

**Project Ile de France challenges**

**Preparing a starting kit and a challenge bundle :**

**Application to time series data with SNCF example**

**Isabelle Guyon**

[**iguyon@lri.fr**](mailto:iguyon@lri.fr)

**Version 1, March 31, 2019**

**Getting started:**

1. Install on your computer **Anaconda Python 3.6** or later. If you already have another version of Python or Anaconda on your computer, use virtual environments:

conda update conda

conda create -n python3 python=3.6 anaconda

Then you will be able to use the 3.6 version by activating the virtual environment:

conda activate python3

To get out, use:

conda deactivate

This installation should include python 3.6, the spyder IDE and jupyter-notebook (verify).

1. Another possibility will be to use the docker codalab/codalab-legacy:py3. More on dockers at the end of this document. If you are going to use OTHER libraries than those provided in Anaconda, definitely use dockers!
2. [Clone](https://help.github.com/articles/cloning-a-repository/) the starting kit template (this directory): If you are new to Github, take a tutorial. Using Git and Github is essential to your project. This example allows you to build a sample competition.
3. **Starting kit structure:**

A starting kit should imperatively have the following structure, do NOT change it:

* **README.md:** Short description of your task and credits.
* **README.ipynb:** Sample code reading data, calling sample code, generating sample submissions.
* **sample\_data/:** A SMALL data sub-sample (could be a subset of the training data, but should NOT include any validation or test examples on which the competitors will be tested).
* **sample\_code\_submission/:** Self-contained, ready-to-submit example code submission, including the predictive model class model.py and a metadata file (with just a comment). This is a template of what the participants are supposed to produce.
* **ingestion\_program/:** Program and libraries needed to run code submissions on the challenge platform and generate prediction results using the input\_data (the input time series) and model.py. The participants cannot make any change to this program.
* **scoring\_program/:** Program and libraries needed to score the prediction results on the challenge platform using the truth values of the time series to be predicted, also called reference\_data. A library of scoring functions to choose from is provided in libscore.py. All you need to do is to put the name of the chosen function in metric.txt. If needed, you can supply the code of your own metric in my\_metric.py. You can also use a scikit-learn metrics.
* **html\_files:** Documentation files in HTML format.

1. **Data preprocessing and formatting:**

* **Data format: The data** formatcan be freely chosen by the organizers. Human legible formats (text rather than binary formats) are preferred, for ease of diagnosis
* **Missing data:** Represent the missing values with the symbol **NaN.**
* **Categorical variables:** Preferably represent all variables with a **NUMERIC code** or supply a data reader that transform them on the fly to numeric values.
* **Size dataset:** Do not consider too small datasets. Avoid datasets with only a few hundred samples. Ideally do not consider datasets with less than 10,000 samples. Evaluate error bars using baseline submissions and increase the size of the dataset as needed.
* **Precision:** We want enough samples, but the overall compressed volume of data should not exceed **1 GB**. Quantizing values to enough precision (but not too much) is recommended, as part of preprocessing. Usually quantizing between 0 and 999 is enough.
* **Data split:** It is best if the organizers supply data already split, to avoid any confusion. Provided data should be partitioned in two directories, as follows:
  + training/ = (optional) training “historical” data, which can be used for off-line training of models.
  + evaluation/ = additional data visible only to the code of the participants, which will be used to evaluate their code (on-line adaptation of the models will be permitted).
* **Documentation:** Document the data with a README.md file and add a license file, as needed.

**Data leakage:** This refers to features that inform inadvertently the future about the past. Check <https://www.cs.umb.edu/~ding/history/470_670_fall_2011/papers/cs670_Tran_PreferredPaper_LeakingInDataMining.pdf> for horror stories.

**Sample data and challenge data**: **sample data** a small data sample, which may be a subset of the training data, used by the organizers to debug the starting kit. It should have the same structure as the final **challenge data** (split into training/ and evaluation/ directories), but just be smaller.

1. **Sample submissions:**

You must prepare a sample code submission put in directory **sample\_code\_submission/**. This directory should contain a Python class model.py. This should be a basic but functional baseline method solving your problem. The directory sample\_code\_submission should also contain a metadata file with just a comment. The directory may or may not also contain a pickle with a pre-trained model.

1. **Ingestion program:**

A sample ingestion program is provided. You need to modify it if you change the data format of the example provided. The ingestion program receives the code submission of the participants on the server and runs it. However, you can also run it locally to produce a result submission: python ingestion\_program/ingestion.py **challenge\_data** sample\_result\_submission ingestion\_program sample\_code\_submission

where challenge\_data/ should contain your real challenge data. For debug purposes, you can use sample\_data/ instead of challenge\_data/. **Remove the pickle in sample\_code\_submission, if any** (may cause an error if you switch between Python versions).

