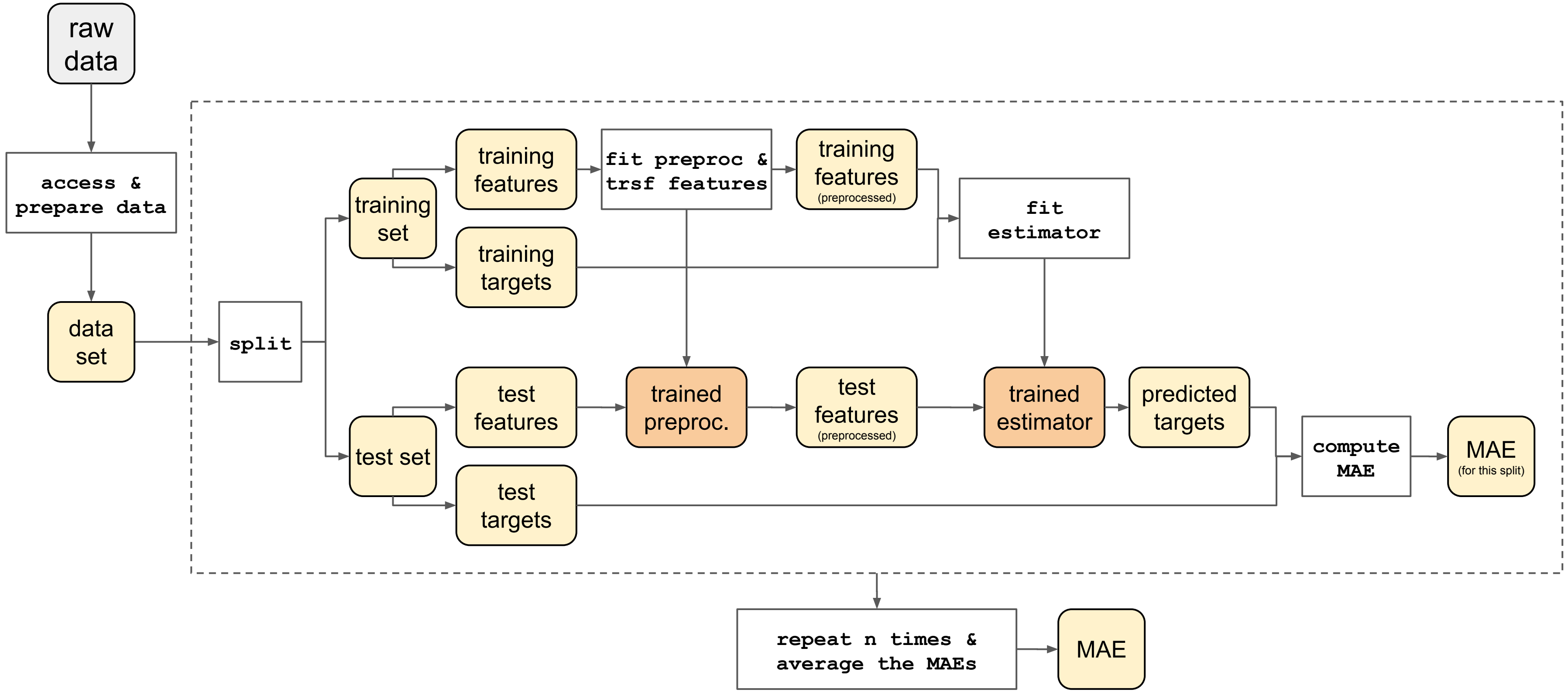
"a clipart of a city skyline, but the buildings are bottles of wine"
generated with OpenAI's DALL-E

M05 mini-project

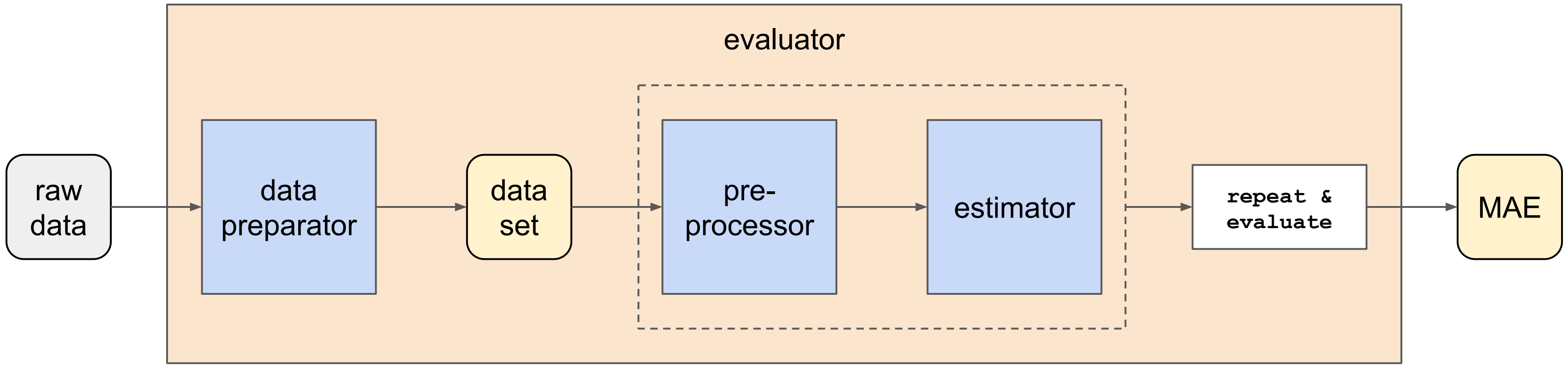
UniDistance - Master AI - SS2023

Benjamin Décaillet
Valentin Décaillet

Workflow - functional overview

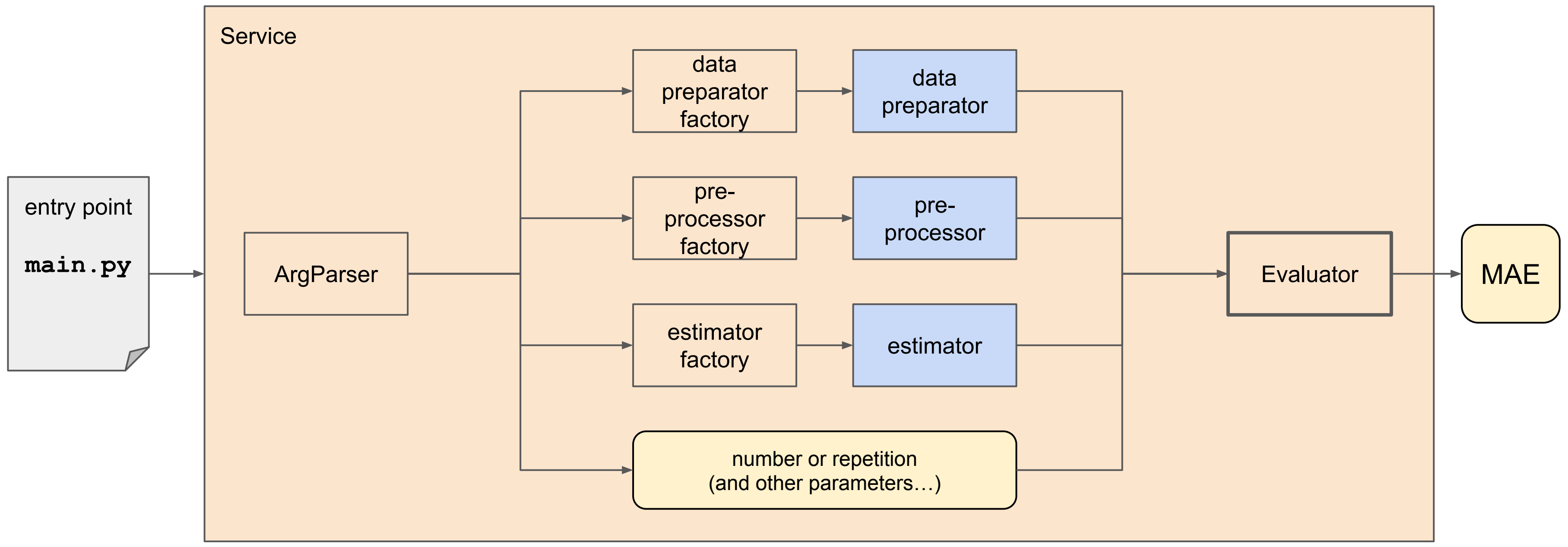


Workflow - class diagram (high level)



- advantage: the evaluator provides flexibility (swap/customize/fine-tune the “blue blocs”)
- disadvantage: constructing an evaluator is technically complex

