Javier Ron
Reflection on pynblint

What did you learn, in your individual session, about static analysis for ML and the pynblint tool?

I learned about the purpose and usage of pynblint. Luigi and I had a very interesting discussion about the usefulness of such a tool, and the adoption of linters and software engineering practices in general in the area of data science. I seldom use python notebooks, so I learned a lot about notebooks technicalities, best practices, and own challenges. With this new perspective, I understood the need for a specific static analysis tool for notebooks. Enforcing a code standard is not enough, notebooks provide several other features such as markdown formatting, and single-block execution, that need to be used properly to achieve good collaboration and maintainability. I also discussed with Luigi about pynblint in the context of automation and CI.

Will pynblint be useful to you in your WASP PhD project? Why or why not?

I am currently not using notebooks for my project. However I might use them in the future, and having a tool like pynblint integrated into my pipeline would be definitely useful. In addition, when sharing artifacts of any kind for reproduction of experiments, we want to make sure that these maintain a certain standard, this consideration also applies to notebooks. Often good development practices are overlooked, and having tools like pynblint to enforce them automatically is very helpful.

Ideas for how the tool could be improved?

As discussed with Luigi, sometimes developers would like to ignore/override a warning for a very specific reason. The linter should allow for this, a possible implementation would be to define an exception within a comment at the beginning of a block.

Automation would help teams to adopt a tool like pynblint with less friction. Integration with CI engines (GitHub actions, Jenkins, etc.) is a very good way to achieve this, and relatively not costly to develop.

What do you see as the limits for static analysis tools in ML? For code, models, and for data?

The obvious limit is program execution, this applies in the same degree to code and models. However, as shown by pynblint, it is not a limit that keeps static analysis rules from evolving and adapting to new domains.