Q1. Could you describe your process for developing an ML system?

"There is no simple, clear solution, but there are already a few rules of thumb, which somehow works quite well for which problems, and if it is somehow table data, which is also somehow still common in the companies, it is just so tree-based somehow such a booster never wrong. The process is actually very explorative and I don't think it's that unusual to first boot up a notebook and try things out a bit. And with me, because I always have to force myself to work a bit faster from the beginning then properly, so that I then relatively quickly from the notebook then also the first modules out again. I then work with VSCode. I always have a point where a notebook becomes too confusing for me. And that's always the jumping off point, so as soon as I have untitled2 or so as a notebook, it's time to just switch and pack the whole thing into modules and then project setup is actually always to make a Python package out of it if you're in the Python system, which we usually are somehow, because then with Poetry, Python package and then utils then first refactor, then the function names, which I've named poorly at the beginning. Once go over, rebuild and then yes then just depending on what just also, how we want to bring then at the end also the model in production, so is yes different whether I have a batch, or whether I want to use that somehow streaming or API moderately, my model then and that determines then just also the procedure as it is built. We do not always make a ML pipeline So it depends on the project stage, so sometimes it is already clear that you somehow just improve the model and you already know or know or can well estimate that it brings value. Otherwise, we also often have somehow so with machine learning problems, somehow again a new uncertainty somehow what we do not have in data engineering now, then we actually know what comes out then in the end but so show this value? Therefore, depending on whether it is the first model, you can also go faster, there is then show value in focus, there is then also no somehow automatic, somehow shadow deployments or so what would then somehow allow. Testing the model against others is then the first, the second throw, then yes the one with which you also simply show the value and the processes then come downstream when the value is provided. Often the notebook is then thrown away. However throw away now also not so so it is already still comprehensible for everyone, but often it is just redundant, because the function then simply already neatly in the utils are stored and there then also simply to be pulled. I usually have an exploration folder in the whole structure, where they are also left in, but even that is now cleaned up more by me, also simply in view of the fact that someone else goes to it and at some point comes the point where someone else goes to it, even if it's just me. Hm, so it is yes it depends on how you also use it in productions. So then we will just have the model artifact. Then in the vast majority of production cases we also have a model registry or we have it in some file storage from which we then somehow fetch it. It also kind of depends on the company, how the processes and the teams are structured, right? So I would see it as context-dependent. I think you can parallelize a lot. However, it is often the case that a data scientist makes an impact without these very classic structures, as was somehow more common 2 or 3 years ago, where the data scientist in the ivory tower first went out with the model at some point.

That's exactly why I think there's much more parallelization nowadays than in the past, where the model development was first in the foreground, even if today you can already better assess where a problem can really be solved with ML."

Q2. Do you continue experiments after deploying the ML system? If so, how do you go about doing this?

Yes, whereby so soon I believe simply so the application stands, is the way to the notebook back rather rarely with me so then I would rather integrate in the functions and then with

tests, which I would then still supplement, instead of there then. Because so much explorative happens then mostly no longer, if one has somehow the first problem version. Sure, new features are engineered, but on the whole, the structure is there. We only have relatively short cycles and also clearly defined problems that we just work off and then yes so not much time passes between the iterations. The time spans in the development were never so long that the environments could have diverged insanely, but rather with the next iteration they just moved on directly and then it starts all over again. That was a process that everyone stuck to.

Q3. Which tools do you use?

already answered in Q1. [Interviewer asks if plugins are used] Only the Black Formatter.

Q4. What are some common challenges you face in this process form a tooling perspective?

[Skipped due to time constraints]

Q5. Which issues do you encounter when working with Jupyter Notebooks?

I think that's also a bit due to my own untidiness, so I think I could have collected more things over time, so to speak, and that while prototyping, they already have some of my building blocks from the past ready, somehow easier access to what I've already done. Then collaboration. That's a good point, I think the one that's still missing, somehow in the development, that you can work simultaneously.