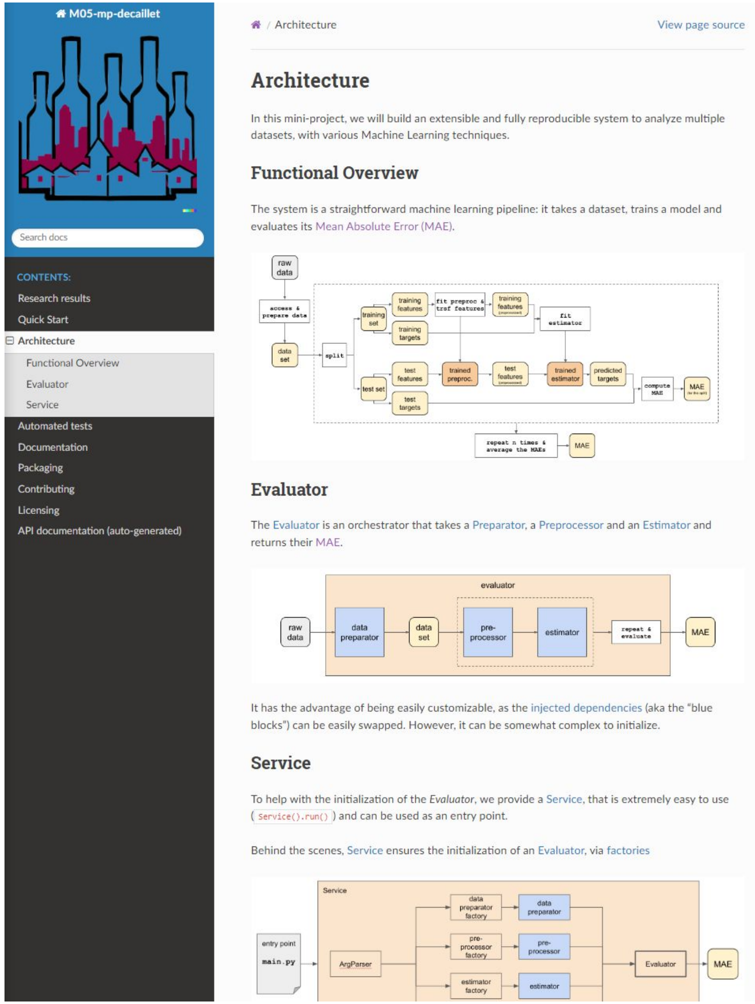
Workflow - construction of the evaluator



- Service is easy to call from any entry point (typically a one-liner: `Service().run()`)
- The entire point is to instantiate an Evaluator (which we can then run…)

Documentation

- Use sphinx
  - Accept .rst and .md files
  - Theme “read the docs”
  - Auto-generated from code and docstrings
- Links from readme.md
  - Badges to sphinx doc
- Sphinx deployed to github pages
  - CI-deploy, branch main only
  - GitHub shenanigans
    - not a jekyll site
    - permissions for github bot



M05-mp-decaillet

Search docs

CONTENTS:

Research results

Quick Start

⊟ Architecture

Functional Overview

Evaluator

Service

Automated tests

Documentation

Packaging

Contributing

Licensing

API documentation (auto-generated)

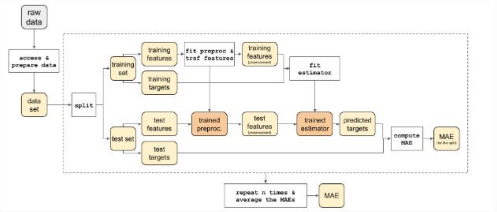
Architecture   View page source

Architecture

In this mini-project, we will build an extensible and fully reproducible system to analyze multiple datasets, with various Machine Learning techniques.

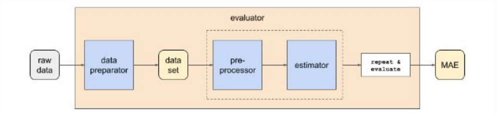
Functional Overview

The system is a straightforward machine learning pipeline: it takes a dataset, trains a model and evaluates its Mean Absolute Error (MAE).



Evaluator

The Evaluator is an orchestrator that takes a Preparator, a Preprocessor and an Estimator and returns their MAE.

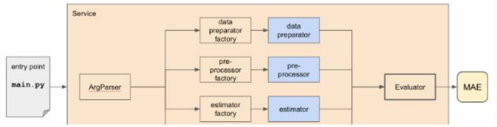


It has the advantage of being easily customizable, as the injected dependencies (aka the “blue blocks”) can be easily swapped. However, it can be somewhat complex to initialize.

Service

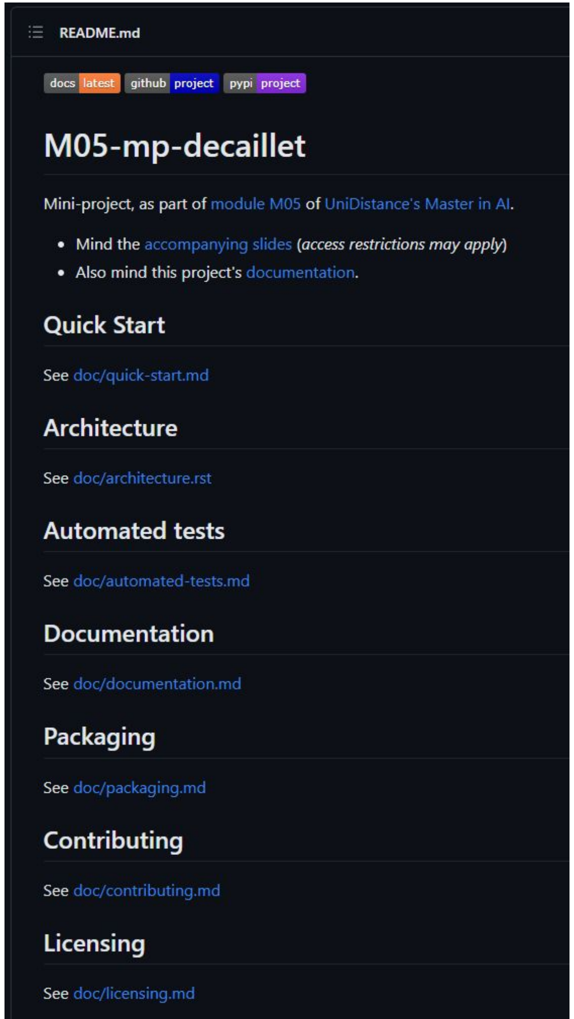
To help with the initialization of the *Evaluator*, we provide a Service, that is extremely easy to use (`Service().run()`) and can be used as an entry point.

Behind the scenes, Service ensures the initialization of an Evaluator, via factories



Documentation

- Use sphinx
  - Accept .rst and .md files
  - Theme “read the docs”
  - Auto-generated from code and docstrings
- Links from readme.md
  - Badges to sphinx doc
- Sphinx deployed to github pages
  - CI-deploy, branch main only
  - GitHub shenanigans
    - not a jekyll site
    - permissions for github bot



M05-mp-decaillet

Mini-project, as part of module M05 of UniDistance's Master in AI.

- Mind the accompanying slides (*access restrictions may apply*)
- Also mind this project's documentation.

