DatAnalyzer

M05 – Open Science and Ethics

Mini Project

Loic Fracheboud & Clément Poncin

2023

FernUni.ch
UniDistance.ch

idiap
RESEARCH INSTITUTE

Summary

- Workflow
- Version control and collaboration
- Unit testing and CI
- Documentation
- Packaging, deployment and licensing
- Demo

[Workflow]

1) Import the data
2) Format it into manageable data
3) Split the Data
4) Preprocess with a scaling method
5) Apply a model
6) Score

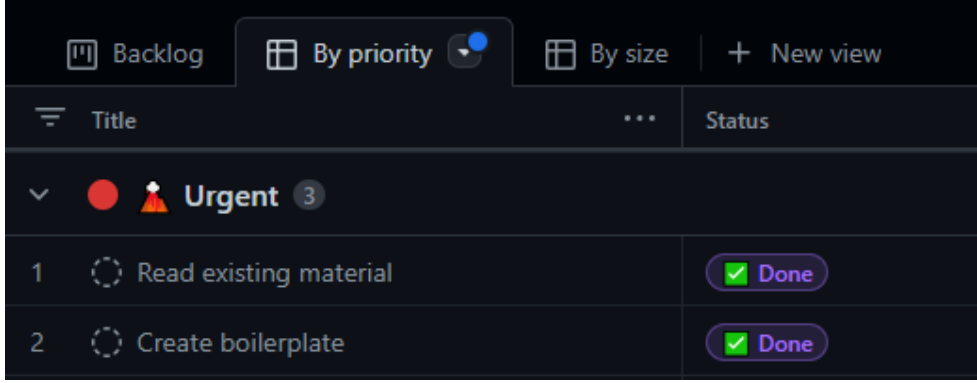
[Version Control]

The code was immediately under version control, using git and GitHub

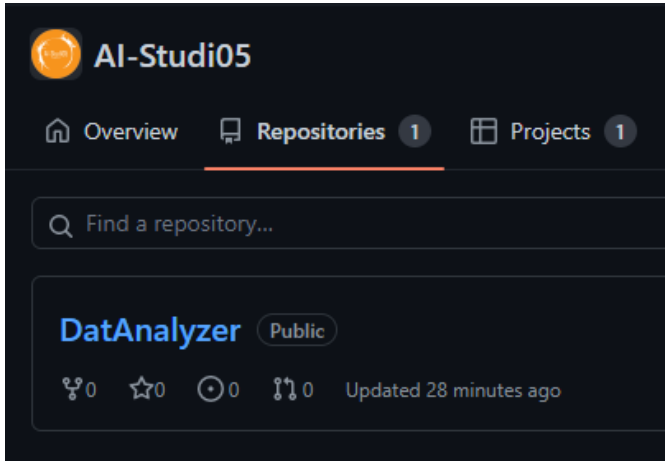
A quick task repartition was done and from there the project advanced

Done our best to keep the commit history clean

[Version Control]



- Github group, host:
  - The repo
  - The project management platform
  - The Actions (CI), the docs
  - And some policies
    - Protected main
    - Commit message format

