Cloud Computing & Software Engineering

Pynblint

Umeå University - Joint Curriculum

Sabine Houy

I will start by giving some background information about my research and my experiences with notebooks to provide a better understanding of my impressions. I come from the field of cyber security and have only worked once or twice with a notebook (jupyter notebook) due to course assignments. These were also the only times I ever worked with machine learning. Since this lack of experience, the Pynblint session allowed me to learn more about the use of jupyter notebooks in general.

The Pynblint tool gives an overview of the "*look*" of the notebook considering programming best practices. This includes checking whether the imports are done in the first code cell as it is always recommended to do the imports at the beginning, not only in notebooks but in general in programming. When following Pynblint's suggestions, the notebook will be easier to use for others as one suggestion is to use meaningful names and have a markdown cell right at the beginning describing the notebook's purpose. However, my main takeaways from this session were not about the introduced tool but about how to use notebooks better, which, from my understanding, is one of the purposes of Pynblint. Since I am not working with data, I did not know that using different version management tools is recommended than git, especially for large data sets. The few times I have used notebooks, the used datasets were so small that we needed no management tools. Pynblint also points out that the developer of a notebook should consider the top to bottom approach for reproducibility. This means that the results or outputs of a notebook can look differently depending on the order in which code cells are run. I have experienced that myself but never gave it a second thought that it affects reproducibility. Therefore, before uploading or publishing a notebook or results depending on a notebook, it should be run once from top to bottom to ensure that everyone can reproduce these results. My last takeaway was how notebooks can be split, which is mentioned in the context of having too long notebooks, e.g., not many cells. There are two possible approaches. The first one is to use a pipeline setup of multiple notebooks, meaning that the output of one notebook is used as the input for another one. The second one is to outsource code parts that can then be imported as external modules. The explanation of how to split notebooks was given to me during the Q&A session after using the tool. Adding these two approaches to Pynblint could be an excellent way to improve it.

Unfortunately, Pynblint will not be helpful in my WASP Ph.D. project since I am not using notebooks or machine learning in general. My research project focuses on Android root exploits, which means that I analyze malicious apps that can gain root access, i.e., almost complete control, of the infected smartphone. My project aims to find ways to facilitate the analysis of such applications to improve detection capabilities.

From my point of knowledge, the limitations and challenges of static analysis of machine learning concern, in particular, the use of huge datasets and the *black-box* nature of ML. Huge datasets can slow down static analysis and make it inefficient as too much data needs to be processed. Machine learning is generally considered a *black box*, which means it is possible to observe the input and the output but not what is going on inside, so how exactly the model produces the results based on the given data. This makes static analysis in that sense basically impossible. However, it is possible to analyze the code itself, e.g., based on programming best practices such as Pynblint. So the "*look*" can be analyzed but not what is actually going on, for that dynamic analysis is needed.