Quick Start

See doc/quick-start.md

Architecture

See doc/architecture.rst

Automated tests

See doc/automated-tests.md

Documentation

See doc/documentation.md

Packaging

See doc/packaging.md

Contributing

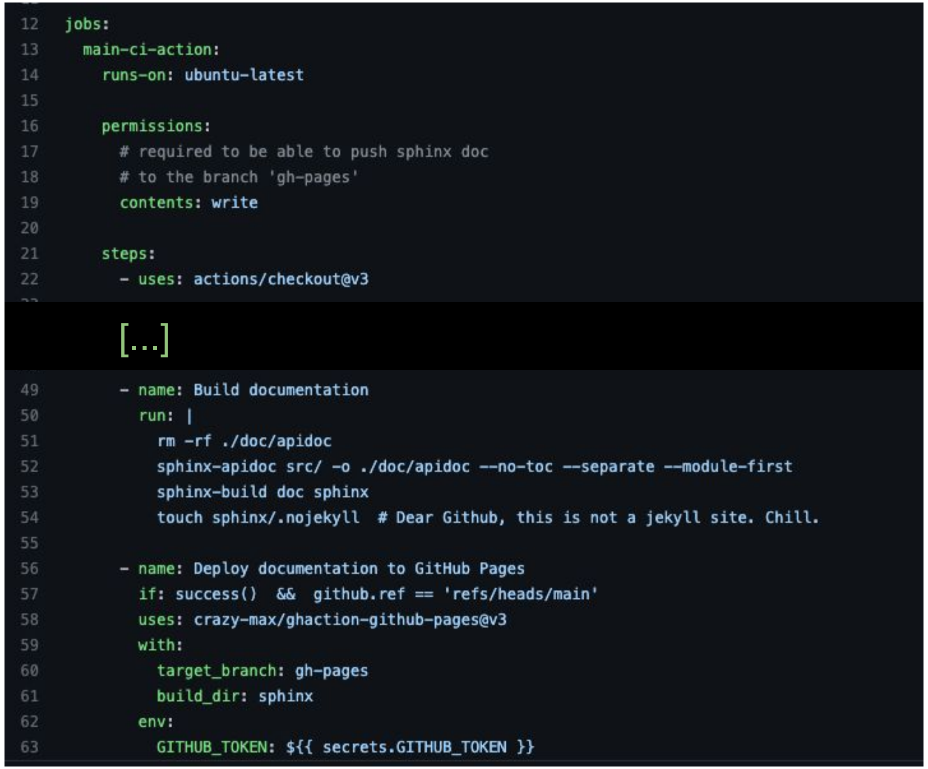
See doc/contributing.md

Licensing

See doc/licensing.md

Documentation

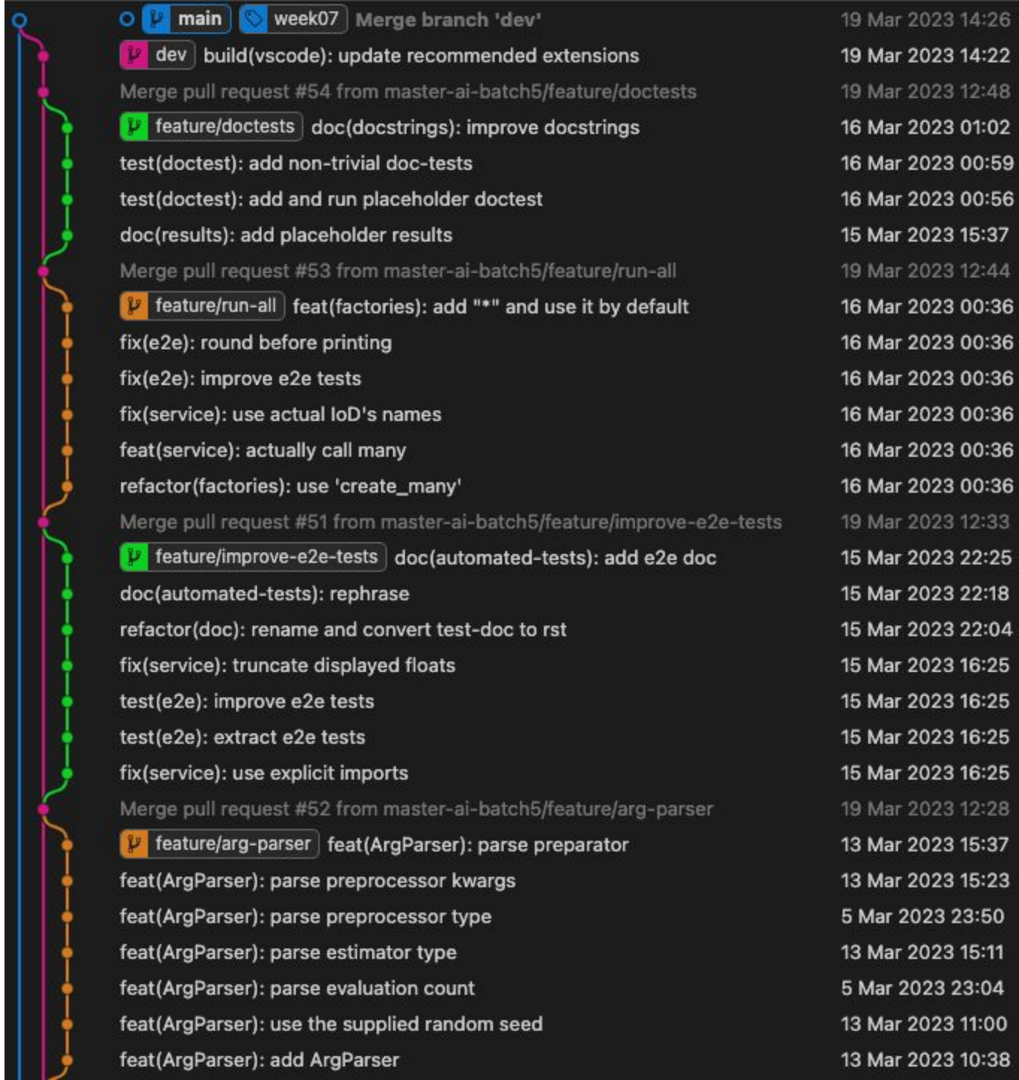
- Use sphinx
  - Accept .rst and .md files
  - Theme “read the docs”
  - Auto-generated from code and docstrings
- Links from readme.md
  - Badges to sphinx doc
- Sphinx deployed to github pages
  - CI-deploy, branch main only
  - GitHub shenanigans
    - not a jekyll site
    - permissions for github bot



```yaml
jobs:
  main-ci-action:
    runs-on: ubuntu-latest

    permissions:
      # required to be able to push sphinx doc
      # to the branch 'gh-pages'
      contents: write

    steps:
      - uses: actions/checkout@v3

      [...]

      - name: Build documentation
        run: |
          rm -rf ./doc/apidoc
          sphinx-apidoc src/ -o ./doc/apidoc --no-toc --separate --module-first
          sphinx-build doc sphinx
          touch sphinx/.nojekyll  # Dear Github, this is not a jekyll site. Chill.

      - name: Deploy documentation to GitHub Pages
        if: success()  && github.ref == 'refs/heads/main'
        uses: crazy-max/ghaction-github-pages@v3
        with:
          target_branch: gh-pages
          build_dir: sphinx
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

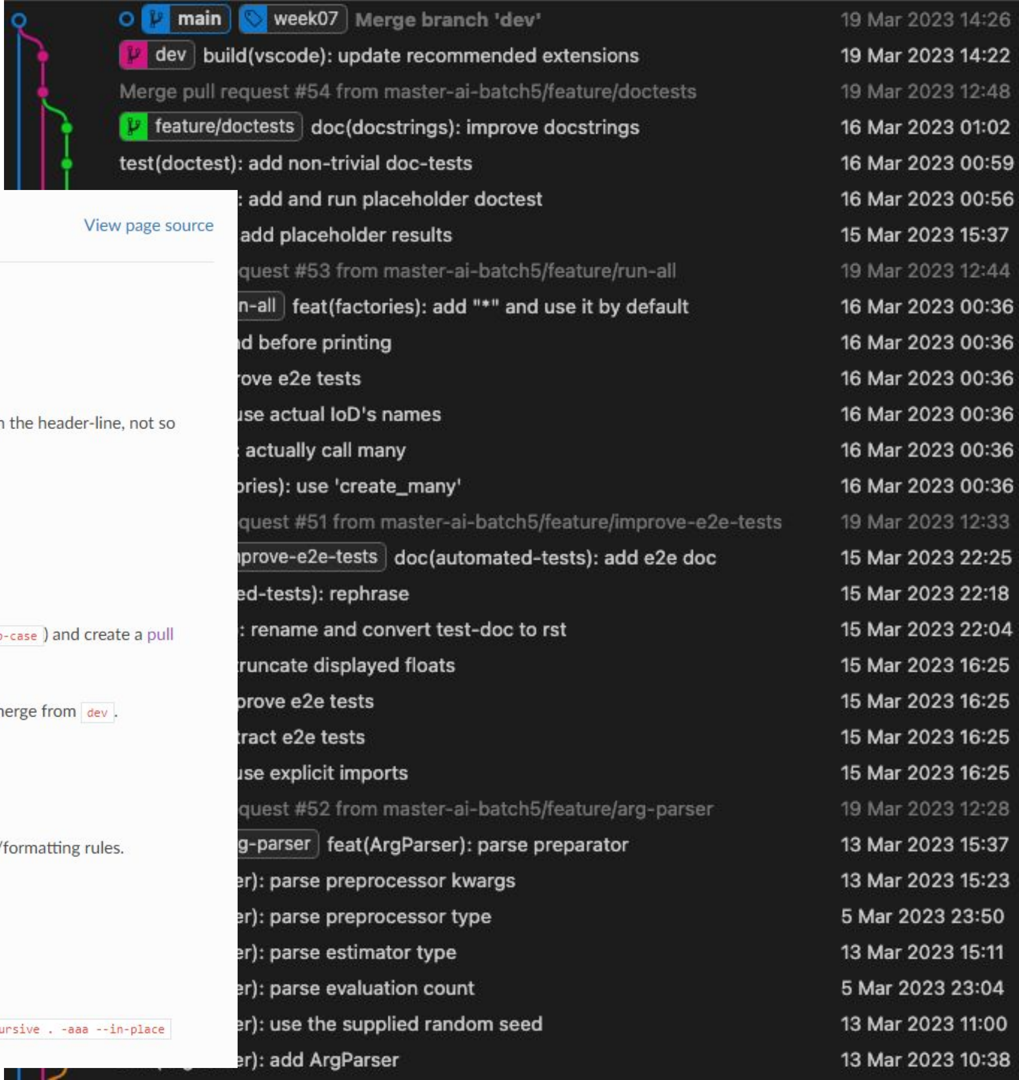
Version control

- GIT
  - feature branches
  - rebase before pull requests inspired from Gitflow
  - commit message convention inspired from Angular
- *It’s all explained in the doc*

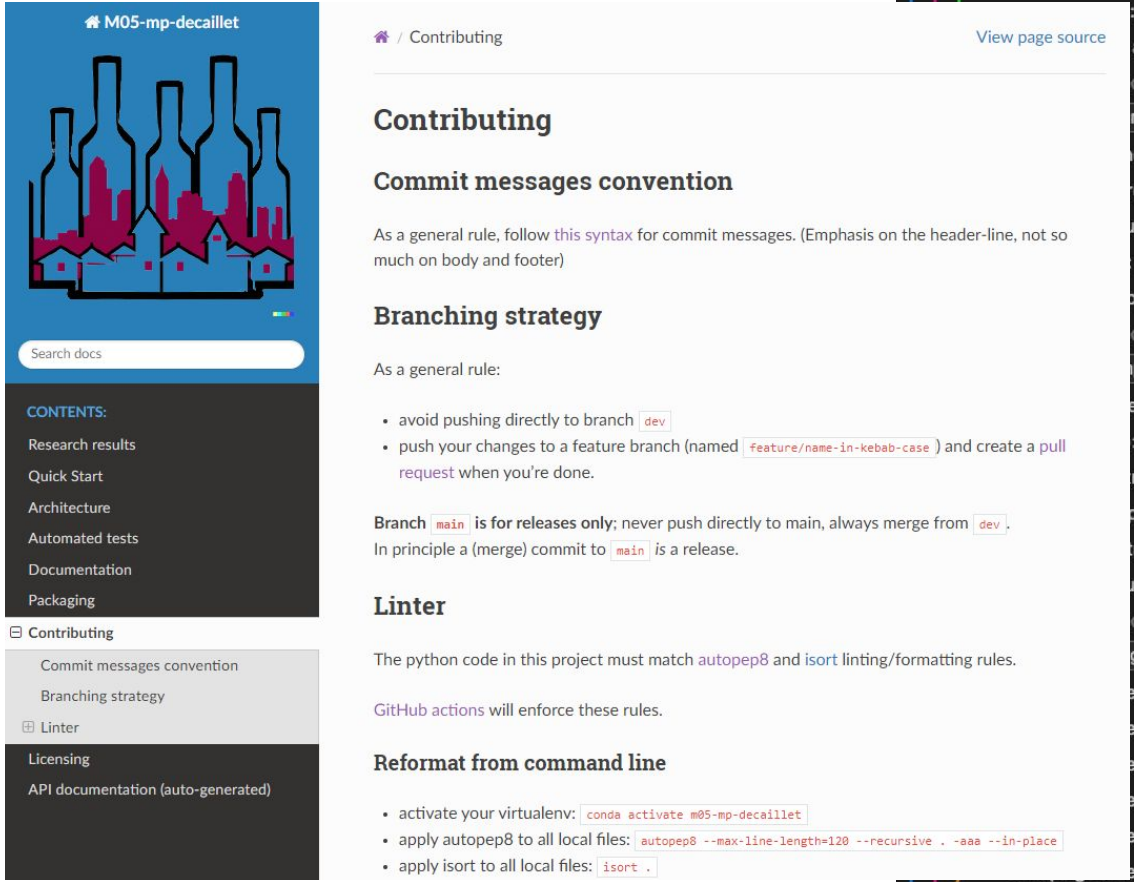