Most likely, what you might want to change is the class DataManager. This is a class that reads the data and transforms it into a little object oriented database that is used by the ingestion program to supply the data as desired. The class DataManager is also used as a data structure for output predictions.

We have implemented for you an example of data manager, with the following interface:

class DataManager:

def \_\_init\_\_(self, datatype="input", …) # “input”/“output”

def loadData (input\_dir, max\_samples=float('inf'), verbose="True") -> success

def getHistoricalData () -> X, t

def getFutureOutcome () -> Y, t

def appendData(X, t)

which allows you to easily read and iterate through the data.

1. **Scoring program:**

A scoring program is provided. You need to modify it if you change the data format of the example provided. If you do not change the format, you may still want to change the scoring metric. Place your code in my\_metric.py. Instead of writing your own code, you may also select one of the metrics for libscore.py and specify it in metric.py:

|  |  |  |  |
| --- | --- | --- | --- |
| **Binary classification** | **Multi-class classification** | **Multi-label classification** | **Regression** |
| bac\_binary | bac\_multiclass | bac\_multilabel | abs\_regression |
| auc\_binary |  | auc\_multilabel | r2\_regression |
| pac\_binary | pac\_multiclass | pac\_multilabel |  |
| f1\_binary |  | f1\_multilabel |  |

Test your metric with:

python scoring\_program/score.py **sample\_data** sample\_result\_submission scoring\_output

where sample\_data/ can be replaced by challenge\_data/ if you want to test on the whole dataset. scoring\_output is a directory that will be created.

1. **Jupyter-notebook:**

Provide a simple Jupyter notebook to help the participants getting started. Keep the basic structure of the template provided but replace the yellow place holders with relevant graphics and explanations:  
jupyter-notebook README.ipynb.

1. **From the starting kit to the challenge bundle:**

The challenge data and starting kit are the basis for the Codalab challenge. Here is your **check list:**

* **DATA:** The challenge **data** should have this structure:

**challenge\_data/ training/**

**evaluation/**

Put it in the directory **FILES/** at the same level as **starting\_kit/**

**In the starting kit, use as sample\_data a small sample taken out of the training data.**

* **STARTING KIT:** Your **starting kit** directoryshould have this structure:

logo.jpg

README.md.

README.ipynb

sample\_data/ **training/**

**evaluation/**

sample\_code\_submission/

ingestion\_program/

scoring\_program/

html\_files/

* **LOGO**: Prepare a **logo** (a small square jpg file), call it **logo.jpg.**
* **TASK:** The contents of **ingestion\_program/** **change as desired, particularly data manager**.
* **METRIC:** Declare your metric in **scoring\_program/metric.py**, eventually using a metric of scikit-learn one from **libscores.py**, OR the code of your own code**.** The rest of **scoring\_program/** can probably remain **unchanged**.
* **SAMPLE SUBMISSION:** You should have prepared a sample code submission in: **sample\_code\_submission/**.
* **HTML FILES:** You should also prepare documentation files in HTML format in **html\_files/**: You can edit them later so do not worry too much about them, you can keep the template.
* **README FILES:** README.ipynb (update with your own code) and README.md (update with your own information).

Compiling your challenge bundle

In the directory **starting\_kit/utilities/,** you will find two files:

competition.yaml

make\_bundle.py

This will allow you to compile your challenge bundle at the prompt.

Usage (1) with sample data only:

**python make\_bundle.py**

Usage (2) with big data:

