Catch Me If You Can: Keeping up with ML Models in Production

Shreya Shankar

June 2021

Introductions

- BS and MS from Stanford Computer Science
- First machine learning engineer at Viaduct, an applied ML startup
  - Working with terabytes of time series data
  - Building infrastructure for large-scale machine learning and data analytics
  - Responsibilities spanning recruiting, SWE, ML, product, and more

Academic and industry ML are different

- Most academic ML efforts are focused on training techniques
  - Fairness
  - Generalizability to test sets with unknown distributions
  - Security
- Industry places emphasis on Agile working style
- In industry, we want to train few models but do *lots* of inference
- What happens beyond the validation or test set?

The depressing truth about ML IRL

- 87% of data science projects don't make it to production[1]
- Data in the "real world" is not necessarily clean and balanced, like canonical benchmark datasets (ex: ImageNet)
- Data in the "real world" is always changing
- Showing high performance on a fixed train and validation set $\neq$ consistent high performance when that model is deployed

[1] https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/

Today's talk

- *Not* about how to improve model performance on a validation or test set
- *Not* about feature engineering, machine learning algorithms, or data science
- Discussing:
  - Case study of challenges faced post-deployment of models
  - Demo'ing mltrace, a coarse-grained lineage & tracing tool

Challenge: lag

- In practice there are many types of lag
  - Feature lag: our system only learns about raw data after it has been produced
  - Label lag: our system only learns of the label after the ride has occurred
- The evaluation metric will inherently be lagging
- We might not be able to train on the most recent data

Challenge: distribution shift

- Data "drifts" over time, and models will need to be retrained to reflect such drift
- Open question: how often do you retrain a model?
  - Retraining adds complexity to the overall system (more artifacts to version and keep track of)
  - Retraining can be expensive in terms of compute
  - Retraining can take time and energy from people
- Open question: how do you know when the data has "drifted?"

What to monitor?

- Agile philosophy: "Continuous Delivery." Monitor proactively!
- Especially hard when there are no labels
- First-pass monitoring
  - Model output distributions
  - Averages, percentiles
- More advanced monitoring
  - Model input (feature) distributions
  - ETL intermediate output distributions
  - Can get tedious if you have many features or steps in ETL
- Still unclear how to quantify, detect, and act on

How do you know when the data has "drifted?"

- Track mean and variance? Fails in:
  - Distributions with multiple modes
  - Skew (symmetry) and kurtosis (tail)
- Run statistical tests on all your features?
- This method can have a high "false positive rate"
  - In my experience, this method has flagged distributions as "significantly different" more than I want it to
  - In the era of "big data" (we have millions of data points), p-values are not useful to look at[2]

[2] Lin, Lucas, and Shmueli, "Too Big to Fail: Large Samples and the p-Value Problem".

How do you know when the data has "drifted?"

You don't really know.

An unsatisfying solution is to retrain the model to be as fresh as possible.
What kinds of bugs will surface when you are continually training models?

Types of production ML bugs

- Data changed or corrupted
- Data dependency / infra issues
- Logical error in the ETL code
- Logical error in retraining a model
- Logical error in promoting a retrained model to inference step
- Change in consumer behavior
- Many more

Why production ML bugs are hard to find and address

- ML code fails silently (few runtime or compiler errors)
- Little or no unit or integration testing in feature engineering and ML code
- Very few (if any) people have end-to-end visibility on an ML pipeline
- Underdeveloped monitoring and tracing tools
- So many artifacts to keep track of, especially if you retrain your models

Retraining regularly

Monitoring is not enough; we also need to retrain.

- Use MLFlow Model Registry[3] to keep track of which model is the best model
- Alternatively, can also manage versioning and a registry ourselves (what we do now)
- At inference time, pull the latest/best model from our model registry

Now, we need some tracing[4]...

