Crash-Course in Contributing to Open Source Projects

Andreas Müller

(and contributions from: Reshama Shaikh)

Gitter: https://gitter.im/scikit-learn/sprint

Crash-Course in Contributing to Open Source Projects

(with some specific instructions to sklearn)

(Great!) Reasons to Contribute to Open Source

- Giving back. Users of open source like to give back and participate in the community.
- Contributing to open source is an excellent *learning* opportunity which advances one's data science and Python coding skills.
- Fix bugs, add features & solve a problem that (we) developers faces in work.
- OS is movement, projects flourish with various contributors (FB: Pytorch, Microsoft: VSC)

(Great!) Reasons to Contribute to Open Source

- In DS more appealing to work with OS tools; the familiarity, allows job transition, increases value of tech skills because they are transferable, synergy of knowledge
- Interact with core developers, who are experts
- Job opportunities
- Fun

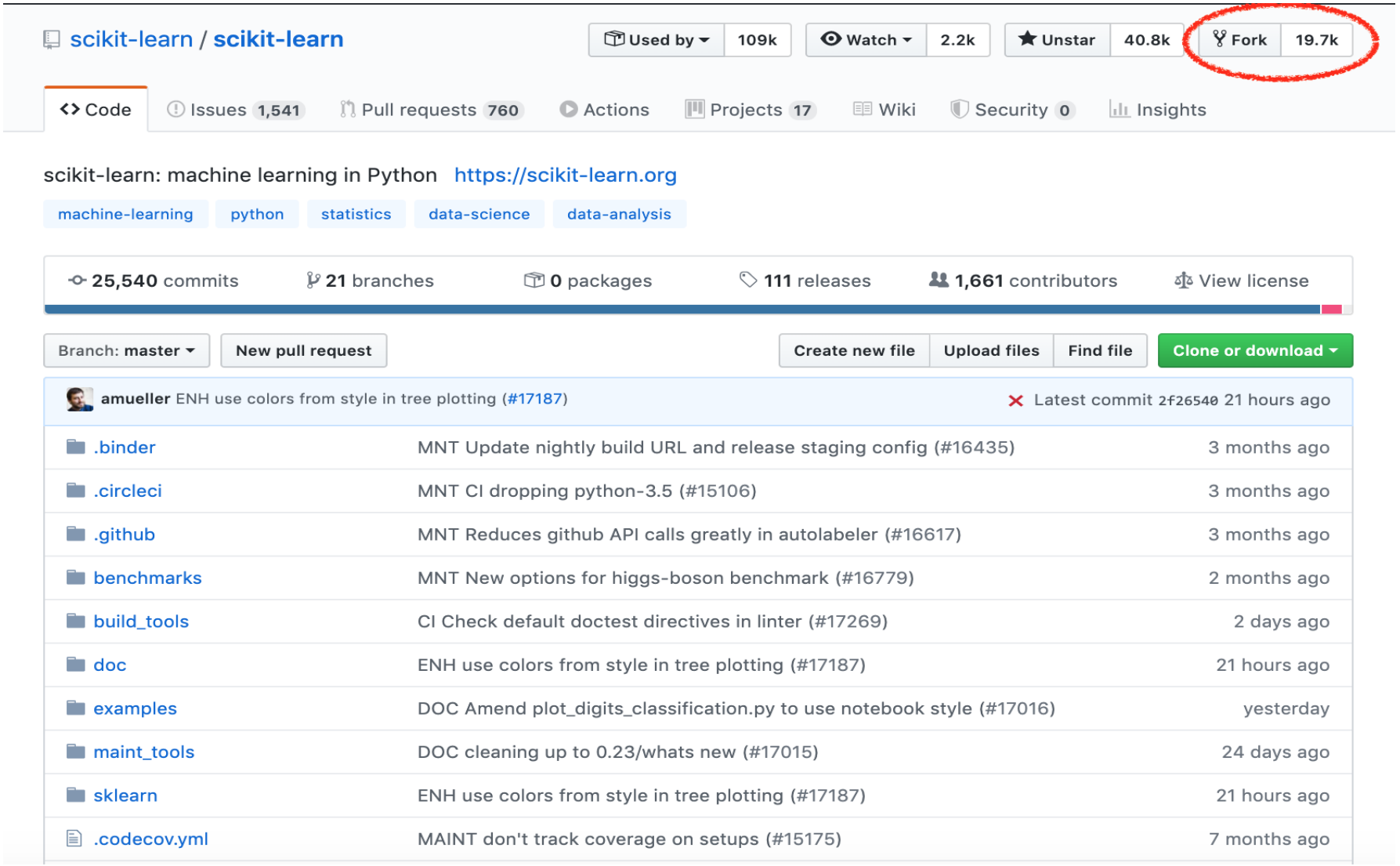
Environment

- If you don’t have a working python installation, install Anaconda.
- If you don’t want to install the scikit-learn development version in your main environment, create a separate environment (with conda or virtualenv):

