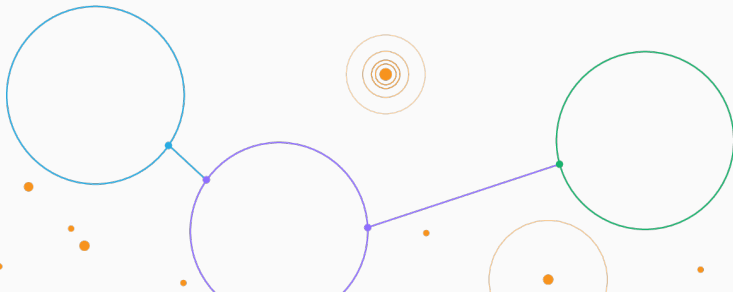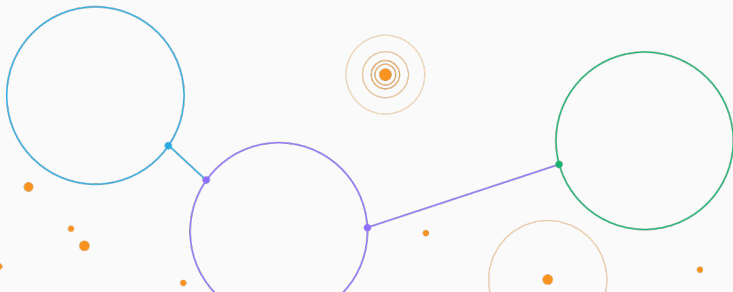# ThatMetricTimeline (TMT)

A fully-open source experiment tracking library

Alessio Molinari
April 12, 2024

About me 🙋‍♂️

Just a quick glance at my history:

- I'm originally from Rome, Italy 🧀
- Got my Master degree and later PhD in Pisa, Italy 👨‍🎓
- Now I work at STRG.at 👨🏻‍💻
- I'm a Linux and open source enthusiast 🐧

In general, I enjoy contributing to OS software. Most relevant contributions are to:

- the `Hyprland` compositor (Contributor, `https://github.com/hyprwm/Hyprland`);
- the `hyprland-virtual-desktops` plugin (Author, `https://github.com/levnikmyskin/hyprland-virtual-desktops`);
- minor contributions to Telegram, Waybar, and many other projects I don't remember anymore;

That Metric Timeline: why?
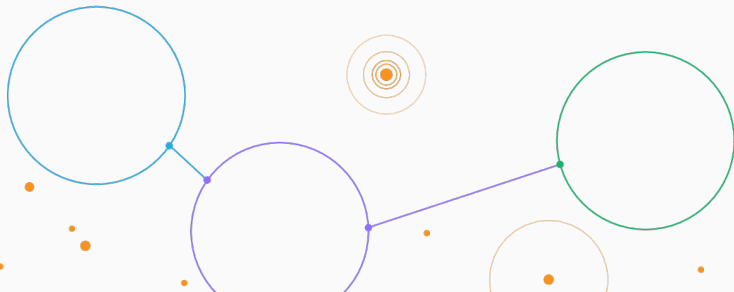
When working on a research project, experiments, their results and the code that generated them can easily get lost:

- git is not a solution, we don't commit every time we change a hyperparameter;
- some solutions were available, but they were overly complicated, or not open source;
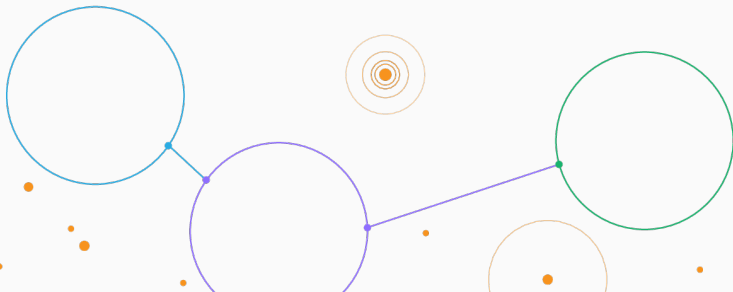
I wanted something that was:

- truly open source (read, you can contribute);
- simple (following KISS);

Where KISS means that `tmt` should be a simple software, with no unnecessary architectures, interfaces etc.
Even if this means the user needs to write their own little scripts around it.

That Metric Timeline: what?

tmt (`https://github.com/levnikmyskin/that_metric_timeline`) is a Python library which:

- keeps a local "database" of your experiments, with names, descriptions, metrics etc;
- keeps local snapshots of the code that generated those experiments;
- is fully open source, fully offline, terminal-centric (read, you can run it on your servers);

Code snapshots between different runs are created by:

- hardlinking all files that didn't change since last snapshot;
- copying only files that changed

Basically, this is to avoid your local storage explodes when you're convinced that
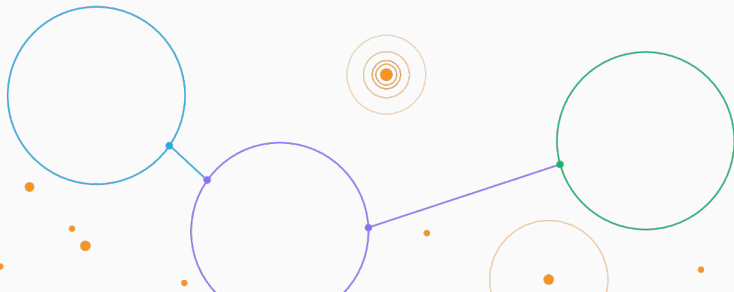those two layers are to blame for such a low F1, and you're determined to try all possible combinations by hand.

