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

But yes, so we definitely take the data that we have and check whether this data is hopefully available, see what is the task at all? So we have labels, which is typically not the case. Yes, if that's the case, you take a very rough look at how big they are, just load small things into the notebook. For larger things, you might have to think about reducing them to a small data set and then looking at the data set to see which columns I have data, types, nans? Yes, just plot a few distributions typically. Can the profiling maybe run over it, if you want to have on the quick just somehow diagrams or paint yourself a few first diagrams, depending on how much time I see for myself also, I would build me maybe also somehow so a small mini dashboard to be able to go through the things better. Is now of course the question what kind of model we have there? Yes, throw the typical approaches on it, if it's a classification and regression probably just throw a grading boosting or regressor on it, if it's clustering you just have to see how big it is and so on if you somehow reduce some dimensionality, run Youmap for example on it and then KMeans or whatever you just have to see what kind of problem it is exactly. If it is supervised and I have a bit of time as I said and want to run several things, then I would just connect to MLFlow and log the parameters and metrics. Depending on how much time is then and how successful that is, I would perhaps also throw one or the other AutoML tool on it, in order not to model too much myself, but to take my base line with it and see how that works. Typically, you discover relatively quickly any flaws that you look at features and then you see ok actually only this and only that feature is relevant and then in reality is just mostly that you then still have to talk to the customer, because no idea the important data is not there or you wonder, there is still something missing or somehow something. Then compare metrics until my model has what is executable in a reasonable time. Theoretically you could do stacking, build ensembles, but factually you don't really do that. And depending on what the desire is, run that already once or if you want to have a service, either build something by hand with FastAPI or similar Python tool or also use ML serving directly. Yeah, and then models could have given the predictions or given clusters back or something. Well, if you still have such a pipeline, I would actually relatively, so factually it is yes ne time question. You actually always have a certain time budget, it's never infinite. So you just look ok how much time do I have approximately and then look that you build with a little buffer before, in any case first the service, because you can, as long as you do not change now later completely everything somehow, you can exchange the model relatively easily. That means you first build a serving layer. And if there is still time at the end, then you just continue to build the model until you arrive at one that is optimized according to the metrics that you want to optimize, whether that is purely MSE or whether you then also maybe pay attention to something, I don't know, maybe you also have to pay attention to fairness or something, I've never actually had that, but it could also be or yes, that you then also pay attention to these metrics. In the end, you choose the model that works best in the metrics and that can be executed in a reasonable time. It always depends on the requirements of the customer. Exactly, so I would definitely not build the serving pipeline right to the end, but I mean, if the model can't deliver any forecasts afterwards, then I would actually try relatively early to build the pipeline through once at the beginning, so that it's already in place, and then rather exchange the model in there bit by bit. I start with the notebook, develop my first methods, then put the whole thing into PyCharm, typically build a class in there, develop it and execute it from the notebook for a while, so that in the end I only execute the train function and the predict function or something like that and then what I do is I just replace my notebook with a script that just calls everything once and maybe put it all into a Docker container frequently and then execute that. Exactly

so I developed so the first little approaches in the beginning, and as soon as a function, let's say, from the gut feeling has more than 10 lines or I somehow need a class or something, then I take that, pack the whole thing once over into PyCharm and make my own package with it. So copy-paste into the package and then quasi in Jupiter I then always call the auto-reload magic, if you know it and then you have quasi the import of your package and then you just have your class and can change it in PyCharm, but the changes immediately in the notebook there, that's my workflow. When developing I like to work in the notebook, because you have everything right there and trigger things from there but of course, if it really should go into production, I don't think it's good to do things in the notebook. So I would just throw those 2, 3 lines, hopefully that's it, define some input variables, call my training class and maybe save the return value or something, into a script and put that into a Docker container. We have the notebooks and the packages in the same repo, plus there's a Docker file to build that, a Docker compose typically to just run the thing reproducibly and typically either ne in environment.yml for Conda or a pyproject file for Poetry.

