Integrating data science and ML engineering

How to design the **handoff** process between data scientists and ML engineers

Challenge: Different *needs* of data science vs ML engineering

- In both we want agility - but:
- The way this is achieved in data science is by the *explorative and iterative notebook* workflow
- By contrast, for a production software system to stay agile, we require *moving away from notebooks, clean code, type safety, and loose coupling between components*.
- The key challenge is that **the former does not automatically scale into the latter**; thus, *what leads to agility in model training leads to a lack of agility in model deployment* (and even just maintenance, if we consider data pipelines).

-> Need to find a good process for handoff!

Handoff between data science and ML

- Need to acknowledge:
  - unique *needs* of both sides (previous slide)
  - unique *strengths and capabilities* of both sides (division of labor)
- Need to find a handoff process that works for both sides

- Starting point: Codify *separate sets of quality standards* for the data science and ML side
  - Important to codify standards anyway

Handoff

- Productionizing ML ***models*** (easier):
    - *Model registry* can serve as a convenient hand-off point
    - Potential problems:
        - Changes to model interface (incl. schema of input data)
        - Dependency management
    - Solution: *Formal contracts* that are *automatically enforceable*
        - Standard process for environment creation
        - Better yet: Use containers
        - Define model interface and data schemas in shared libraries
            - Class interface can be cheaply enforced through static analysis (mypy) in CICD pipeline (and IDE plugin)
            - Data schema checks sometimes require executing code for validation (make this part of acceptance tests)

Handoff

- Productionizing <u>*code*</u> (hard):
  - 1. Make it as easy as possible for data scientists to follow software development best practices
    - e.g., provide templates for recommended IDE configuration, etc.
    - ...but it's not realistic to expect data scientists to become engineers.
    - ML Engineers should make themselves available to help (and coach) with things in their area of expertise. And vice versa.
  - 2. Apply automatic code formatting (AutoPEP-8, Black, YAPF)
  - 3. Manual re-factoring by ML engineers
  - 4. Find production-ready solution for any hacks
    - E.g., unofficial data sources need to be productionized

-> Need to balance quality control with ability of DS team to self-serve

Handoff

- Productionizing ***data*** *(hard)*
  - Productionize data **transform code**?
  - Additional challenges: Productionizing *data* **requires wider support from leadership** due to upstream dependencies
    - Assign *data owners* who are domain experts
    - Collaborate with data owners to fix any data problems that data scientists discovered at the source
    - Ideally, the *general* data engineering ("silver tables") is handled by dedicated teams/data engineers.
    - In the short term, ML engineers may have to lend a hand in order to show the value of this model (and because they have an interest in it).
    - In the long term, ML engineers should only be productionizing data transformations that are related to feature engineering or are very specific to their use case ("gold tables")

Handoff

- Create **cross-functional teams** of data scientists and ML engineers?
- Handoff process of code, models, or data *between teams* works best if we have a **stable, formal contract defined and ideally enforced in code.**
  - e.g., data schemas, API schema, gives interfaces/data classes defined in shared libraries, Gherkin scenarios
  - …because this decouples teams from each other, thus reducing blockers and communication inefficiencies
  - Requires an *engineering* mindset