```
main  week07  Merge branch 'dev'                                                 19 Mar 2023 14:26
dev  build(vscode): update recommended extensions                                19 Mar 2023 14:22
Merge pull request #54 from master-ai-batch5/feature/doctests                    19 Mar 2023 12:48
feature/doctests  doc(docstrings): improve docstrings                            16 Mar 2023 01:02
test(doctest): add non-trivial doc-tests                                         16 Mar 2023 00:59
test(doctest): add and run placeholder doctest                                   16 Mar 2023 00:56
doc(results): add placeholder results                                            15 Mar 2023 15:37
Merge pull request #53 from master-ai-batch5/feature/run-all                     19 Mar 2023 12:44
feature/run-all  feat(factories): add "*" and use it by default                  16 Mar 2023 00:36
fix(e2e): round before printing                                                  16 Mar 2023 00:36
fix(e2e): improve e2e tests                                                      16 Mar 2023 00:36
fix(service): use actual IoD's names                                             16 Mar 2023 00:36
feat(service): actually call many                                                16 Mar 2023 00:36
refactor(factories): use 'create_many'                                           16 Mar 2023 00:36
Merge pull request #51 from master-ai-batch5/feature/improve-e2e-tests           19 Mar 2023 12:33
feature/improve-e2e-tests  doc(automated-tests): add e2e doc                     15 Mar 2023 22:25
doc(automated-tests): rephrase                                                   15 Mar 2023 22:18
refactor(doc): rename and convert test-doc to rst                                15 Mar 2023 22:04
fix(service): truncate displayed floats                                          15 Mar 2023 16:25
test(e2e): improve e2e tests                                                     15 Mar 2023 16:25
test(e2e): extract e2e tests                                                     15 Mar 2023 16:25
fix(service): use explicit imports                                               15 Mar 2023 16:25
Merge pull request #52 from master-ai-batch5/feature/arg-parser                  19 Mar 2023 12:28
feature/arg-parser  feat(ArgParser): parse preparator                            13 Mar 2023 15:37
feat(ArgParser): parse preprocessor kwargs                                       13 Mar 2023 15:23
feat(ArgParser): parse preprocessor type                                         5 Mar 2023 23:50
feat(ArgParser): parse estimator type                                            13 Mar 2023 15:11
feat(ArgParser): parse evaluation count                                          5 Mar 2023 23:04
feat(ArgParser): use the supplied random seed                                    13 Mar 2023 11:00
feat(ArgParser): add ArgParser                                                   13 Mar 2023 10:38
```

Version control