[3] `https://www.mlflow.org/docs/latest/model-registry.html`

[4] Hermann and Del Balso, *Scaling Machine Learning at Uber with Michelangelo*.

mltrace

- Coarse-grained lineage and tracing
- Designed specifically for complex data or ML pipelines
- Designed specifically for Agile multidisciplinary teams
- Alpha release[5] contains Python API to log run information and UI to view traces for outputs

[5] https://github.com/loglabs/mltrace

mltrace design principles

- Utter simiplicity (user knows everything going on)
- No need for users to manually set component dependencies – the tool should detect dependencies based on I/O of a component's run
- API designed for both engineers and data scientists
- UI designed for people to help triage issues **even if they didn't build the ETL or models themselves**
  - When there is a bug, whose backlog do you put it on?
  - Enable people who may not have developed the model to investigate the bug

mltrace concepts

- Pipelines are made of components (ex: cleaning, featuregen, split, train)
- In ML pipelines, **different** artifacts are produced (inputs and outputs) when the **same** component is run more than once
- `Component` and `ComponentRun` objects[6]
- Decorator interface similar to Dagster "solids"[7]
  - User specifies component name, input and output variables
- Alternative Pythonic interface similar to MLFlow tracking[8]
  - User creates instance of `ComponentRun` object and calls `log_component_run` on the object

[6] `https://mltrace.readthedocs.io/en/latest`
[7] `https://docs.dagster.io/concepts/solids-pipelines/solids`
[8] `https://www.mlflow.org/docs/latest/tracking.html`

mltrace integration example

```
def clean_data(raw_data_filename: str, raw_df: pd.DataFrame,
                                    month: str, year: str,
                                   component: str) -> str:
    first, last = calendar.monthrange(int(year), int(month))
    first_day = f'{year}-{month}-{first:02d}'
    last_day = f'{year}-{month}-{last:02d}'
    clean_df = helpers.remove_zero_fare_and_oob_rows(
        raw_df, first_day, last_day)

    # Write "clean" df to s3
    output_path = io.save_output_df(clean_df, component)

    return output_path
```

mltrace integration example

```
@register('cleaning', input_vars=['raw_data_filename'],
                                  output_vars=['output_path'])
def clean_data(raw_data_filename: str, raw_df: pd.DataFrame,
                                   month: str, year: str,
                                  component: str) -> str:
    first, last = calendar.monthrange(int(year), int(month))
    first_day = f'{year}-{month}-{first:02d}'
    last_day = f'{year}-{month}-{last:02d}'
    clean_df = helpers.remove_zero_fare_and_oob_rows(
        raw_df, first_day, last_day)

    # Write "clean" df to s3
    output_path = io.save_output_df(clean_df, component)

    return output_path
```

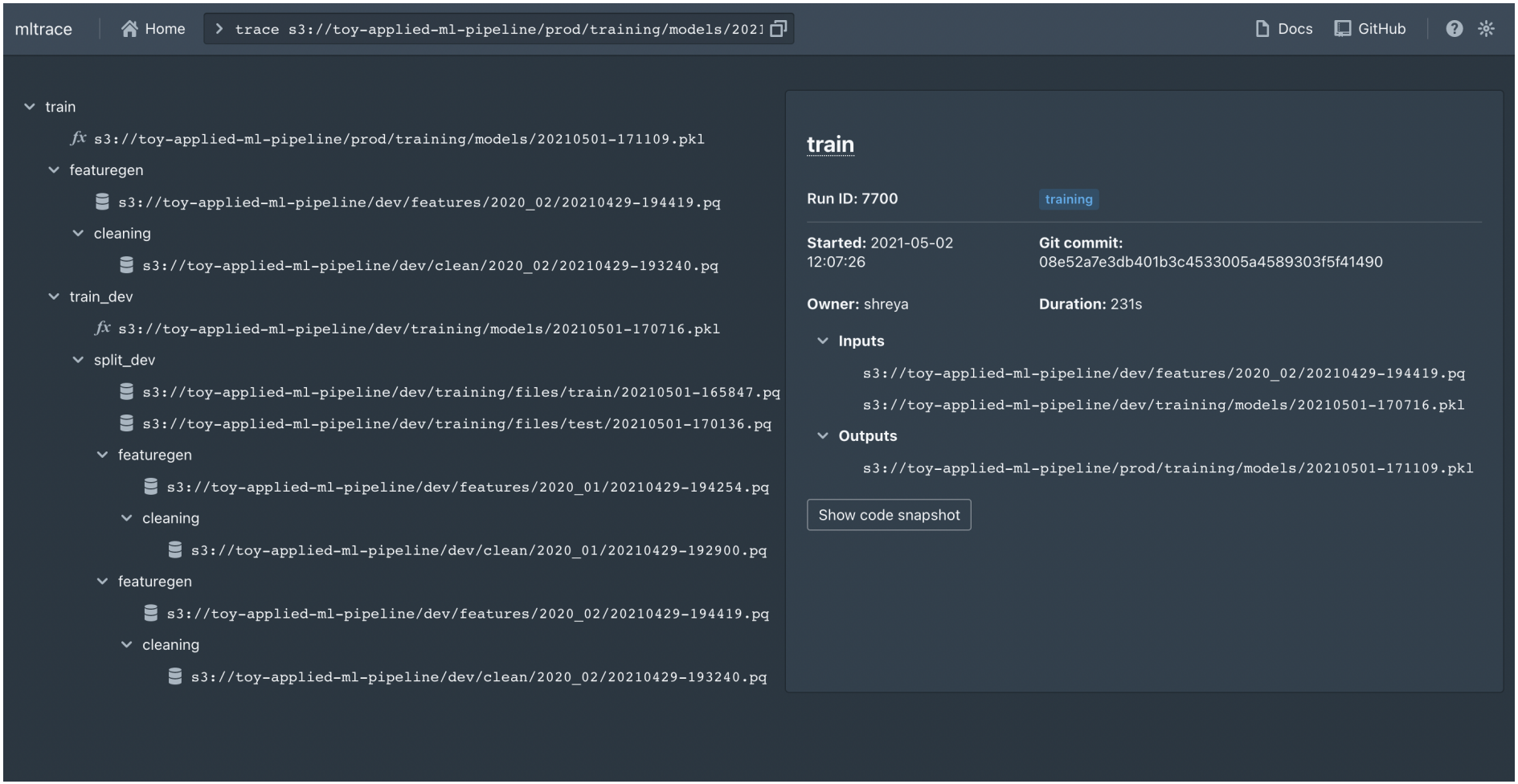
mltrace integration example

```
from mltrace import create_component, tag_component,
                                    register

create_component('cleaning', 'Cleans the raw data with basic
                                     OOB criteria.', 'shreya')
tag_component('cleaning', ['etl'])

# Run function
raw_df = io.read_file(month, year)
raw_data_filename = io.get_raw_data_filename('2020', '01')
output_path = clean_data(raw_data_filename, raw_df, '01', '
                                    2020', 'clean/2020_01')
print(output_path)
```

