
38		08/10/2020 15:58	17	

38		08/10/2020 14:32	8	
38		08/10/2020 14:32	8	
38		08/10/2020 14:33	8	
38		08/10/2020 14:35	8	
38		08/10/2020 15:05	8	

04/12/2020 15:43 3

```
def gradient_likelihood_logistic(weights, dataset):# v
    y = dataset.iloc[:,-1]
    X = dataset.drop(df.columns[-1], axis = 1)
    w = weights[0:len(weights)-1] # recover w as weig
    b = weights[len(weights)-1]   # recover b as weig

    X_tilde = np.concatenate((X, np.repeat(1, X.shape
#     temp = y * (np.dot(X_tilde, weigths))

    temp = 1+ np.exp( y * (np.dot(X,w)+ np.repeat(b, 
    y_temp = y/temp
    gradient = -1/dataset.shape[0]*np.dot(np.matrix.t

    return(gradient)
```

04/12/2020 16:26 3

```
def gradient_likelihood_logistic(weights, dataset):
    y = dataset.iloc[:,-1]
    X = dataset.drop(df.columns[-1], axis = 1)
    X = np.concatenate((X, np.repeat(1,len(y)).reshape
    w = np.array(weights)
    ############
    z = np.multiply(X @ w.T,y)
    y_change = y/((1+np.exp(z)))
    #divid_change = np.broadcast_to((1+np.exp(z))[:,No
    temp = y_change.T @ X
    grad = -temp/len(y)
    return grad
```

04/12/2020 15:25 3

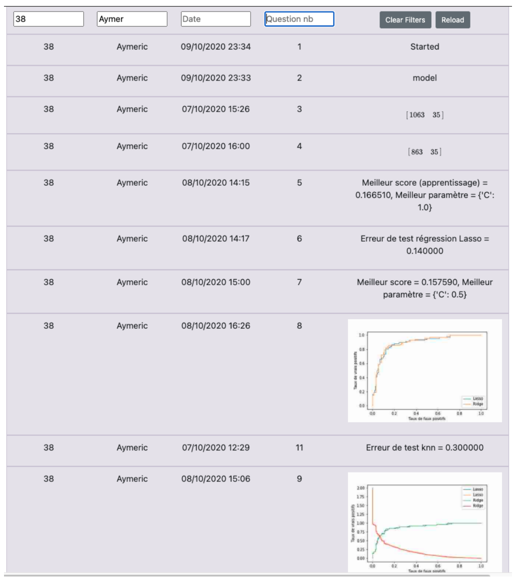
```
def gradient_likelihood_logistic(weights, dataset):# v
    y = dataset.iloc[:,-1]
    X = dataset.drop(df.columns[-1], axis = 1)
    w = weights[0:len(weights)-1] # recover w as weig
    b = weights[len(weights)-1]   # recover b as weig

    X_tilde = np.concatenate((X, np.repeat(1, X.shape
#     temp = y * (np.dot(X_tilde, weigths))

    temp = 1+ np.exp( y * (np.dot(X,w)+ np.repeat(b, 
    y_temp = y/temp
    gradient = -1/dataset.shape[0]*np.dot(np.matrix.t

    return(gradient)
```

Figure 4: Example of records: plots or the code of a function can be sent. It allows to improve and compare visualizations, observe variability, correct mistakes (e.g., an absolute value missing in the 2nd answer in the middle)



38	Aymer	Date	Question nb	
38	Aymeric	09/10/2020 23:34	1	Started
38	Aymeric	09/10/2020 23:33	2	model
38	Aymeric	07/10/2020 15:26	3	[1063 35]
38	Aymeric	07/10/2020 16:00	4	[863 35]
38	Aymeric	08/10/2020 14:15	5	Meilleur score (apprentissage) = 0.166510, Meilleur paramètre = {'C': 1.0}
38	Aymeric	08/10/2020 14:17	6	Erreur de test régression Lasso = 0.140000
38	Aymeric	08/10/2020 15:00	7	Meilleur score = 0.157590, Meilleur paramètre = {'C': 0.5}
38	Aymeric	08/10/2020 16:26	8	
38	Aymeric	07/10/2020 12:29	11	Erreur de test knn = 0.300000
38	Aymeric	08/10/2020 15:06	9	

Figure 5: Track one student's progress.