```
main  week07  Merge branch 'dev'                                        19 Mar 2023 14:26
dev  build(vscode): update recommended extensions                       19 Mar 2023 14:22
Merge pull request #54 from master-ai-batch5/feature/doctests          19 Mar 2023 12:48
feature/doctests  doc(docstrings): improve docstrings                   16 Mar 2023 01:02
test(doctest): add non-trivial doc-tests                                16 Mar 2023 00:59
: add and run placeholder doctest                                       16 Mar 2023 00:56
add placeholder results                                                 15 Mar 2023 15:37
quest #53 from master-ai-batch5/feature/run-all                         19 Mar 2023 12:44
n-all  feat(factories): add "*" and use it by default                   16 Mar 2023 00:36
d before printing                                                       16 Mar 2023 00:36
rove e2e tests                                                          16 Mar 2023 00:36
use actual IoD's names                                                  16 Mar 2023 00:36
: actually call many                                                    16 Mar 2023 00:36
ories): use 'create_many'                                               16 Mar 2023 00:36
quest #51 from master-ai-batch5/feature/improve-e2e-tests               19 Mar 2023 12:33
nprove-e2e-tests  doc(automated-tests): add e2e doc                     15 Mar 2023 22:25
ed-tests): rephrase                                                     15 Mar 2023 22:18
: rename and convert test-doc to rst                                    15 Mar 2023 22:04
runcate displayed floats                                                15 Mar 2023 16:25
prove e2e tests                                                         15 Mar 2023 16:25
ract e2e tests                                                          15 Mar 2023 16:25
use explicit imports                                                    15 Mar 2023 16:25
quest #52 from master-ai-batch5/feature/arg-parser                      19 Mar 2023 12:28
g-parser  feat(ArgParser): parse preparator                             13 Mar 2023 15:37
er): parse preprocessor kwargs                                          13 Mar 2023 15:23
er): parse preprocessor type                                            5 Mar 2023 23:50
er): parse estimator type                                               13 Mar 2023 15:11
er): parse evaluation count                                             5 Mar 2023 23:04
er): use the supplied random seed                                       13 Mar 2023 11:00
er): add ArgParser                                                      13 Mar 2023 10:38
```



M05-mp-decaillet

Search docs

CONTENTS:

- Research results
- Quick Start
- Architecture
- Automated tests
- Documentation
- Packaging
- Contributing
  - Commit messages convention
  - Branching strategy
  - Linter
- Licensing
- API documentation (auto-generated)

Contributing — View page source

Contributing

Commit messages convention

As a general rule, follow this syntax for commit messages. (Emphasis on the header-line, not so much on body and footer)

Branching strategy

As a general rule:

- avoid pushing directly to branch `dev`
- push your changes to a feature branch (named `feature/name-in-kebab-case`) and create a pull request when you're done.

Branch `main` is for releases only; never push directly to main, always merge from `dev`. In principle a (merge) commit to `main` *is* a release.

Linter

The python code in this project must match autopep8 and isort linting/formatting rules.

GitHub actions will enforce these rules.

Reformat from command line

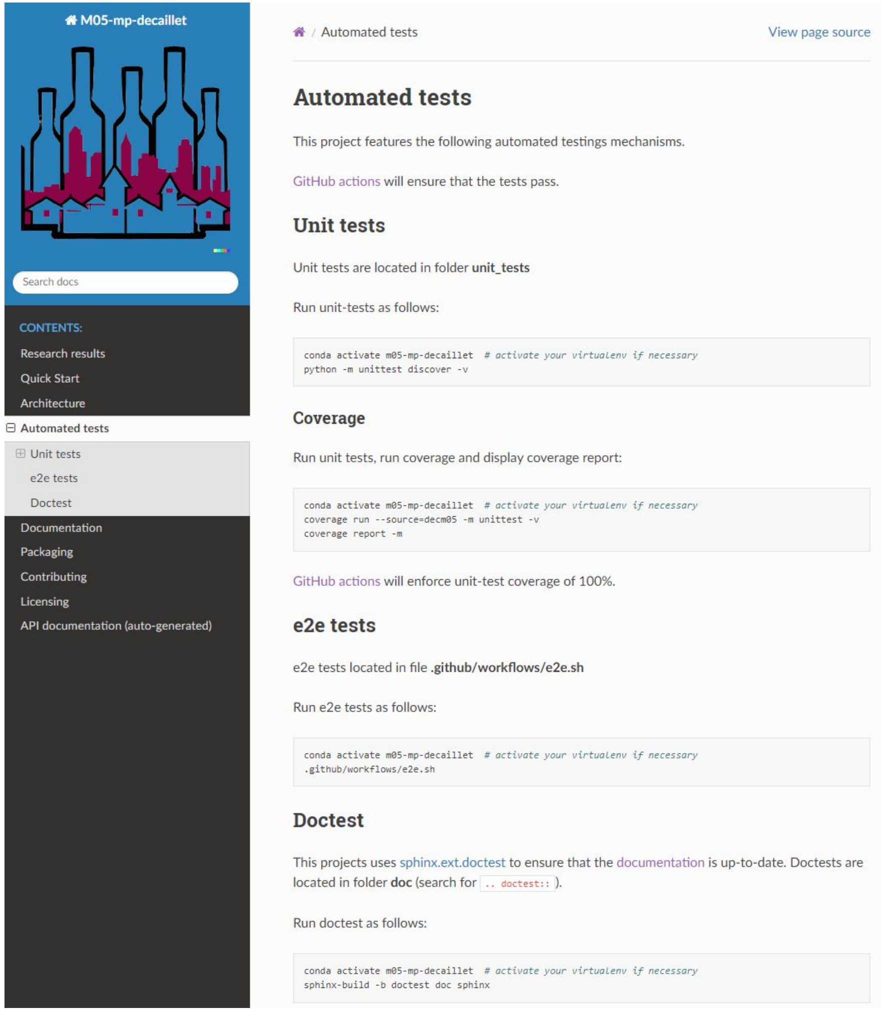
- activate your virtualenv: `conda activate m05-mp-decaillet`
- apply autopep8 to all local files: `autopep8 --max-line-length=120 --recursive . -aaa --in-place`
- apply isort to all local files: `isort .`

Code Sharing (GitHub)

- Organization created: “master-ai-batch5”
- Repository under the organization → Role management for Repository
- Protection on branches:
  - main:
    - Require linear history
    - Can only be updated from dev
  - dev: Set up as default
    - Require a pull request before merging
    - Require approvals
    - Dismiss stale pull request approvals when new commits are pushed
    - Require conversation resolution before merging
    - Allow force pushes from certain member only
  - feature branches: no security
- Issues used to track tasks and assignments

Automated testing

- unit-tests
- e2e tests
- doctests



/ Automated tests — View page source

Automated tests

This project features the following automated testings mechanisms.

GitHub actions will ensure that the tests pass.

Unit tests

Unit tests are located in folder **unit_tests**

Run unit-tests as follows:

```
conda activate m05-mp-decaillet  # activate your virtualenv if necessary
python -m unittest discover -v
```

Coverage

Run unit tests, run coverage and display coverage report:

```
conda activate m05-mp-decaillet  # activate your virtualenv if necessary
coverage run --source=decm05 -m unittest -v
coverage report -m
```

GitHub actions will enforce unit-test coverage of 100%.

e2e tests

e2e tests located in file **.github/workflows/e2e.sh**

Run e2e tests as follows:

```
conda activate m05-mp-decaillet  # activate your virtualenv if necessary
.github/workflows/e2e.sh
```

Doctest

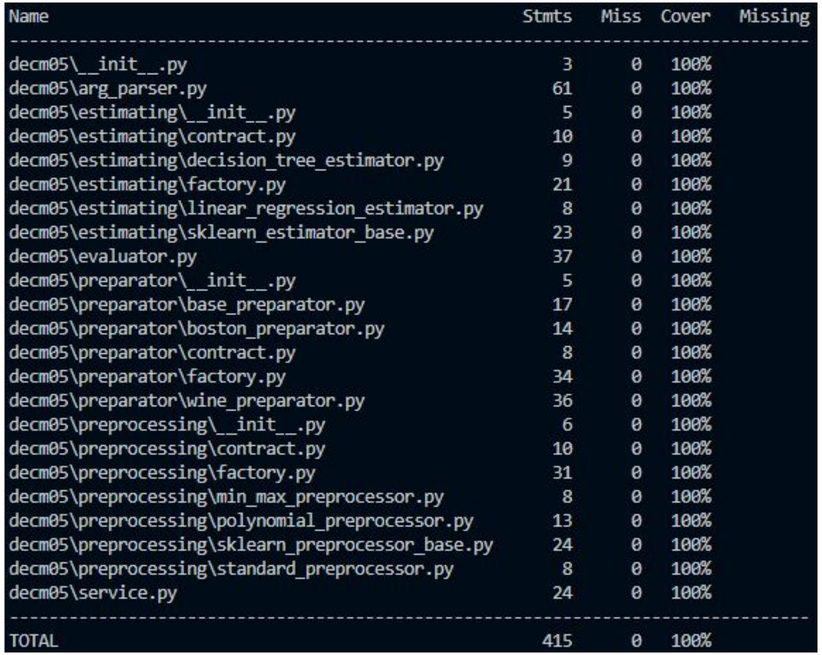
This projects uses sphinx.ext.doctest to ensure that the documentation is up-to-date. Doctests are located in folder **doc** (search for `.. doctest::`).

Run doctest as follows:

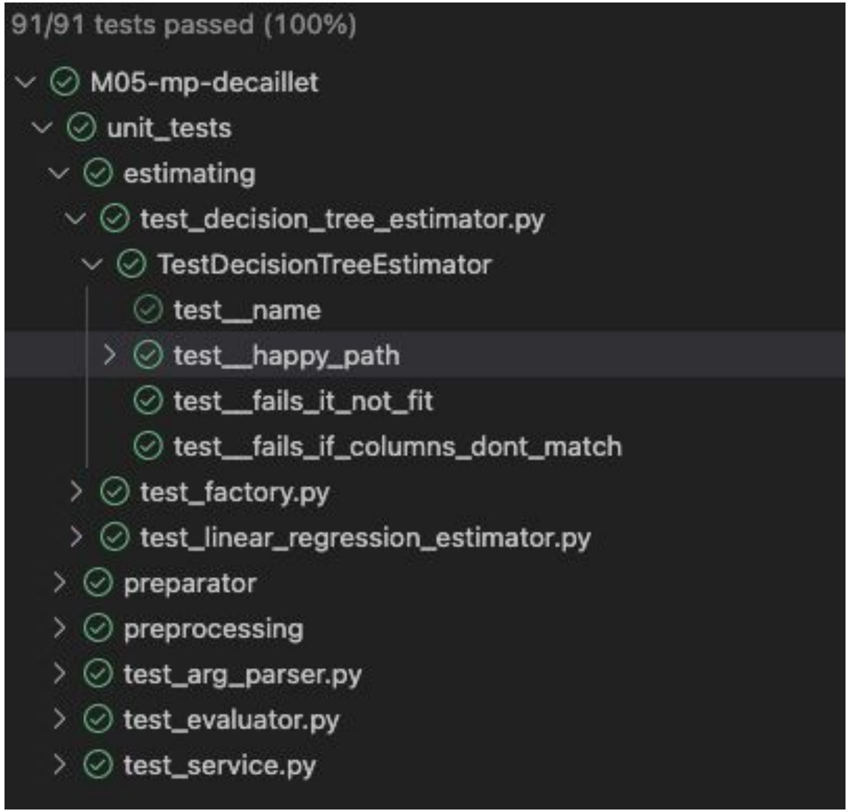
```
conda activate m05-mp-decaillet  # activate your virtualenv if necessary
sphinx-build -b doctest doc sphinx
```

Automated testing - unit-tests

- native ‘unittest’ framework
- coverage 100%



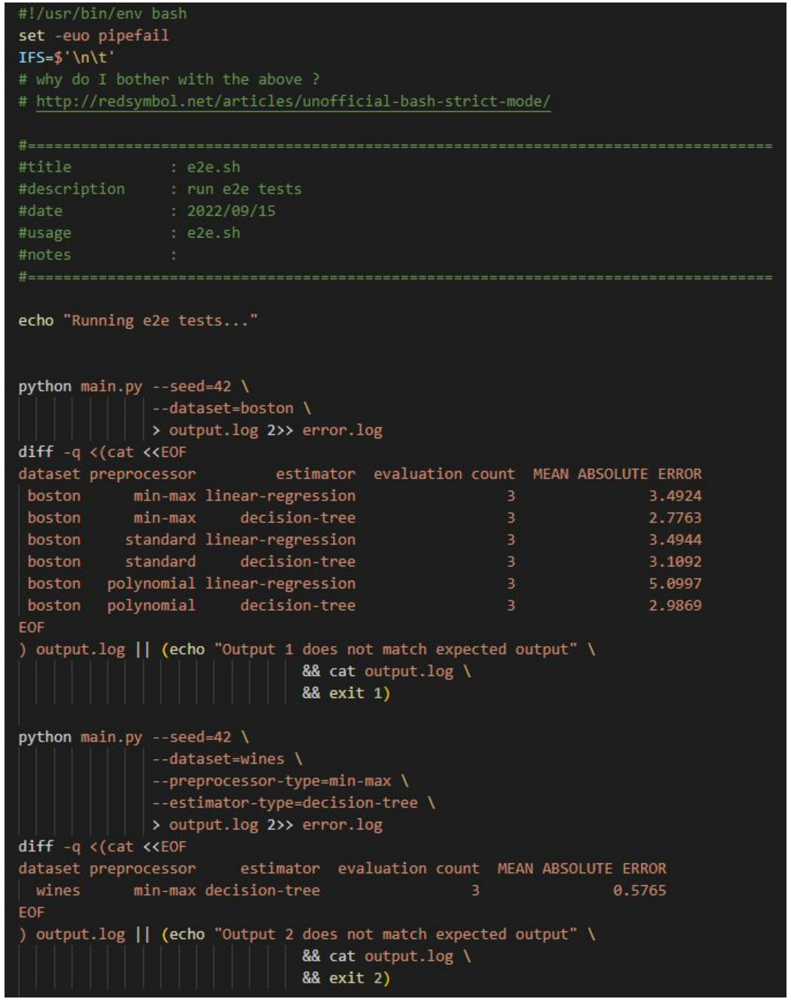
```
Name                                              Stmts   Miss  Cover   Missing
-------------------------------------------------------------------------------
decm05\__init__.py                                    3      0   100%
decm05\arg_parser.py                                 61      0   100%
decm05\estimating\__init__.py                         5      0   100%
decm05\estimating\contract.py                        10      0   100%
decm05\estimating\decision_tree_estimator.py          9      0   100%
decm05\estimating\factory.py                         21      0   100%
decm05\estimating\linear_regression_estimator.py      8      0   100%
decm05\estimating\sklearn_estimator_base.py          23      0   100%
decm05\evaluator.py                                  37      0   100%
decm05\preparator\__init__.py                         5      0   100%
decm05\preparator\base_preparator.py                 17      0   100%
decm05\preparator\boston_preparator.py               14      0   100%
decm05\preparator\contract.py                         8      0   100%
decm05\preparator\factory.py                         34      0   100%
decm05\preparator\wine_preparator.py                 36      0   100%
decm05\preprocessing\__init__.py                      6      0   100%
decm05\preprocessing\contract.py                     10      0   100%
decm05\preprocessing\factory.py                      31      0   100%
decm05\preprocessing\min_max_preprocessor.py          8      0   100%
decm05\preprocessing\polynomial_preprocessor.py      13      0   100%
decm05\preprocessing\sklearn_preprocessor_base.py    24      0   100%
decm05\preprocessing\standard_preprocessor.py         8      0   100%
decm05\service.py                                    24      0   100%
-------------------------------------------------------------------------------
TOTAL                                               415      0   100%
```



```
91/91 tests passed (100%)
⌄ ✓ M05-mp-decaillet
  ⌄ ✓ unit_tests
    ⌄ ✓ estimating
      ⌄ ✓ test_decision_tree_estimator.py
        ⌄ ✓ TestDecisionTreeEstimator
            ✓ test__name
          › ✓ test__happy_path
            ✓ test__fails_it_not_fit
            ✓ test__fails_if_columns_dont_match
      › ✓ test_factory.py
      › ✓ test_linear_regression_estimator.py
    › ✓ preparator
    › ✓ preprocessing
    › ✓ test_arg_parser.py
    › ✓ test_evaluator.py
    › ✓ test_service.py
```

Automated testing - e2e tests

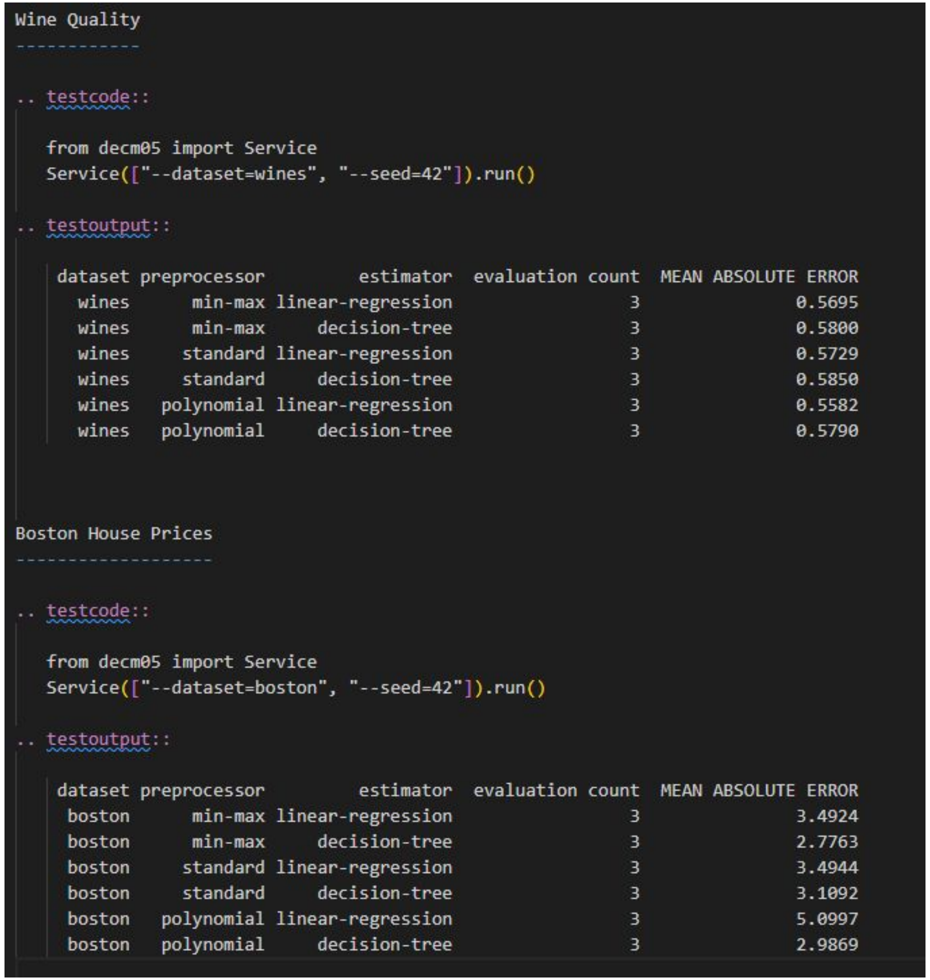
- a bash script