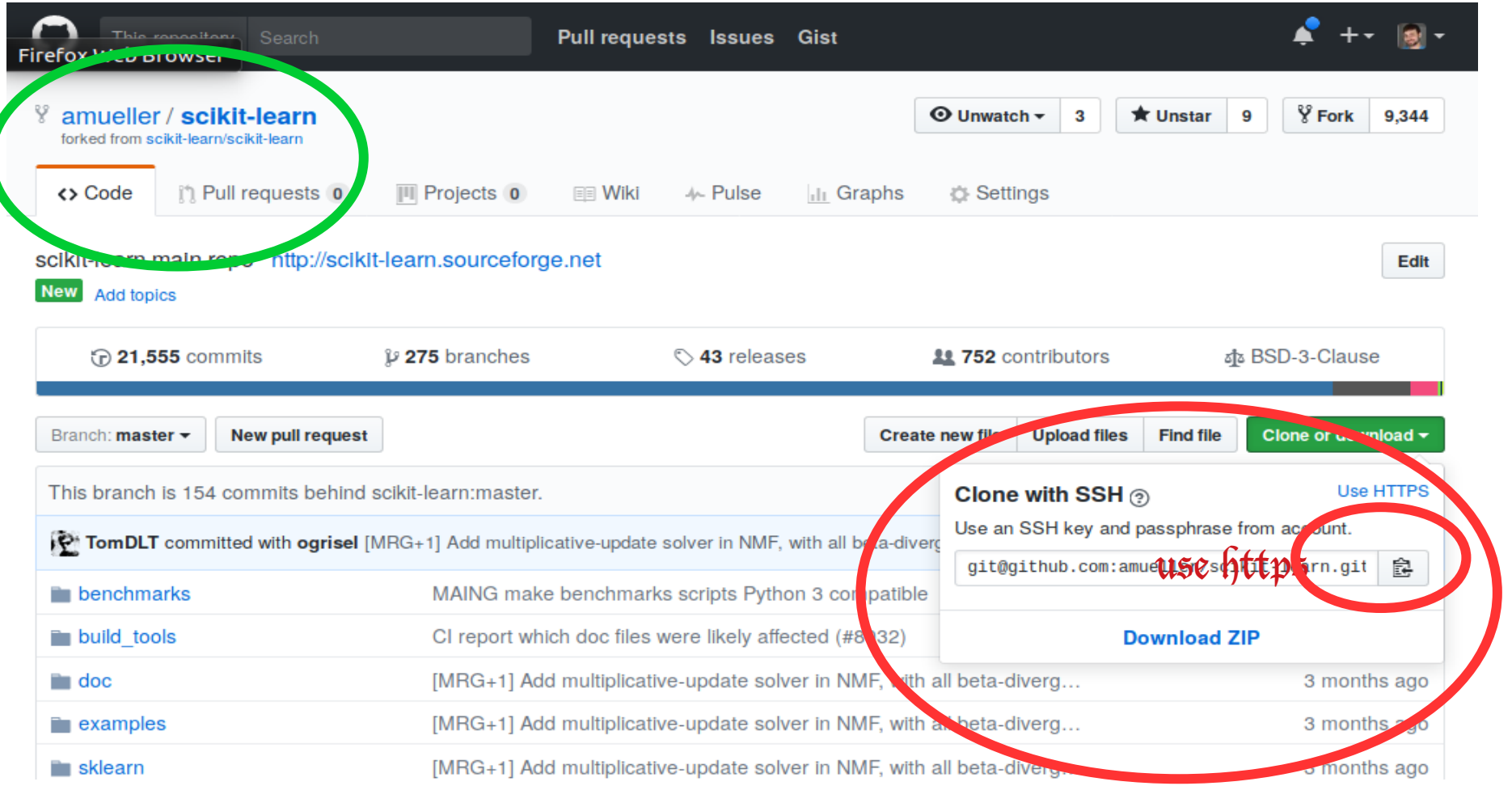
```
conda create -n sklearndev numpy scipy matplotlib pytest
sphinx cython ipykernel

source activate sklearndev

conda install -c conda-forge sphinx-gallery
```

Fork sklearn on github

https://github.com/scikit-learn/scikit-learn



Clone the Fork



If you don't have ssh keys set up, use https!