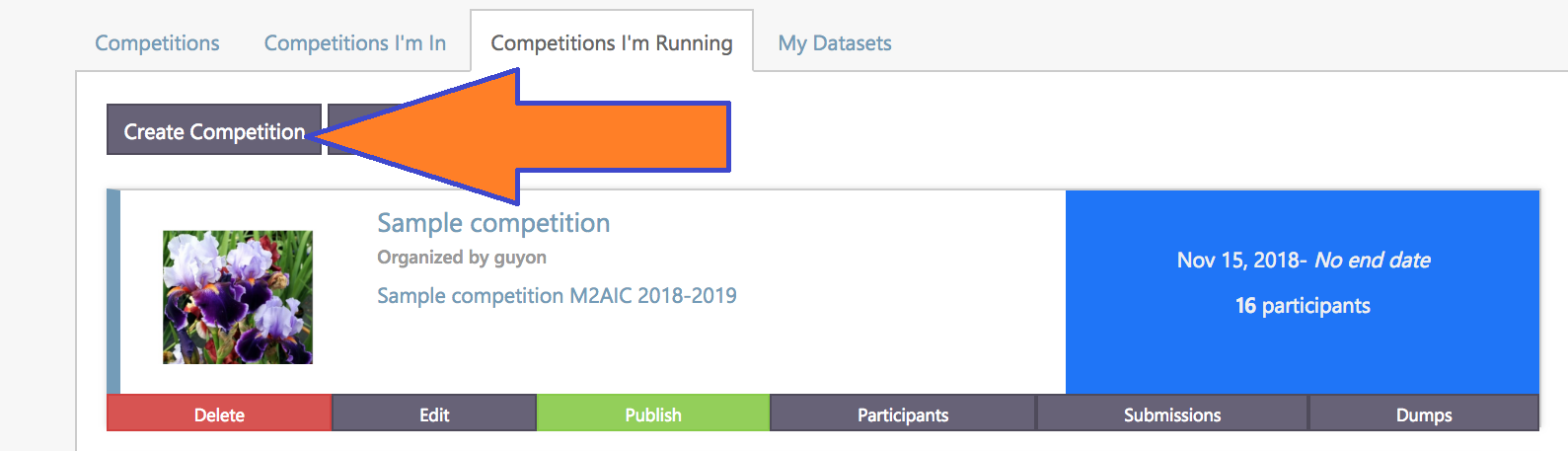
**python make\_bundle.py ../ ../../FILES/challenge\_data**

This creates a zip file **DataName\_bundle\_date.zip** in the same directory.

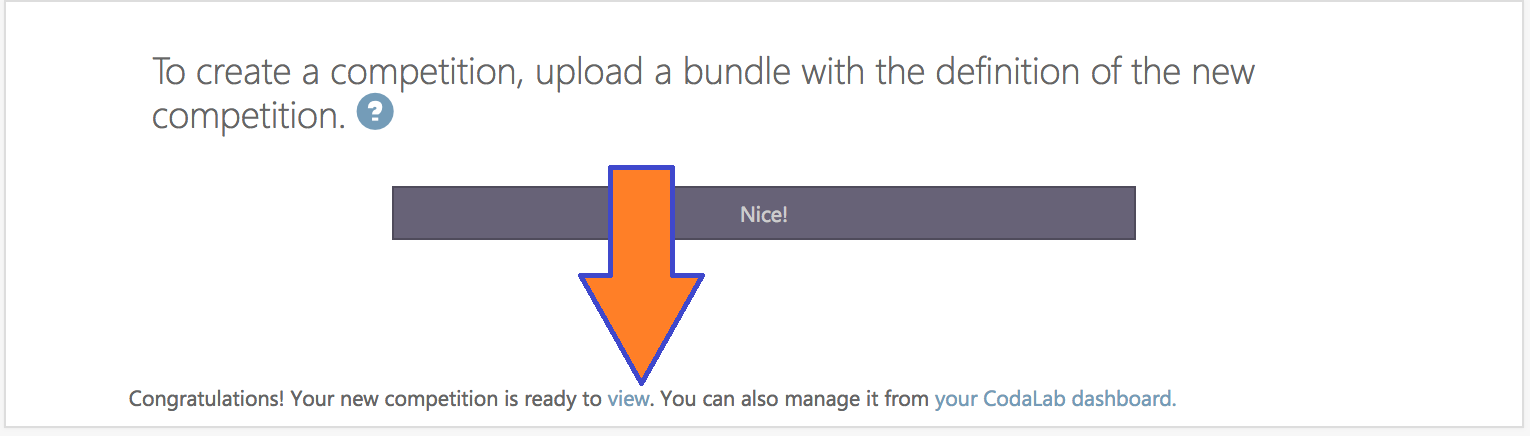
Upload the challenge bundle to Codalab

Upload **DataName\_bundle\_date.zip** to <https://codalab.lri.fr/>.

To do this, use create a Codalab account, login and the go to “My Competititions” > “Competitions I’m Running” and click “Create Competition”.

