Reaktor

MLOps training case themes

2023 Reaktor / Saku Suuriniemi

Case 1: A disaster in a small company

Essentially: documentation and accessibility. The small size of the company discourages investment, while dramatically increases the risks. To control the damage, the following would help

- Careful customer requirement documentation
- Data versioned in a backed-up storage with accident-proof owner transition plan
- Data catalog or some lighter version
  - Where to find what?
  - Rights to use
  - Semantics
  - Train-validate-test-split
- Model card, experiment tracking
- Code with README, requirements and informative commit messages.

Reaktor

IML4E

Case 2: The newcomer

Communication of objectives and automation to tenderly encourage good practises

1. Quality: Nightly automatic analysis run
    a. jupyter nbconvert or jupytext
    b. Make code quality visible (a linter, e.g. radon / pylama), pinpoint bad spots
    c. Gamify code quality?
2. Versioning
3. Review process (with quality prerequisite?)
4. Easier derivative work (Metaflow, templates for notebook text, model cards)
5. Repeatability: Help with e.g. pipenv to pin deps versions

Reaktor

IML4E

Case 3: The world changes

Expectations, mostly:

- Document the data schemas we work on
- Automated scheduled test reloads (Still on line? Changed data?)
- Regular queries of API Info objects

Reliability & contingency:

- Established primary sources
- Second sources

Pipelines that get triggered on data changes

- Updates of models
- Notifications for manual update needs

Reaktor

IML4E

Case 4: Let’s always synthesize!

Data catalog and feature store for

- feature discovery,
- explainability, and
- skew avoidance.

Reaktor

IML4E

Case 5: Learning online

Cold starts

- Start with an average profile for new customers
- Use genre for new titles, combine with ML predictions

Failed recommendations

- Monitor performance
- Automatic re-training, thresholds on average perf in multiple facets

ML blackouts

- E.g. raw pre-sorted likes in each genre to somehow sensibly bridge the gaps
- Plain popularity for absolute disasters (no ML & no customer data)

Data poisoning

- Outlier/poisoning/conflict analysis for training on low-stakes features (“like”s)

Reaktor

IML4E

Case 6: Can’t know for sure if it’s still working

Acquisition & monitoring

- Questions/questionnaires/user feedback surveys
- Reminders to keep the profile up to date
  - Targeted discounts to incentivize?

Re-training on

- Drop in user engagement
- User data distribution shifts
  - E.g. changes in regional hobby popularities
- Data from sport and culture organizations

Reaktor

IML4E

Case 7: Block by block

Composite predictors are awful. The specific main challenges are

- the ground truth for each block (or lack thereof)
  - If GT available, the blocks are individually trainable to spec and monitorable
  - else end-to-end training for all the irreducible composites that the block resides in
  - Prediction IDs and dependency graph for predictions, input-output logging
- the composite performance
  - Bounds for error and accuracy indicator accumulation (TP rates, …)
    - Derive pessimistic (often unrealistic) bounds for performance requirements
  - Limit interdependence: no GT => split the block to several that each serve a prediction path

CI/CD

- Test to detect great shifts in input and output distributions
- Test all relevant paths end-to-end for a changed block
- Good ML metadata associated to artefacts

Reaktor

IML4E

Case 8: Can you show it?

A practical dashboard set-up exercise. Free choice of tool (Grafana …).

- What aspects of the service status should be indicated?
- How?
- How to reasonably assess the aspects?

Reaktor

IML4E

Case 9: Are we still in business?

It is important to quickly test and deploy models that are properly understood.

- Data and model versioning and documentation
  - Especially model assumptions & data dependencies
  - Experiment tracking
  - Versioning system to explicitly reflect different situations?
  - Model performance indicators for data from all situations
  - Lime and/or Shaps to quickly assess the impact of changes.
- Quick deployments
  - Training artefacts ready, versioned and available
  - Parallel deployment to choose the best candidate with A/B/(C etc.) testing
    - Automation to support this

Reaktor

IML4E

Case 10: Freeze!

Ideas:

- Separation of the certified core code form corner case handling.
  - Core code modifications minimised
- Good documentation on what the certified domain is
  - CI tests to show that it is not transgressed
- A substantial amount of input-output data from the certified input domain plus automated regression tests to prove that the core has not changed.
  - Proof for fairly simple, low-dimensional models
  - Only circumstantial support for complicated, high-dimensional models.
  - Shaps to complement?
- If corner case handling requires ML, introduce a separate model

Reaktor

IML4E

Case 11: Bad habits

Communication

- Explain the ensuing problems
  - Concrete examples that have happened in the past
- There are tools that automatically quantify the code quality
  - Offer help for set-up, frequent runs to keep corrections manageable
- Criteria for good commit messages
- Well-named branches and **PR reviews** (with a genuine reject possibility)
  - Required human-readable READMEs or other documentation, from templates

Tooling

- Linting with strict-ish policies with good back-communication
  - Required docstrings, maybe also strict typing for the code
  - Indicate “dead code”
  - To be run **before** commits
- Soft limit single-commit size?

Reaktor

IML4E

Case 12: I’ve seen this before

Examples:

- Priority:
  - Deploy a model that is known to work (from live tests) or can be presumed to work
  - Fix the duplicate problem to enable training of non-compromised models
  - Preclusion measures
- Present model
  - Ditch it and train a definitely non-compromised model as soon as you can
- Debug
  - Split to inspections of
    - acquisition for bugs that cause repetition
    - parquet file creation (duplicate content, same files with different names…) and
    - subsequent pipeline
  - Run inspections in parallel, the author involved only as requested
- Preclusion
  - If sample identifiers are available, check them for duplicates in wrong places. Introduce sample IDs if not.
  - Train-validation-test-split at an early stage, descriptive names
  - Data versioning, preferably related to code versioning (helps mishap timing and recovery)

Reaktor

IML4E

Case 13: You just don’t understand

Likely problems, from too little communication overall:

- DS: perfectionist attitude towards a secondary feature?
- PO: not much experience of machine learning projects?
- The training data set acquisition failed.
- Experimental nature of ML misunderstood (also poorly explained?).

Mitigations:

- Discuss to forge common understanding of the objectives for the feature.
- Take data acquisition seriously and let DS or DE have a say.
- Task wording: “Study strategy X to make a go/no go decision on it.”
- Concretely worded grounds to support the go/no gos.

Reaktor

IML4E