WASP Software Engineering and Cloud Computing Module 2, Assignment 1

Leonard Papenmeier
(leonard.papenmeier@cs.lth.se)

I have a bachelor degree in software engineering and worked in a software development company during that time. Therefore, I have some experience with static code analysis but did not really use it since then. Personally, I think that linters and style guides are crucial in projects with multiple developers. Here, pynblinter is no exception. Most features are most useful in cases where a notebook is shared. For example, the analysis whether a notebook has markdown cells at the top or every couple of cells is useful in cases where others have to read a notebook. In cases where a notebook is intended for personal use, this feature might be not be as important even though it still encourages good style.

A quite unique feature seems to be the check whether cells were executed in order. This, apart from the Python syntax check, is maybe the most useful feature as running cells in the wrong order can lead to severe problems. As an addition, a check whether the first cell has cell count "1" could be useful as this could avoid cases where the code still depends on variables that have been removed in the meantime.

Generally, I think that pynblinter encourages good style in notebooks (e.g., imports at the top of the notebook, sufficient documentation, naming of notebooks). I think that it can be useful for my work in cases where I want to share notebooks with others. However, I am not sure how often I will actually use it because I find it a bit too hard to use. In general, it would be great if pynblinter would be implemented as a plugin for Jupyter lab. In cases where I want to manage notebooks in git repositories, it could also be used with a pre-commit hook but if I just share the .ipynb file, I might not actually run it via the command line all the time.

I was hesitant in managing notebooks in repositories because just executing a cell again already can the .ipynb file by a lot which leads to hard-to-read commit diffs. However, Luigi recommended me to remove the output before checking in a notebook file which seems to be a great idea.

I do not have a lot of experience with static code analysis for machine learning, however, I would suspect that one difficulty is that many errors are conceptual (e.g., categorical variables are represented as ordinal variables). This requires a lot of rules to spot such errors that are framework-dependent. Furthermore, such rules would need to change quickly as ML is a field that develops continuously.