Add main scikit-learn repo as remote called "upstream":

```
git remote add upstream https://github.com/scikit-learn/scikit-learn.git
```

Build and run tests!

- Easiest way:

```
cd scikit-learn
pip install -e .
```

- Will overwrite existing installations!

Starting on Issues

- Go to:

  https://github.com/data-umbrella/2020-sklearn-sprint/projects/1

- Comment on issue saying: “I'm working on this.”
- Update local master branch from main sklearn repo

  ```
  git pull upstream master
  ```

- Fetching someone else’s PR:

  ```
  git fetch
  https://github.com/theirusername/reponame.git
  theirbranch:ourbranch
  ```

Starting on Issues

- Create feature branch

  `git checkout -b <branchname>`

- Commit changes to branch, run tests

  `pytest sklearn`

  (or individual test files)

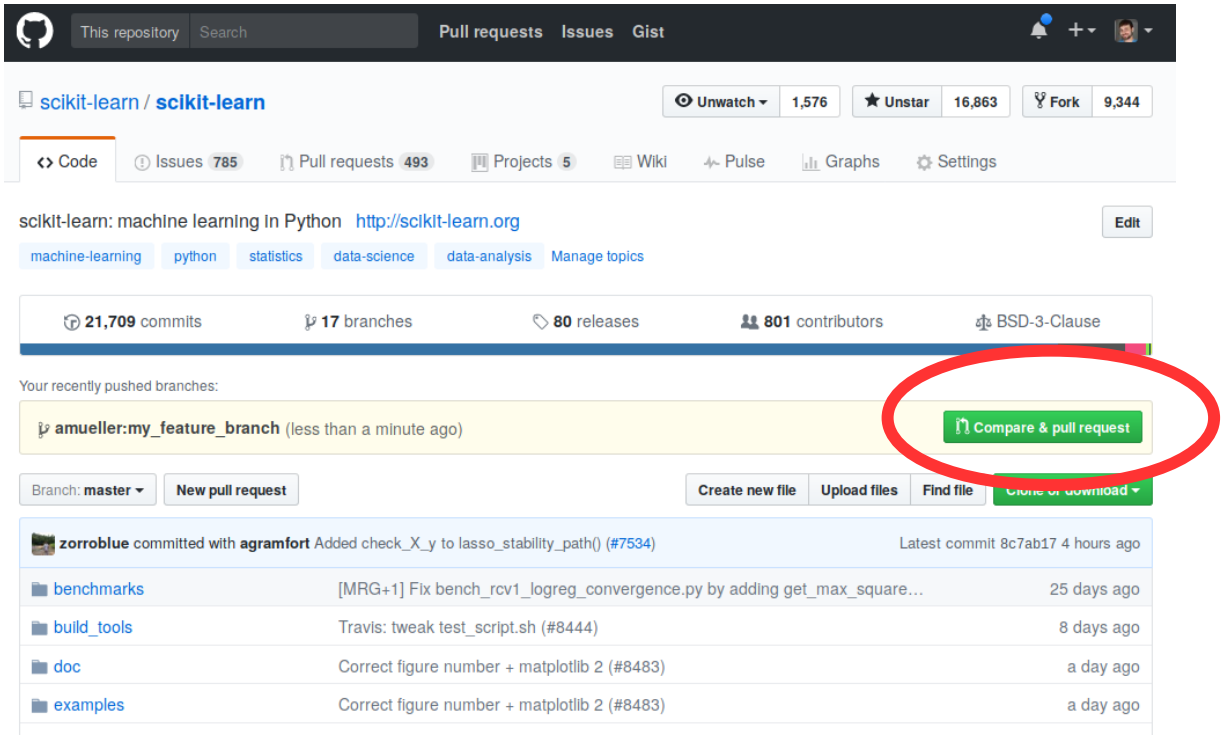
- Run `flake8` on changed files

Creating a PR

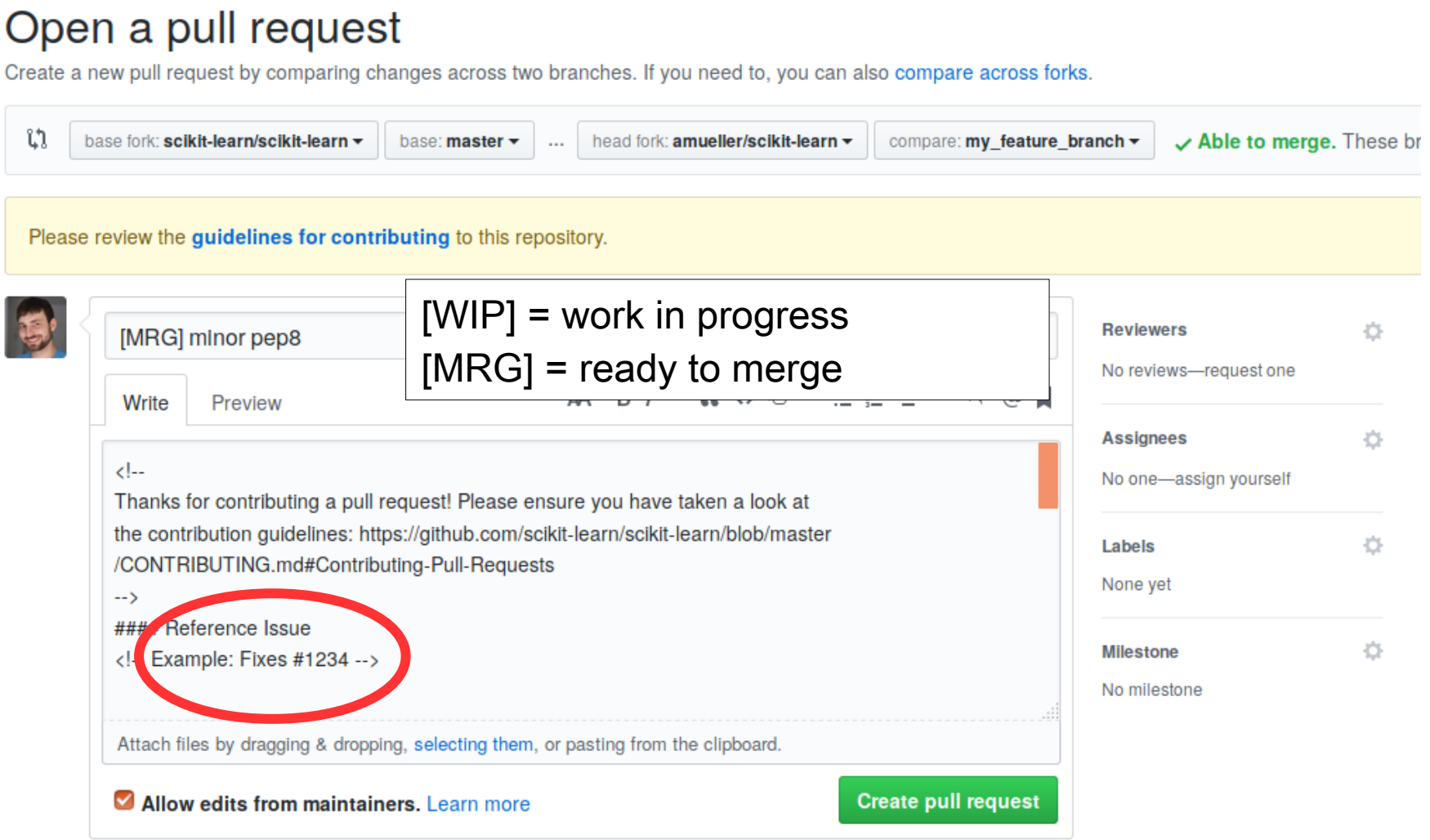
- Push changes to your fork

  `git push origin <branchname>`

- Create PR using GitHub UI:

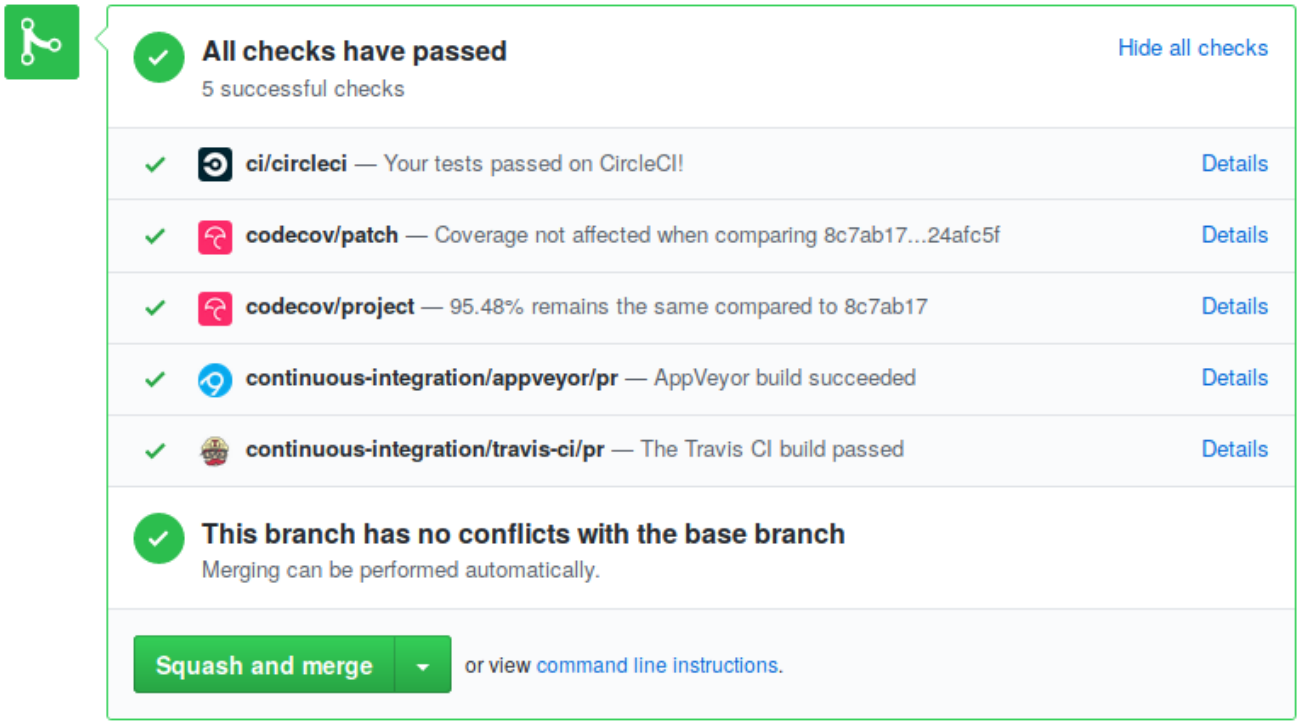


Describing PR



Regression tests

- Are mandatory! For everything (except documentation changes)!
- Make sure continuous integration passes:



What's next?

- Wait for reviews (be patient).
- Address review comments in the same branch.
- Pushing to your fork will update the PR.
- Reviewers will "approve" PR or change title to `[MRG + 1]`
- You need two approvals for a merge.

Finding Issues

- Check “need contributor”, “easy” and “sprint” issues.
- Something unclear in the docs? Fix it!
- Can’t fix something that’s unclear? Open an issue!
- Problem that you keep running into: Open an issue!
- Find stalled PRs (the author didn’t address reviews for ~>1 month) and continue them!

https://github.com/data-umbrella/2020-sklearn-sprint/projects/1

Reviewing

- You can review PRs and issues!
- Some bugs are not confirmed. See if you can confirm them and under what conditions?
- You can review documentation PRs for language and whether they are clear to you.
- You can review code changes on whether they address the issue (might be a bit tricky).
- Don’t be afraid to ask clarifying questions!

Workflow Reminders

- Why we do pair programming
- Include @mention of other pair programmer in PR submission
- Following up on work; expect a lot of back / forth, it's part of the process. It is worth the experience to see a PR through to the merge state!
- Watch your GitHub notifications (especially for folks who use an alternative email address that is not checked often, or it gets filtered into a folder)
- Policy of "abandoned / stalled PRs." Etiquette is to communicate status.

Final words

- Pick something TRIVIALLY SIMPLE as the first contribution.
- You can do the cool stuff afterwards!
- There might be interesting issues that are not appropriately tagged.

Thank you for your help!
Enjoy!