Assignment 1 – pynblint

My session with Luigi started off with a general discussion about how I use notebooks and my experience with linting. I had already prepared a diverse set of notebooks from my research, of varying quality, and of varying purposes. One that was only used for the initial exploration of an idea and one that constituted the main code for a paper.

Overall I had quite little experience with notebook linters before the session, and before the course I had put very little thought into the importance of well-structured notebooks. The main reason for this is that I mostly use notebooks for initial experiments, where I work with a notebook for some days and then never open it again, so the lack of structure does not affect me too much. The session convinced me that this is something I have to improve on. After running the tool on my notebooks and refactoring them to remove all problems, the clarity of the notebook improved immensely.

In most of my research I do not use notebooks or ML so overall it will likely not be too useful to me. However, for appropriate projects I think it has its uses, but this depends on how well the tool actually performs on a wider set of notebooks. As with all linting tools, the usefulness of a positive detection and the false positive rate are key. Additionally, for me to use a linter it has to be tightly integrated in my editor of choice, such that I do not have to explicitly run it, which is currently not the case for pynblint.

Improvements

Me and Luigi discussed several ideas of how to improve the tool. The most important one is the one discussed above, i.e. the integration with the editor. In addition, we also discussed possible integration with more common code style linters, as it can be cumbersome to run several linters on your code.

Limits

Overall I like the idea of static analysis of notebooks for ML. However, the main limitation I see comes from the fact that notebooks are commonly not used in the same way as other code. As they are typically used for individual initial experiments, are often executed in a non-sequential fashion, and are cumbersome to use with conventional coding practices, I think the benefit of static analysis is diminished. Currently, I would also argue, that "good style" in notebooks hasn't converged to the same stable level as in regular structured code. This will naturally limit what a linter should test as perceived false positives are the bane of every linting tool.