`tmt` local "database" is actually a bunch of `json` files stored in the `.tmt` folder at
the root of your repository.
You can simply read those files, but `tmt` exposes a few utilities to deal with
them.

TMT: tracking your experiments

```python
from tmt import tmt_recorder

@tmt_recorder(name="some_experiment")
def train_and_predict(x_tr, y_tr, x_te, y_te):
    lr = LogisticRegression()
    lr.fit(x_tr, y_tr)
    preds = lr.predict(x_te)
    return {
      'f1': f1_score(y_te, preds),
      'accuracy': accuracy_score(y_te, preds)
    }
```
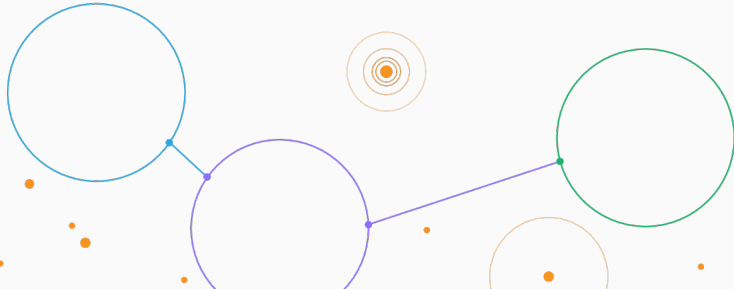
```python
from tmt import tmt_recorder, tmt_save

@tmt_recorder(name="some_experiment_with_data",
↪   description="saving preds this time")
def train_and_predict(...):
    ...
    preds = lr.predict(x_te)
    tmt_save(preds, name='lr_predictions')
    return {
        'f1': f1_score(y_te, preds),
        'accuracy': accuracy_score(y_te, preds)
    }
```

TMT: searching and loading your experiments 🔍

```
═════════════════════════════| That Metric Timeline (TMT) |═════════════════════════════
This is the Terminal User Interface (TUI) of That Metric Timeline library. You can look for recorded experiments and see their details
Choose an option from below:

                                    Search experiment by name

                                    Search experiment by ID

                                    Search experiment by date




<ctrl>q: quit   <tab/arrows>: select   <enter>: confirm
```

TMT: the TUI interface

```
                                    | Experiment: trajnozerolonger |

ID:                         fd92bc93-b9d8-40d5-b688-2db8a818ba2c

Name:                       trajnozerolonger

Description:                Trajectory with no z ... <enter> to expand

Args:                       rl_strg/experiments/navigation/graph_trajectory_emulation/graph_trajectory_emulation.py --name trajnozerolonger --desc Trajectory with no zeros. Longe
                            r trajectory -s
Date created:
                            Tue Mar  5 09:21:04 2024
Metrics:
                            <enter> to expand
Results path:
                            <enter> to expand
Code snapshot path <enter to copy>:
                            /home/alessio.molinari/rl-strg/.tmt/snapshots/fd92bc93-b9d8-40d5-b688-2db8a818ba2c
Date saved:
                            Tue Mar  5 10:40:31 2024




                        <ctrl>q: quit   <tab/arrows>: select   <enter>: confirm
```

13

```python
from tmt import TmtManager

manager = TmtManager()
manager.set_entry_by_id('example')

for name, path in manager.results_paths():
    with open(path, 'rb') as f:
        res = pickle.load(f)

for name, res in manager.load_results():
    print(res.mean())

for name, val in manager.get_metrics():
    print(f"{name}: {val}")
```
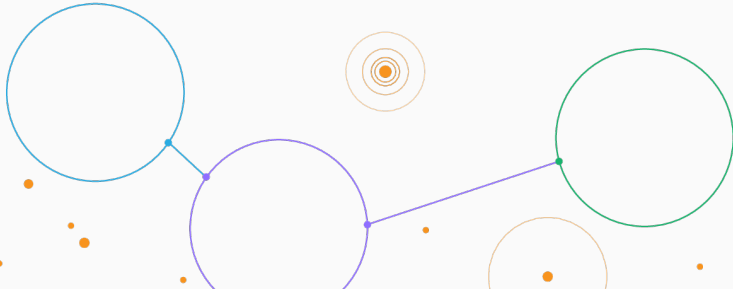
`tmt` flaws, aka I might need your help

`tmt` works and I use it more or less regularly, even today. However:

- since I finished my PhD, I don't really run so many experiments;
- I've been caught up in other open source projects;
- as a result, haven't been working on it for a while.

Mostly:

- the TUI is cool, but a CLI would be more practical;
- accessing experiments (e.g., last run) should be more straight forward;
- `TmtManager` should be able to deal wth more than one experiment;
- probably more;

Contributing to open source is cool:

- you make the software you use your own 🧑‍💻
- you get internet points 🧝
- you get resume points 👦
- you get to deal with strangers that ask you all kinds of weird stuff on your github issues 😅

`tmt` is a small project and it might be a good place to start :)

17

Sometimes it can be scary to publish or contribute code, because we think it's not good enough, or it'll be judged harshly. But we all write bad code, and that's fine.

Thank you! 👋
Questions? 🙋