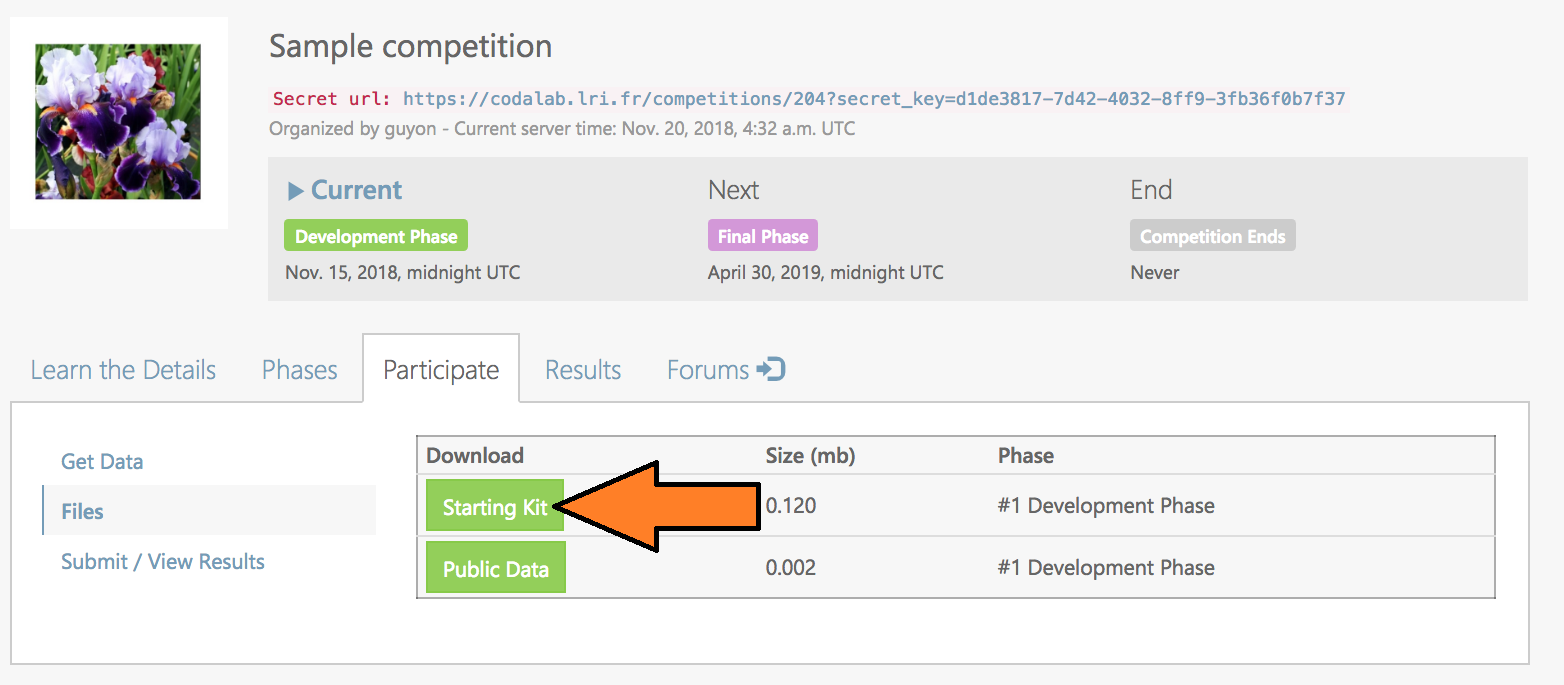


Once you upload your bundle, if all goes well, you can click on a microscopic link to view your competition:



If you upload the sample competition, you get something like: <https://competitions.codalab.org/competitions/22438?secret_key=b85045b7-17af-4a9c-a523-538a43882582>

Do not upload a submission from your computer yet. First test the starting kit that is on the challenge website. Download it from the “Files” tab under “Participate”:

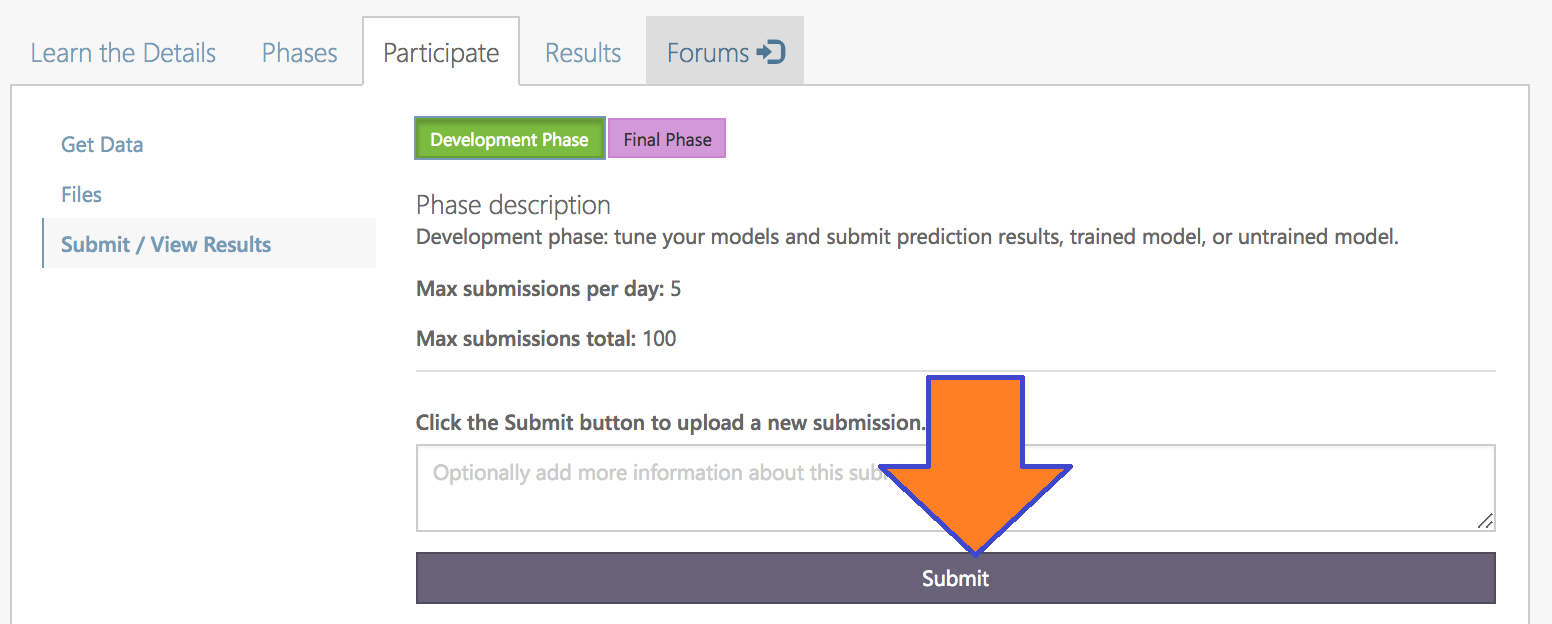


In the same place, you can also download your “big” data (the public part only, not the validation and test labels).

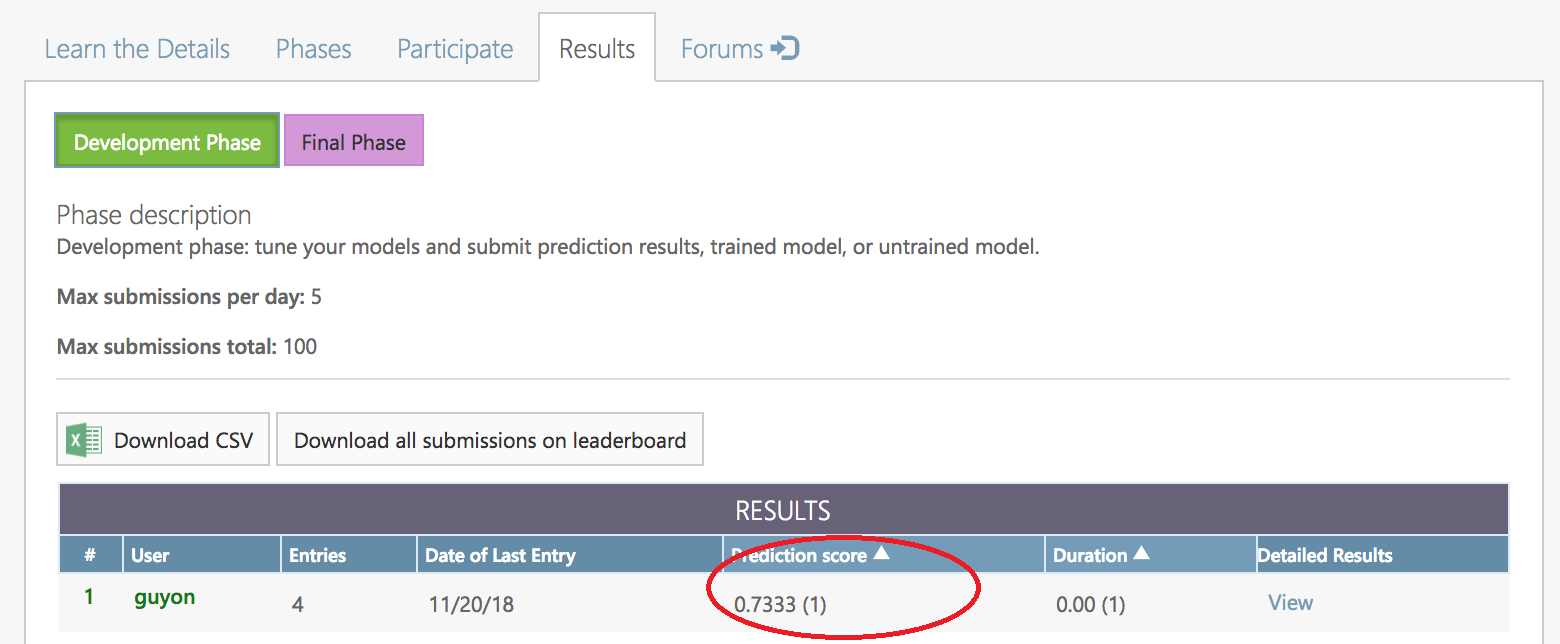
Then in your starting kit, locate your sample submissions:

sample\_code\_submission.zip

Upload it to Codalab to check that it works:



Then look at the results on the leaderboard:



To do this you will need to go to **Participate>View/Submit results.**

**WARNING: you will get errors if the correct docker is not specified. See page 11.**

**Large datasets or evaluation data uploaded later**

If you have a very large dataset or if you want to upload data for the final phase at a later stage, you can compile your bundle with sample data only and then replace the place holder dataset by the real one.

1. Separate the training and evaluation data and create 3 archives:

* **public\_training\_data.zip**, containing:

training/

training0.csv

training1.csv

training2.csv

* **evaluation1\_data.zip**, containing:

training/

training0.csv

training1.csv

training2.csv

evaluation/

evaluation1.csv

evaluation2csv

* **evaluation2\_data.zip**, containing:

training/

training0.csv

training1.csv

training2.csv

evaluation1.csv

evaluation2csv

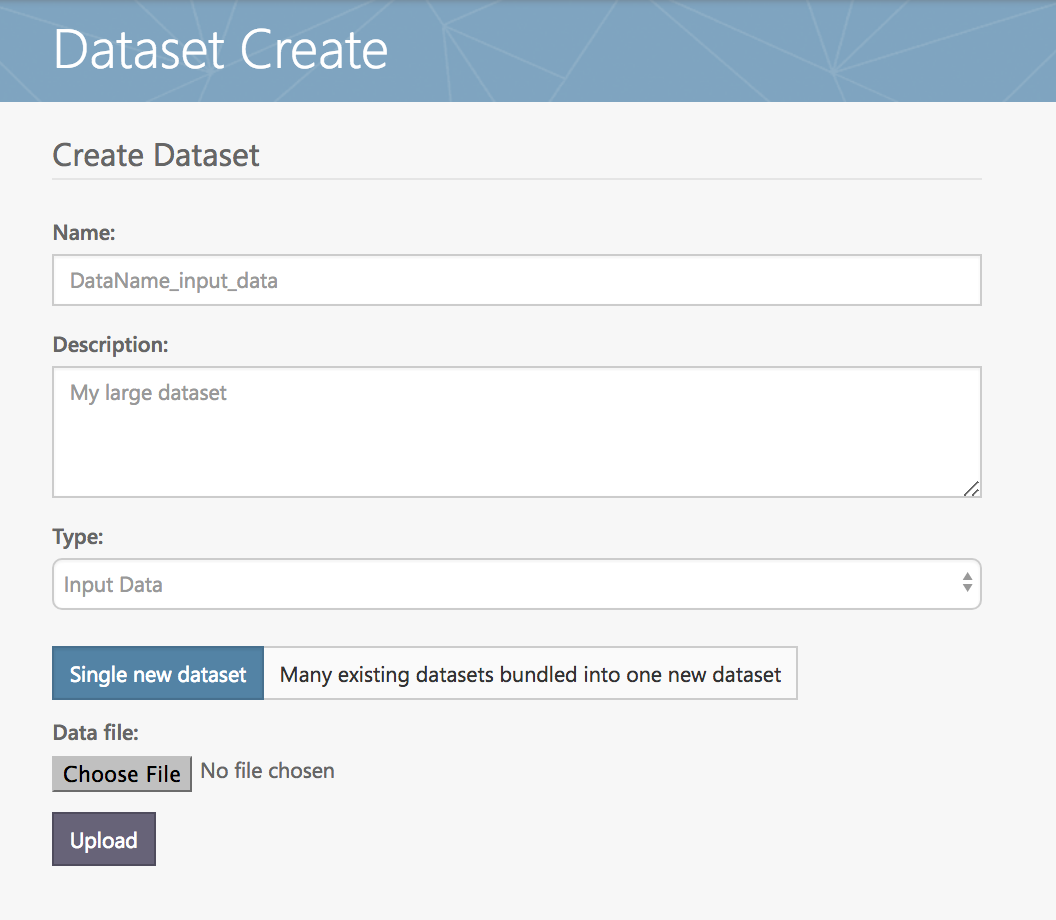
evaluation/

evaluation3csv

evaluation4csv

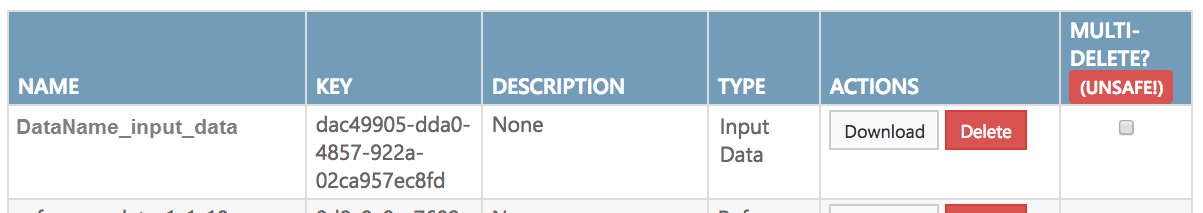
In each zip the contents of the directories training and evaluation can be different. We show here that in the final evaluation (evaluation 2) you may use as training data some files used for evaluation In the previous phase.