```bash
#!/usr/bin/env bash
set -euo pipefail
IFS=$'\n\t'
# why do I bother with the above ?
# http://redsymbol.net/articles/unofficial-bash-strict-mode/

#===================================================================================
#title           : e2e.sh
#description     : run e2e tests
#date            : 2022/09/15
#usage           : e2e.sh
#notes           :
#===================================================================================

echo "Running e2e tests..."


python main.py --seed=42 \
               --dataset=boston \
               > output.log 2>> error.log
diff -q <(cat <<EOF
dataset preprocessor         estimator  evaluation count  MEAN ABSOLUTE ERROR
 boston      min-max linear-regression                 3               3.4924
 boston      min-max     decision-tree                 3               2.7763
 boston     standard linear-regression                 3               3.4944
 boston     standard     decision-tree                 3               3.1092
 boston   polynomial linear-regression                 3               5.0997
 boston   polynomial     decision-tree                 3               2.9869
EOF
) output.log || (echo "Output 1 does not match expected output" \
                                && cat output.log \
                                && exit 1)

python main.py --seed=42 \
               --dataset=wines \
               --preprocessor-type=min-max \
               --estimator-type=decision-tree \
               > output.log 2>> error.log
diff -q <(cat <<EOF
dataset preprocessor     estimator  evaluation count  MEAN ABSOLUTE ERROR
  wines      min-max decision-tree                 3               0.5765
EOF
) output.log || (echo "Output 2 does not match expected output" \
                                && cat output.log \
                                && exit 2)
```

Automated testing - doctests

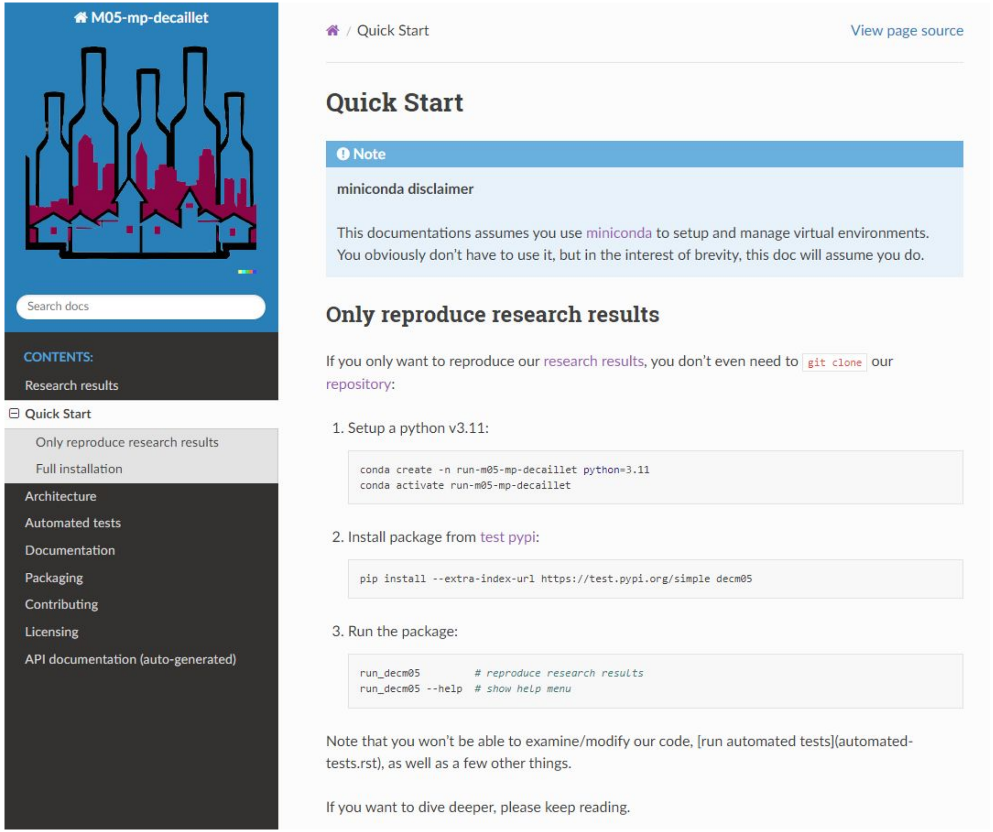
- sphinx.ext.doctest



```
Wine Quality
------------

.. testcode::

   from decm05 import Service
   Service(["--dataset=wines", "--seed=42"]).run()

.. testoutput::

    dataset preprocessor         estimator  evaluation count  MEAN ABSOLUTE ERROR
      wines      min-max linear-regression                 3               0.5695
      wines      min-max     decision-tree                 3               0.5800
      wines     standard linear-regression                 3               0.5729
      wines     standard     decision-tree                 3               0.5850
      wines   polynomial linear-regression                 3               0.5582
      wines   polynomial     decision-tree                 3               0.5790


Boston House Prices
-------------------

.. testcode::

   from decm05 import Service
   Service(["--dataset=boston", "--seed=42"]).run()

.. testoutput::

    dataset preprocessor         estimator  evaluation count  MEAN ABSOLUTE ERROR
     boston      min-max linear-regression                 3               3.4924
     boston      min-max     decision-tree                 3               2.7763
     boston     standard linear-regression                 3               3.4944
     boston     standard     decision-tree                 3               3.1092
     boston   polynomial linear-regression                 3               5.0997
     boston   polynomial     decision-tree                 3               2.9869
```

Environment management

- Python 3.11
  - using conda virtual environments
  - dependencies pinned in `build-requirements.txt` and `dev-requirements.txt`
- "**quick start**" section in documentation:
  - project-specific command lines
  - links to 3rd parties



Quick Start

Note

miniconda disclaimer

This documentations assumes you use miniconda to setup and manage virtual environments. You obviously don't have to use it, but in the interest of brevity, this doc will assume you do.

Only reproduce research results

If you only want to reproduce our research results, you don't even need to `git clone` our repository:

1. Setup a python v3.11:

```
conda create -n run-m05-mp-decaillet python=3.11
conda activate run-m05-mp-decaillet
```

2. Install package from test pypi:

```
pip install --extra-index-url https://test.pypi.org/simple decm05
```

3. Run the package:

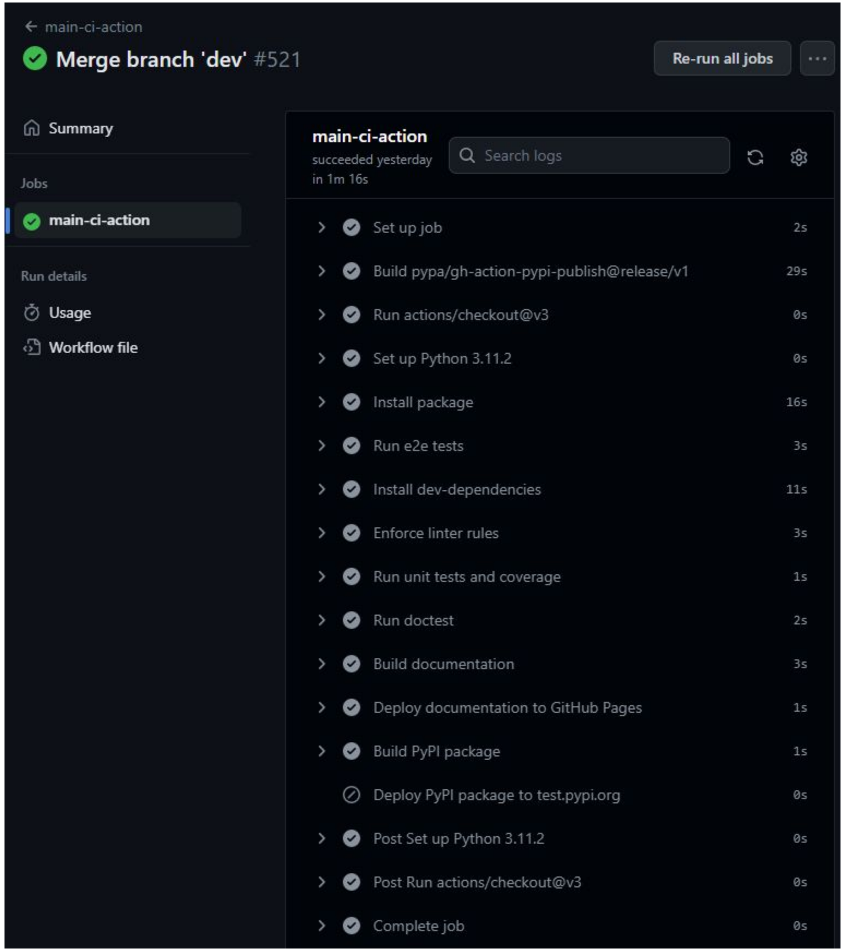
```
run_decm05         # reproduce research results
run_decm05 --help  # show help menu
```

Note that you won't be able to examine/modify our code, [run automated tests](automated-tests.rst), as well as a few other things.

If you want to dive deeper, please keep reading.

Continuous Integration (CI)

- run for each
  - push (branch / version tags)
  - pull request
- one job, several steps
  - cf image
  - abort on failure
- conditional deploy steps
  - deploy doc only for branch `main`
  - deploy to PyPI only for version tags



Merge branch 'dev' #521

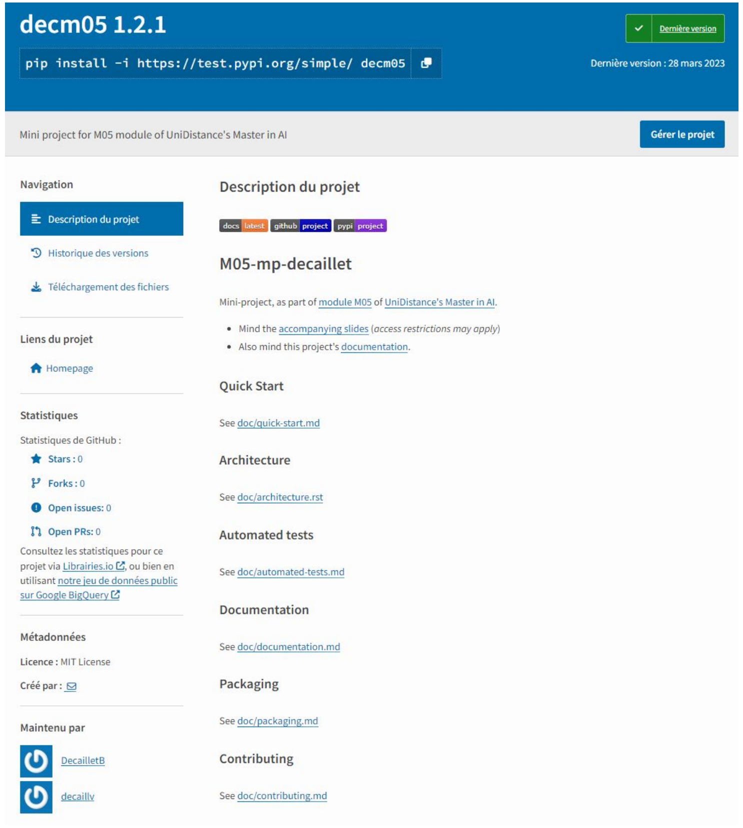
main-ci-action

main-ci-action
succeeded yesterday in 1m 16s