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

In the notebook there is not so much logic in it, so in the notebook I can take the return values and then I get a DataFrame back and wonder ok why is this line like this? And then I look in there and so that's my interactive shell more or less also just a bit more powerful interactive shell. The actual code goes into the Python class so that it's also version controlled. I typically don't check in the notebook at all or only at the very end. Mostly is then but rather copy-paste and 5 minutes of work, but yes so. I would not take that as a large amount of work.

Q3. Which tools do you use?

"So exactly tools: Jupiter Notebook, PyCharm, tracking of experiments with MLFlow typically around environment variable management with Conda or Poetry typically. Typically pre-commit configs with Code Black for codestyling, iSort to structure imports and so on, so mainly if you're working with someone, then mainly so you don't put stupid changes in each other, but on your own you can sometimes be less than consistent. Typically I don't set everything to new every time, but with Cookie-Cutter. For me PyScaffold is too big, it's too heavy, it gives me too much stuff I don't want and puts a lot of files in my repo that I don't need. I prefer to have this very custom mostly tailored to me. [Interviewer asks if Jupyter Notebook plugins are used] I definitely have a number of plugins. So as I said the Auto-Reload Magic is definitely important to me, other than that plugins. What do I have installed? Yes, a few plugins for plotting, plotly express and something that works interactively.
JupyText and some sublime extension to have automatic keyboard shortcuts in it. Rendering Service especially so a few plugins just for plotting no, that's just the plots are neatly displayed but nothing what extreme"

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

So you're in your own bubble and you don't see the big ones now. What I immediately think of are various little things that bother me about individual tools, about MLflow, some bugs and things like that where I say that things are not as mature as I would like them to be, where a few things are missing or smaller things don't work. I think it's important to consider where you still lose a lot of time, that is, with what you spend a lot of time and typically that is not in dealing with the tools that you know, but then again to understand the data

correctly and understand why your model now works so or does not work. Data quality is often said to be an issue, there are, as I said, a few nice tools that try to take something off, but that's just not so that they tell me now exactly you have in the data set the lines that are all funny or something else. A second issue is, which is also an issue in any case, at least then my tools stack is perhaps not optimal for, typically you start with something small like Pandas and then you have to change to the Dask and then you have to rewrite just somehow felt half functions, because that is then somehow different and is not so completely compatible. Yes, scaling is, if you build it that way from the beginning, it's all ok, but typically requirements evolve during the project. I have my tools locally, I like to work locally, I don't like to work directly in the cloud, because I have somehow adapted everything better to me locally. And now you have the whole thing, because the big data set you pulled sample at the beginning and it all worked nicely and now you want to run everything and that just doesn't work at all with. Time just doesn't run forever and then I haven't found the ultimate solution for me yet, so what I then quite typically actually often do is I just go into the cloud, I start my notebook instance just like locally on, scales the nice big with the GPU and clone my repo and also set up everything there again so a bit. And then make a few runs there and develop locally again, but there I have to push locally again and again and then pullen there again and somehow that's not so nice. I would prefer to have the memory and the CPU and the GPU that I can simply get in the cloud here locally in my environment. So here's this shift from I have it locally and then I have to get it in the cloud because I don't have the performance locally. That's still not well solved for me so I don't have a good solution there yet. Yes, otherwise, as I said, AutoML is a topic where I'm not yet as fit as I would like to be, so I think there's a lot going on, but of course it's also very computing-intensive, then you probably have to quickly come back to another problem.

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

Scalability: I don't see why this is a problem in the notebook versus elsewhere. In principle it can support multiple CPus or something like that. Artifact management is done by MLFlow for me, so experiment tracking in MLFlow. Cleaning refactoring is irrelevant for me, because I build some nonsense in the notebook and then I transfer it into my PyCharm and that's the point where I clean up and refactor. As I said, I also just program the executable in the notebook or just fast stuff and then go somewhere else. What I'm missing then is actually a debug. Typically I have to pull the few command lines from the notebook over into a script and then execute it in PyCharm or have a per license and be able to execute the notebooks there. Then you can set debug points, that would be just still horny. Yes, static analysis in the notebook would be nice too. Code that is in the notebook I don't want to test, so as soon as I would want to start testing it, I would definitely put it in the package and then I can test yes.