1. go to **My Competitions>My Datasets>Create Dataset.** Fill out the form:



Upload the file **public\_training\_data.zip**. After you upload, you should see your dataset in the data table. The KEY can be used to refer to it from the YAML file, as public\_data.

public\_data: dac49905-dda0-4857-922a-02ca957ec8fd

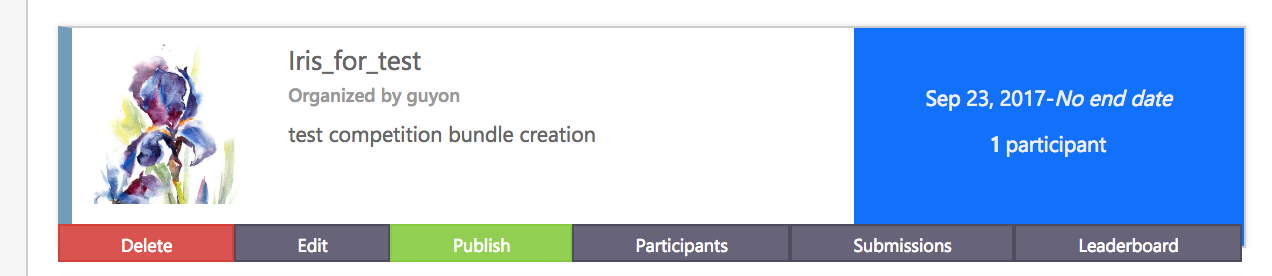


Do the same thing for **reference\_evaluation1\_data.zip** and **reference\_evaluation2\_data.zip**. Upload them each as inout data and as reference data. You can then use the key obtained in the YAML file:

input\_data: 0d8c9c0e-7609-4314-bf1c-3154cdee3462

reference\_data: 0d8c9c0e-7609-4314-bf1c-3154cdee3462

You can also use the editor. In Competitions I’m running, find your competition and click “Edit”:

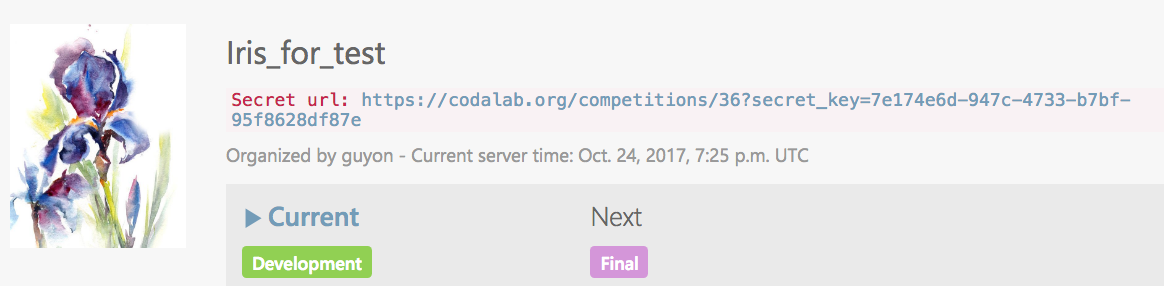


Find the menus for Input Data, Reference Data, and Public Data. Select the right dataset, as don’t forget to **SAVE YOUR CHANGES.**



**Editing your challenge on Codalab**

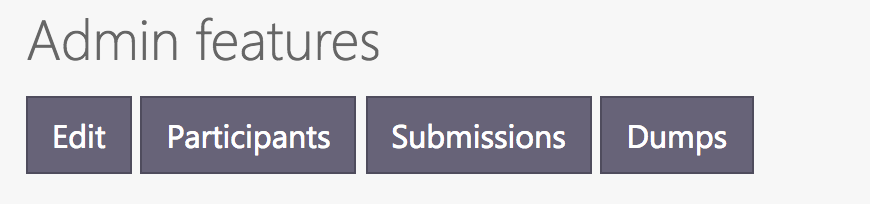
Once your challenge is uploaded to Codalab, your whole team can contribute to improving it. Before you make your challenge public (by clicking “Publish”) you can share the secret URL with others.