- **Personalized/shared feedback:**. Checking the result of a student of at a question allows to directly correct the simplest mistakes. If several students have the same mistake, you can talk to them as a group. The records are then used.
- **Interactivity**: seeing different results for different persons helps to illustrate variability or promote the best solutions given by the students.

2.3 Tips and examples

Tip 1: it can be better to alternate records

- that aim **at being challenging** (e.g., the code of a function)
- that aim at checking progress, (e.g., a simple string to fill, say *"the number of variables is ??"*, where the student just has to fill the ??)

This avoids underestimation of the number of students following the lab of having reached a question but not finalized a question.

Tip 2: it can be better to alternate records

- that allow the teacher to check if the answer is correct immediately (the value of the function at a chosen point)
- that allow to find the mistake (e.g., the code of the function)

Remarks:

- If a student makes several submissions for the same question, only the *last* record is kept.

3 Preparing the lab: what needs to be done before.

To use the website, one needs to follow these steps:

1. Sign in on the website (Create an account at the first access).

2. Before a lab, create a *session* on the website (in tab: Session), and check its *remember the session name.*

3. In the notebook that will be shared to the students,

   (a) add the following lines in the first cell, and provide the session_name and session_owner (your id) in the following code:
   ```
   import requests
   exec(requests.get("https://courdier.pythonanywhere.com/get-send-code").content)
   npt_config = {
   'session_name': 'your-session-name-here',
   'session_owner': 'your-username-here',
   }
   npt_config['sender_name'] = input("Name:")
   ```
   (b) In the code, for each question on which you want to have feedback, add the `send(result, question_number)` function, with `result` the object that will be sent, and `question_number` the number of the question. To send the current plot, just use `plt` for result.

4 More details

4.1 How it works

There is no black magic! The crux of the problem is to send the result from the Notebook tracker to the server. This is done through the `requests` package, and the function `send`. The code of the `send` fucntion is given here.

Running it on one's own server Instructions to store the records and run the website on your own server are given on the GitHub repository here.

However this is not recommended for the moment as the software is often updated.

4.2 Disclaimers and Known minor bugs

This is not a professionally developed software. The authors decline any responsibility in case of bugs. :)

The tool is **not necessarily meant** to be robust to attacks. It is well suited for a group of students that understand the benefits of such a tool.

Students can e.g. easily block the upload of the record (by removing the function `send`), or send unrelated content, or try to attack the server by sending high amounts of data.

We list here a few things the tool is not robust too for the moment:

- Overflowing the server
- Sending results to another open session (normally solved, unless they change session owner, could be protected by a key but makes things more complex)
- 2 students with the exact same name may overwrite (filter by ip)
- Connection through VPN may create a but.
- The question number are voluntarily left to be hand filled by the teacher. Having an incremental system (adding one after each `send`) would fail if a student launches several time the same cell. We are working on a package to automatically re-number the “send” once the lab is ready.

Similarly, the students needs to “trust” the teacher, which runs a code found on internet on their own machine. However, this is a general constraint for any notebook, in which students are asked to run code that has been previously developed by the teachers or community.

In other words, a malicious session organizer could always easily harm the students that blindly run code on their machine, the notebook tracker does not really change this aspect.

5 Licensing and citation

The code is provided on : `https://github.com/theevann/notebook-progress-tracker`.

Please provide us feedback if you use the tool.

The project was created by Evann Courdier and Aymeric Dieuleveut. The code was mostly developed by Evann Courdier.