```
Set up job                                          2s
Build pypa/gh-action-pypi-publish@release/v1        29s
Run actions/checkout@v3                             0s
Set up Python 3.11.2                                0s
Install package                                     16s
Run e2e tests                                       3s
Install dev-dependencies                            11s
Enforce linter rules                                3s
Run unit tests and coverage                         1s
Run doctest                                         2s
Build documentation                                 3s
Deploy documentation to GitHub Pages                1s
Build PyPI package                                  1s
Deploy PyPI package to test.pypi.org                0s
Post Set up Python 3.11.2                           0s
Post Run actions/checkout@v3                        0s
Complete job                                        0s
```

Packaging

- Standard python package repository PyPi
  - only deployed on test https://test.pypi.org/
  - package name decm05
- setuptools
- dependencies pinned with two requirements files
  - “build-requirements.txt” contains runtime dependencies
  - “dev-requirements.txt” contains all other dependencies
- Semantic version numbering i.e: “1.0.0”
  - tag must match [0-99].+[0-99].+[0-99]
  - without “alpha,beta, etc” version candidates
- CI uploading new version directly on test PyPi



decm05 1.2.1

```
pip install -i https://test.pypi.org/simple/ decm05
```

Dernière version

Dernière version : 28 mars 2023

Mini project for M05 module of UniDistance's Master in AI

Gérer le projet

Navigation

- Description du projet
- Historique des versions
- Téléchargement des fichiers

Liens du projet

- Homepage

Statistiques

Statistiques de GitHub :

- Stars : 0
- Forks : 0
- Open issues: 0
- Open PRs: 0

Consultez les statistiques pour ce projet via Librairies.io, ou bien en utilisant notre jeu de données public sur Google BigQuery

Métadonnées

Licence : MIT License

Créé par :

Maintenu par

DecailletB

decaillv

Description du projet

docs latest | github project | pypi project

M05-mp-decaillet

Mini-project, as part of module M05 of UniDistance's Master in AI.

- Mind the accompanying slides (*access restrictions may apply*)
- Also mind this project's documentation.

Quick Start

See doc/quick-start.md

Architecture

See doc/architecture.rst

Automated tests

See doc/automated-tests.md

Documentation

See doc/documentation.md

Packaging

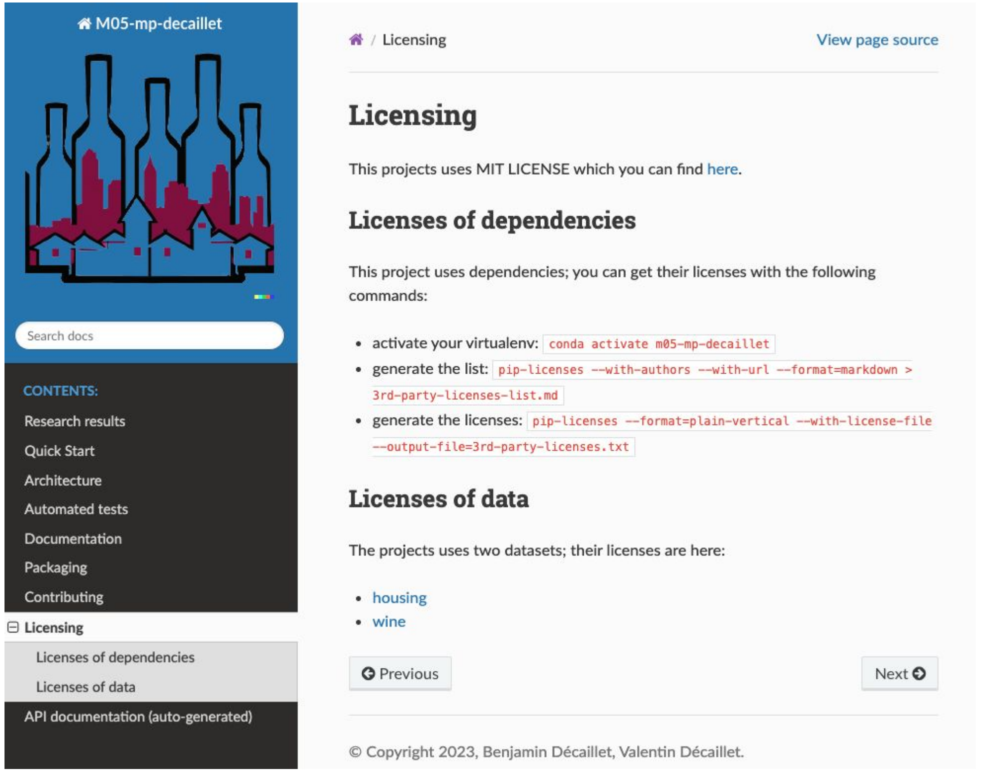
See doc/packaging.md

Contributing

See doc/contributing.md

Licensing

- MIT licensing
- Licenses of dependencies
  - Auto-generated with pip-licenses
  - See our documentation for generation
- Licenses of data
  - *.names file containing information about licensing and content included in package



Licensing

This projects uses MIT LICENSE which you can find here.

Licenses of dependencies

This project uses dependencies; you can get their licenses with the following commands:

- activate your virtualenv: `conda activate m05-mp-decaillet`
- generate the list: `pip-licenses --with-authors --with-url --format=markdown > 3rd-party-licenses-list.md`
- generate the licenses: `pip-licenses --format=plain-vertical --with-license-file --output-file=3rd-party-licenses.txt`

Licenses of data

The projects uses two datasets; their licenses are here:

- housing
- wine

© Copyright 2023, Benjamin Décaillet, Valentin Décaillet.

Sources

M05: Open Science and Ethics (Dr. André Anjos, Flavio Tarsetti, Joël Dumoulin)

See our documentation for more information

Any questions?

1. Dr. Anjos, A., Tarsetti, F. & Dumoulin, J. (2023). M05: Open Science and Ethics [Teaching Course]. IDIAP Research Institute. Martigny, Switzerland.
2. "Question Mark Question Response" by OpenClipart-Vectors from Pixabay. Available at: https://pixabay.com/illustrations/question-mark-question-response-1020165/
3. Github pages by Décaillet V. & Décaillet B. at "https://master-ai-batch5.github.io/M05-mp-decaillet/index.html"