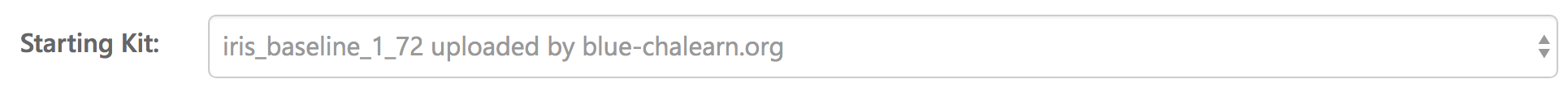
But this will not allow them to edit the challenge. To give them this privilege, go to the editor and add their Codalab ID to the list of admins:



The Codalab editor can be used to edit your challenge. However, try to minimize to avoid winding up with a broken site. Save your intermediate challenge bundles by using the **Dump** button.



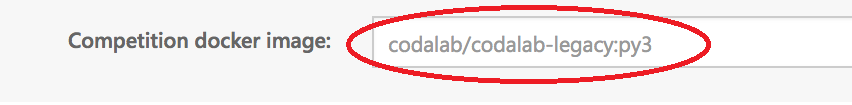
Editing your challenge allows you to update your starting kit. You can upload it in my datasets, then point to it from the editor:



If your score is an “accuracy” (better is higher), you want the leaderboard sorted in descending order, but if it is an “error” (better is lower), you want it in ascending order. You can change that:



We want to run the code into Python 3 dockers. Make sure the correct docker is indicated:



**Using dockers**

Codalab allows you to specify a docker in which you code will be running. Thus it is convenient to test your code in this docker ahead of time.

When you submit code to the Codalab platform, your code is executed inside a docker container. This environment can be exactly reproduced on your local machine by downloading the corresponding docker image. The codalab/codalab-legacy:py3 docker, which can be pulled from <https://hub.docker.com/r/codalab/codalab-legacy/tags/> contains a large number of pre-loaded programs, including Python 3 and many libraries including scikit-learn. See <https://github.com/codalab/codalab-dockers/tree/master/legacy-py3> for details.

To run the starting kit inside the Codalab docker from a terminal:

**Step 1) Create a temporary folder** to put everything you will need (data and starting kit):

mkdir ~/aux

cp starting\_kit.zip ~/aux

Copy any other data/code you want to that folder, as we will instruct you to map such local folder to be visible when you are within the docker. If you don’t do such mapping procedure, you will not be able to see what you need from within the docker.

**Step 2) Run the docker** (and map such folder, described above, to /home/aux, inside the docker):

docker run -it -v ~/aux:/home/aux codalab/codalab-legacy:py3

The first time around it takes a while because docker must download the entire docker image. if everything goes well, you will be inside the docker and the prompt will be like below:

root@c74a8b1ccaf7:/#

**Step 3) Run the starting kit.** Inside the docker, go to “/home/aux” and unzip and run the starting kit.

# cd /home/aux

# unzip starting\_kit.zip

# cd starting\_kit

# python3 ingestion\_program/ingestion.py sample\_data sample\_result\_submission ingestion\_program sample\_code\_submission

# python3 scoring\_program/score.py sample\_data sample\_result\_submission scoring\_output

# exit

WARNING: inside the docker, python 3.6 is called python3.

**Step 4) Run the jupyter notebook.**

Same thing as before, but add : **-p 8888:8888**

docker run -it -p 8888:8888 -v ~/aux:/home/aux codalab/codalab-legacy:py3

Go to a web browser and check that the notebook is running at http://localhost:8888/.

Then open README.pynb. WARNING: the default notebook kernel is Python 2, you’ll have to switch to Python 3. If you do not have a Python 3 kernel… you are in trouble. [Fixing it is complicated](https://ipython.readthedocs.io/en/latest/install/kernel_install.html).

**Modifying your docker**

The docker we recommend contains a lot of libraries, but you may need more for your specific competition. You can then make changes to it [following these instructions](https://github.com/codalab/codalab-competitions/wiki/User_Docker). But AVOID THIS if you can because you will then have another docker as everyone else, this may confused the L2 students. Also, you will need to test that it works fine on Codalab. If you make a new docker, it must be posted on [Docker Hub](https://hub.docker.com/). Preferable use a docker file to specify your changes and put them under revision control on Github.

How to use Git and Github

There are several good tutorials and cheat sheets on the Internet. You should start by cloning the repo of the starting kit, this is better than downloading the zip:

git clone https://github.com/madclam/m2aic2019.git

If you want to create another repo, [follow these instructions](https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/).