Bug triaging and issue curation

The issue tracker is important to the communication in the project: it helps developers identify major projects to work on, as well as to discuss priorities. For this reason, it is important to curate it, adding labels to issues and closing issues that are not necessary.

Working on issues to improve them

Improving issues increases their chances of being successfully resolved. Guidelines on submitting good issues can be found :ref:`here <filing_bugs>`. A third party can give useful feedback or even add comments on the issue. The following actions are typically useful:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\developers\[scikit-learn-main][doc][developers]bug_triaging.rst, line 15); *backlink***
>
> Unknown interpreted text role "ref".

- documenting issues that are missing elements to reproduce the problem such as code samples
- suggesting better use of code formatting
- suggesting to reformulate the title and description to make them more explicit about the problem to be solved
- linking to related issues or discussions while briefly describing how they are related, for instance "See also #xyz for a similar attempt at this" or "See also #xyz where the same thing happened in SomeEstimator" provides context and helps the discussion.

Fruitful discussions

Online discussions may be harder than it seems at first glance, in particular given that a person new to open-source may have a very different understanding of the process than a seasoned maintainer.

Overall, it is useful to stay positive and assume good will. The following article explores how to lead online discussions in the context of open source.

Working on PRs to help review

Reviewing code is also encouraged. Contributors and users are welcome to participate to the review process following our :ref:`review guidelines <code_review>`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\developers\[scikit-learn-main][doc][developers]bug_triaging.rst, line 49); *backlink***
>
> Unknown interpreted text role "ref".

Triaging operations for members of the core and contributor experience teams

In addition to the above, members of the core team and the contributor experience team can do the following important tasks:

- Update :ref:`labels for issues and PRs <issue_tracker_tags>`: see the list of the available github labels.

  > **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\developers\[scikit-learn-main][doc][developers]bug_triaging.rst, line 59); *backlink***
  >
  > Unknown interpreted text role "ref".

- :ref:`Determine if a PR must be relabeled as stalled <stalled_pull_request>` or needs help (this is typically very important in the context of sprints, where the risk is to create many unfinished PRs)

  > **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\developers\[scikit-learn-main][doc][developers]bug_triaging.rst, line 63); *backlink***
  >
  > Unknown interpreted text role "ref".

- Triage issues:
  - **close usage questions** and politely point the reporter to use Stack Overflow instead.
  - **close duplicate issues**, after checking that they are indeed duplicate. Ideally, the original submitter moves the discussion to the older, duplicate issue
  - **close issues that cannot be replicated**, after leaving time (at least a week) to add extra information

:ref:`Saved replies <saved_replies>` are useful to gain time and yet be welcoming and polite when triaging.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\developers\[scikit-learn-main][doc][developers]bug_triaging.rst, line 79); *backlink***
>
> Unknown interpreted text role "ref".

See the github description for roles in the organization.

Closing issues: a tough call

When uncertain on whether an issue should be closed or not, it is best to strive for consensus with the original poster, and possibly to seek relevant expertise. However, when the issue is a usage question, or when it has been considered as unclear for many years it should be closed.

A typical workflow for triaging issues

The following workflow [1] is a good way to approach issue triaging:

1. Thank the reporter for opening an issue

   The issue tracker is many people's first interaction with the scikit-learn project itself, beyond just using the library. As such, we want it to be a welcoming, pleasant experience.

2. Is this a usage question? If so close it with a polite message (:ref:`here is an example <saved_replies>`).

   > **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\developers\[scikit-learn-main][doc][developers]bug_triaging.rst, line 104); *backlink***
   >
   > Unknown interpreted text role "ref".

3. Is the necessary information provided?

   If crucial information (like the version of scikit-learn used), is missing feel free to ask for that and label the issue with "Needs info".

4. Is this a duplicate issue?

   We have many open issues. If a new issue seems to be a duplicate, point to the original issue. If it is a clear duplicate, or consensus is that it is redundant, close it. Make sure to still thank the reporter, and encourage them to chime in on the original issue, and perhaps try to fix it.

   If the new issue provides relevant information, such as a better or slightly different example, add it to the original issue as a comment or an edit to the original post.

5. Make sure that the title accurately reflects the issue. If you have the necessary permissions edit it yourself if it's not clear.

6. Is the issue minimal and reproducible?

   For bug reports, we ask that the reporter provide a minimal reproducible example. See this useful post by Matthew Rocklin for a good explanation. If the example is not reproducible, or if it's clearly not minimal, feel free to ask the reporter if they can provide and example or simplify the provided one. Do acknowledge that writing minimal reproducible examples is hard work. If the reporter is struggling, you can try to write one yourself.

   If a reproducible example is provided, but you see a simplification, add your simpler reproducible example.

7. Add the relevant labels, such as "Documentation" when the issue is about documentation, "Bug" if it is clearly a bug, "Enhancement" if it is an enhancement request, ...

   If the issue is clearly defined and the fix seems relatively straightforward, label the issue as "Good first issue".

   An additional useful step can be to tag the corresponding module e.g. *sklearn.linear_models* when relevant.

8. Remove the "Needs Triage" label from the issue if the label exists.

[1] Adapted from the pandas project maintainers guide