Reflection on pynblint

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

An interesting takeaway from my discussion with Luigi was that, aside from their main purpose, the use of linters can also be educational and help the user understand (language-specific) best practices for software development. Especially for someone with not much of a software engineering background like myself, this would be an additional motivation to turn to tools like pynblint when evaluating my own code.

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

Although I have not been using notebooks much in my own research (aside from WASP course assignments), I do see their value when having to provide a rundown on how to best use some code, a proof of concept or demo, or some idea I would want to share with colleagues. Mainly because of the fact you have executable code cells together with Markdown-styled explanations in a single (hopefully) neatly-organized file that does not have to necessarily be run all at once or in order. I would think, especially given my limited software engineering background, a tool like pynblint will help in making sure everything checks out or point to the bits and pieces that do not and should thus be improved, and as a result aid in making the notebook easier to follow for those I would be sharing it with.

Ideas for how the tool could be improved?

Although it depends on where the notebook is executed, indentation issues may lead to errors/complaints, i.e., inconsistent use of tabs and spaces; this seems like a thing a linter like pynblint should be capable of detecting and precisely specifying where the mixing occurs. I could also see it being useful to have pynblint as an extension for, for example, Visual Studio Code.

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

Something that also came up a couple of times during my discussion with Luigi was that things that some would consider errors or otherwise not quite best practice may depend on the use case of that which is being analyzed. I could see it become difficult to create static analysis tools that can deal with codebases that have tons of dependencies. In the ML space, there are so many frameworks, different tasks and ways of handling data for said tasks that it becomes hard to standardize.