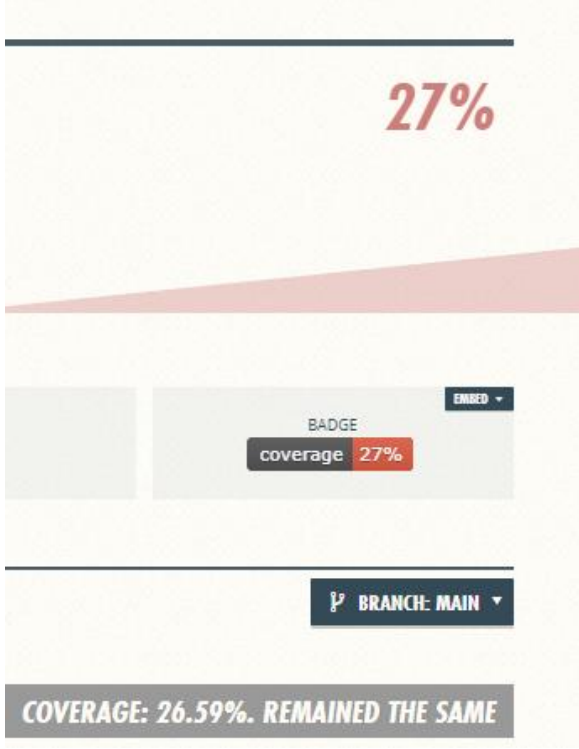


Type	Description
ADD	something new is added
REM	something was removed
CHG	something was changed
FIX	problem was corrected
MOV	something was moved
NOTE	additional important message
WARN	additional warning message

Example "ADD: new get method. MOV: resources"

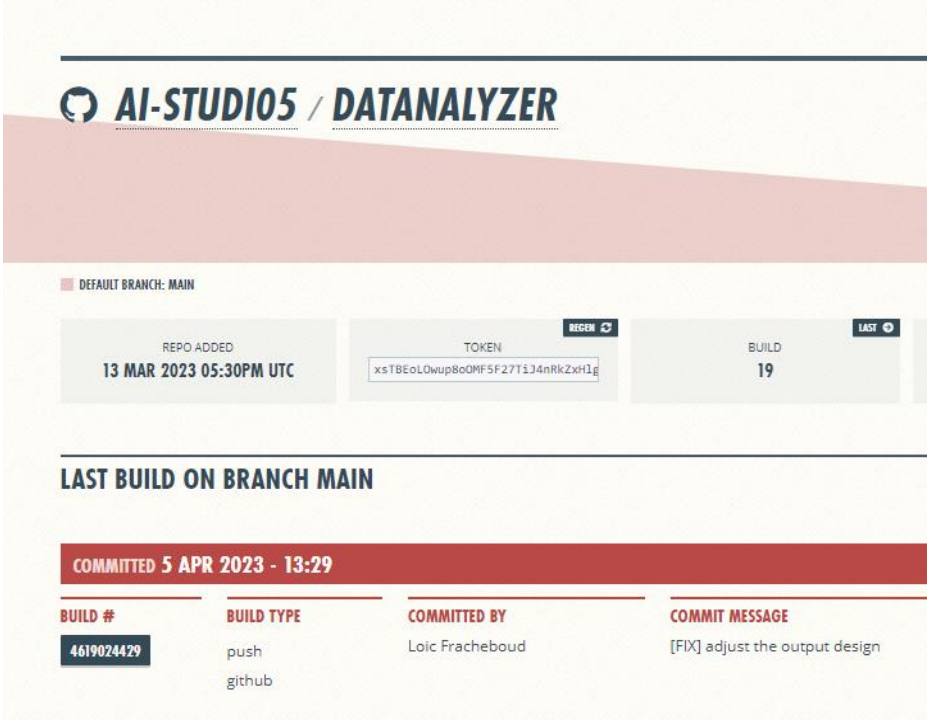
[Unit testing]

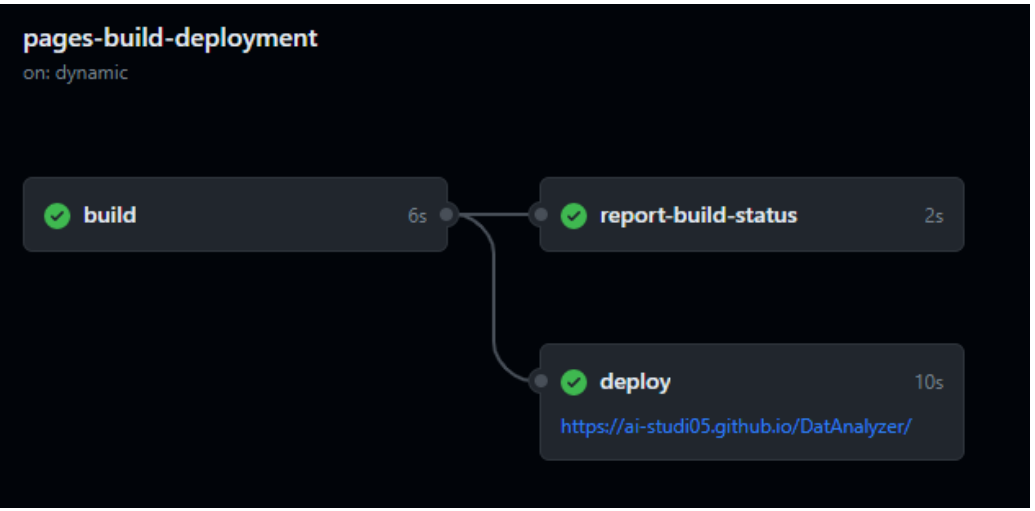
- We did some basic testings using PyTest
- The coverage is 19%
  - It was definitely not our main goal, we focussed on creating a system more than a full-solution