mltrace UI demo



mltrace immediate roadmap[9]

- gRPC integrations for logging outside of Python
- Prometheus integrations to easily monitor outputs
- DVC integration to version data
- MLFlow integrations
- Causal analysis for ML bugs — if you flag several outputs as mispredicted, which component runs were common in producing these outputs? Which component is most likely to be the biggest culprit in an issue?
- Lineage at the record/row level

[9] Please email shreyashankar@berkeley.edu if you're interested in contributing!

Talk recap

- Discussed challenges around continuously retraining models and maintaining ML production pipelines
- Introduced mltrace, a tool developed for ML pipelines that performs coarse-grained lineage and tracing

Areas of future work in MLOps

- Many more problems around deep learning
  - Embeddings as features – if someone updates the upstream embedding model, do all the data scientists downstream need to immediately change their models?
  - "Underspecified" pipelines can pose threats[10]
- How to enable anyone to build dashboards or monitor pipelines?
  - ML people know what to monitor, infra people know how to monitor
  - We should strive to build tools to allow engineers and data scientists to be more self sufficient

[10] D'Amour et al., *Underspecification Presents Challenges for Credibility in Modern Machine Learning*.

Miscellaneous

- Code for this talk: `https://github.com/shreyashankar/debugging-ml-talk`
- Toy ML Pipeline with mltrace, Prometheus, and Grafana integrations: `https://github.com/shreyashankar/toy-ml-pipeline/tree/shreyashankar/db`
- mltrace: `https://github.com/loglabs/mltrace`
- Contact info
  - Email: shreyashankar@berkeley.edu
  - Twitter: @sh_reya
- Thank you!

References

D'Amour, Alexander et al. *Underspecification Presents Challenges for Credibility in Modern Machine Learning*. 2020. arXiv: 2011.03395 [cs.LG].

Hermann, Jeremy and Mike Del Balso. *Scaling Machine Learning at Uber with Michelangelo*. 2018. URL: https://eng.uber.com/scaling-michelangelo/.

Lin, Mingfeng, Henry Lucas, and Galit Shmueli. "Too Big to Fail: Large Samples and the p-Value Problem". In: *Information Systems Research* 24 (Dec. 2013), pp. 906–917. DOI: 10.1287/isre.2013.0480.