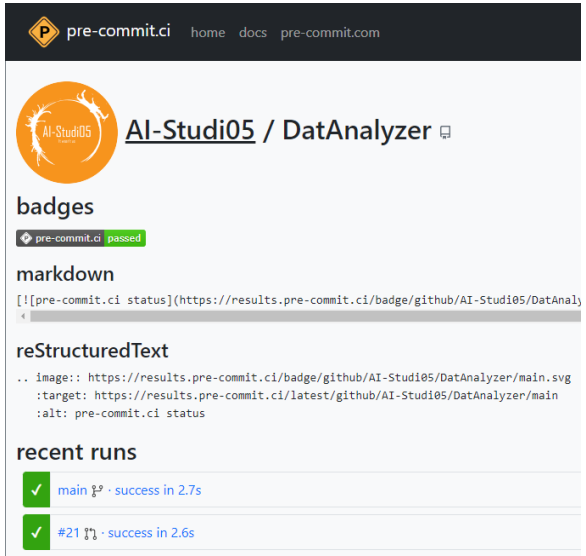


[Continuous Integration]

- CI covers:
  - Quality
  - Tests
  - Doc building and deployment







FernUni.ch
UniDistance.ch

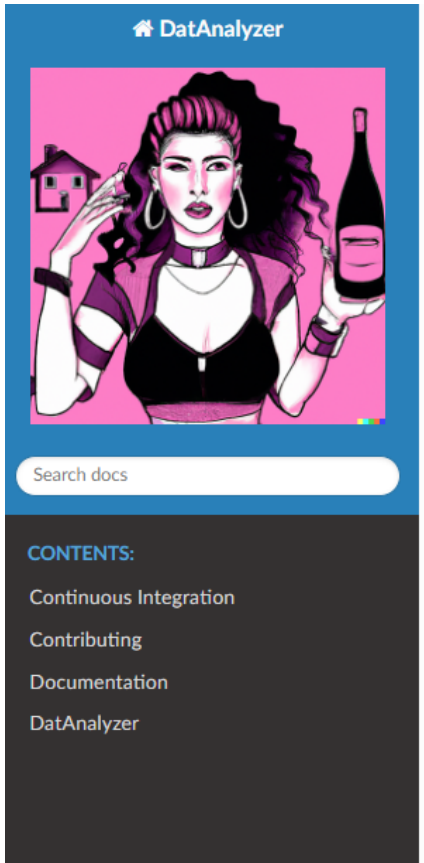
idiap
RESEARCH INSTITUTE

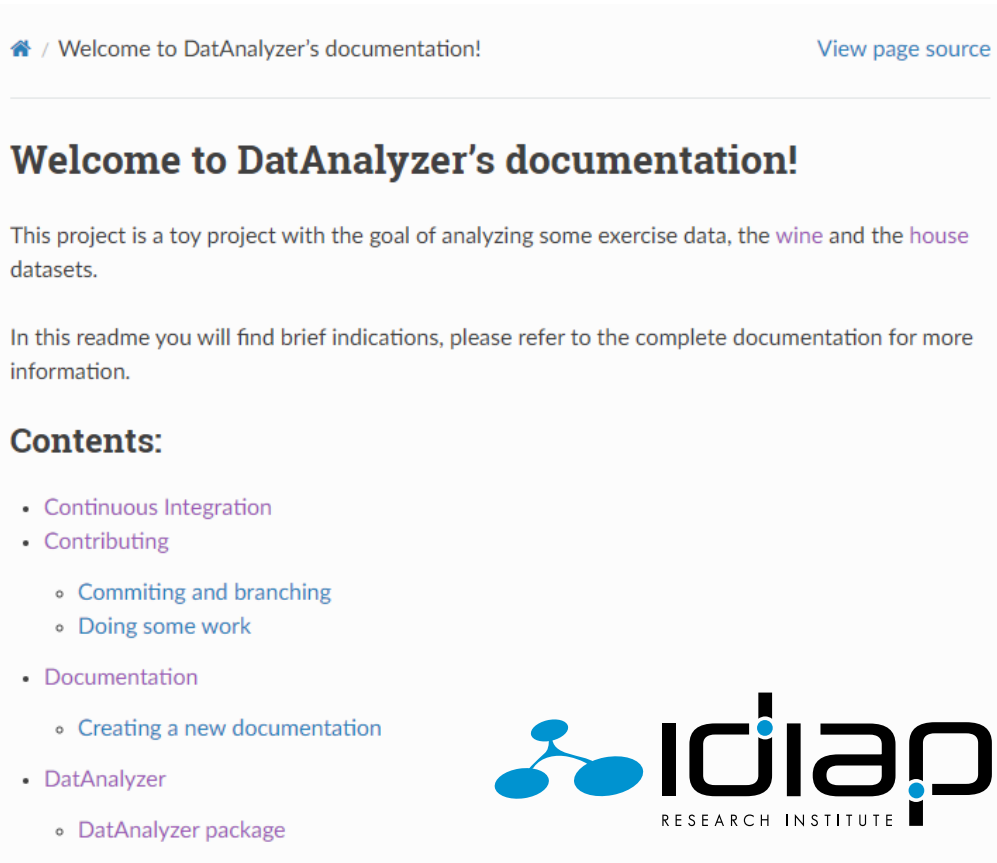
[Documentation]

Automatically generated via GitHub actions

Based on *.rst content

Deployed when merged on the main





FernUni.ch
UniDistance.ch

idiap RESEARCH INSTITUTE

[Documentation]

```yaml
- name: Sphinx build
  run: |
    sphinx-apidoc -o docs DatAnalyzer/ --separate --module-first
    sphinx-build docs _build
    touch _build/.nojekyll
```

A GitHub Action run some sphinx to convert the docstrings to an automatic doc.

```python
def fit(self, X, y=None):
    """
    Fit the data

    :param X: Data to fit
    :param y: Target

    :return: self
    :rtype: SmoothData
    """
    return self
```

fit(*X*, *y=None*) [source]

Fit the data

Parameters:
- **X** – Data to fit
- **y** – Target

Returns: self

Return type: SmoothData

[Packaging and Deployment]

1) Import the project
2) Create the environment dedicated to the project
3) Run the application to get the first result

[Licensing]

The project is licensed under the WTFPL
A very permissive license, doesn't take long to grab its philosophy

Why so?

- As a toy project, there is no strong argument for one or another
- It's fun
- But, we have to accommodate the dependencies licenses

[Licensing]

Our dependencies are:

- NumPy
- Pandas
- SciKit-Learn

All of them under the BSD licence

Then the data are under the CC0 and the Open Database Licence

To ensure that we did not miss something, we used some external resources such as JLA Compatibility Checker.

Shortly: as our project is under a very permissive licence, it is the BSD to respect.

[Sources and references]

Vectors and icons by svgrepo.com

Memes thought for too long before using imgflip.com

https://builtin.com/data-science/train-test-split

https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

http://www.wtfpl.net/

https://joinup.ec.europa.eu/collection/eupl/solution/joinup-licensing-assistant/jla-compatibility-checker

The M05 material