CONTINUOUS INTELLIGENCE
KEEPING YOUR AI APPLICATION IN PRODUCTION

Emily Gorcenski
Arif Wider

HOW & WHY TO APPLY CONTINUOUS DELIVERY TO MACHINE LEARNING
5500 technologists with 40 offices in 14 countries

Partner for technology driven business transformation

100+ books written

#1 in Agile and Continuous Delivery
TECHNIQUES

Continuous delivery #8 for machine learning (CD4ML) models

©ThoughtWorks 2019
Strata Data Conference London, May 1, 2019
“Continuous Delivery is the ability to get changes of all types — including new features, configuration changes, bug fixes and experiments — into production, or into the hands of users, safely and quickly in a sustainable way.”

- Jez Humble & Dave Farley
PRINCIPLES OF CONTINUOUS DELIVERY

→ Create a Repeatable, Reliable Process for Releasing Software
→ Automate Almost Everything
→ Build Quality In
→ Work in Small Batches
→ Keep Everything in Source Control
→ Done Means “Released”
→ Improve Continuously
WHY CONTINUOUS DELIVERY?

TO MAKE MY LIFE EASIER!
<table>
<thead>
<tr>
<th>Market</th>
<th>Germany</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make</td>
<td>BMW</td>
</tr>
<tr>
<td>First registration</td>
<td>2019</td>
</tr>
<tr>
<td>Model</td>
<td>i8 Convertible 2 doors</td>
</tr>
<tr>
<td>Fuel type</td>
<td>Electric/Gasoline</td>
</tr>
<tr>
<td>Power</td>
<td>170 kW (231 hp)</td>
</tr>
<tr>
<td>Transmission</td>
<td>Automatic</td>
</tr>
<tr>
<td>Mileage (km)</td>
<td>20000</td>
</tr>
<tr>
<td>I am:</td>
<td>a prospective buyer of this vehicle</td>
</tr>
</tbody>
</table>

Car evaluation:
Make your car stand out.
With the right price.
Continuous Delivery isn't a technology or a tool; it is a practice and a set of principles. Quality is built into software and improvement is always possible.

But machine learning systems have unique challenges; unlike deterministic software, it is difficult—or impossible—to understand the behavior of data-driven intelligent systems. This poses a huge challenge when it comes to deploying machine learning systems in accordance with CD principles.

PRODUCTIONIZING ML IS HARD

**How do we apply decades of software delivery experience to intelligent systems?**

Production systems should be:

- Reproducible
- Testable
- Auditable
- Continuously Improved

Machine Learning is:

- Non-deterministic
- Hard to test
- Hard to explain
- Hard to improve
WHY BOTHER WITH CONTINUOUS DELIVERY FOR MACHINE LEARNING?
MANY SOURCES OF CHANGE

Data

- Schema
- Sampling over Time
- Volume

+ Model

- Research, Experiments
- Training on New Data
- Performance

+ Code

- New Features
- Bug Fixes
- Dependencies

©ThoughtWorks 2019
Strata Data Conference London, May 1, 2019

Icons created by Noura Mbarki and I Putu Kharismayadi from Noun Project
HOW DOES CODE CHANGE VERSION?

- Bug fixes
- New features
- Dependencies
- Operating systems
- Scope change
- Etc.

This axis is the best-understood axis of version change for machine learning systems.

Code

Code is a broad concept. It can refer to the source code of our services and systems. It can represent our infrastructure. It can refer to the code used to access, transform, or prepare data. It can refer to the code used to train, validate, and assess models.
Model

Model here refers to both the *modeling approach* (e.g. random forest classification) and the actual artifact produced (e.g. a pkl file). A model is a product of **hyperparameters** (not varied during training) and **parameters** (computed during training).

**HOW DOES A MODEL CHANGE VERSION?**

- Research — trying new approaches
- Dependencies
- New data
- Performance improvements
- Etc.

This axis is not as well-understood and machine learning, as a field, is changing so rapidly that many models can be expected to be obsolete in short order.
DATA IS MORE COMPLICATED

Data is an assessment of the state of the universe at a point in time. It has a measurement and a shape.
Sampling

Sampling refers to the values in the data themselves. We can’t measure everything, but what we do measure is a sample of the universe. These values have properties like support, distribution, cardinality, etc.

HOW DOES SAMPLING CHANGE VERSION?

- Time—it marches ever on
- Extrinsic changes in the universe
- Inserts, Deletions, Updates
- Etc.

It is very difficult to specify or detect shifts in “versions” here except by example. We often think of this as moving forward with time, but that versions are not necessarily time-dependent.
Schema

Schema refers to the shape of our data. This can be a database schema, e.g. a table/column/constraint definition, or it could just refer to shape of the incoming data (e.g. JSON fields, XML schema, etc).

HOW DOES SCHEMA CHANGE VERSION?

- Software updates
- Requirements changes
- Migrations
- Etc.

It is very difficult to specify or detect shifts in “versions” here except by example. We often think of this as moving forward with time, but that versions are not necessarily time-dependent.
ML SYSTEMS HAVE FOUR “VERSION AXES”

Any data-driven service can experience version drift along any of these axes independently or jointly.

Sampling

Schema

Code

Model
BUILDING A FULL STACK
Our code, data, and models should be versioned and shareable without unnecessary work.

What are the challenges?
- Data and models can be very large
- Data can vary invisibly
- Data scientists need to share work and it must be repeatable

What does the ideal solution look like?
- Large artifacts stored in arbitrary storage linked to source repo
- Data scientists can encode work process at repeat in one step

What solutions are out there now?
- Storage: S3, HDFS, etc
- Git LFS
- Shell scripts
- dvc
- Pachyderm
- jupyterhub
MODEL PERFORMANCE TRACKING

We should be able to scale model development to try multiple modeling approaches simultaneously.

What are the challenges?
- Hyperparameter tuning and model selection is hard
- Tracking performance depends on other moving parts (e.g. data)

What does the ideal solution look like?
- Links models to specific training sets and parameter sets
- Is differentiable
- Allows visualization of results

What solutions are out there now?
- dvc
- MLFlow
- etc...
Our code, data, and models should be versioned and shareable without unnecessary work.

What are the challenges?
- We should be monitoring the health and quality of our production ML system
- Deployment is usually deeply coupled to code

What does the ideal solution look like?
- One-click deployment
- Easy, context-aware logging
- Performance can be benchmarked to research results

What solutions are out there now?
- Elasticsearch
- Kibana
- Logstash
- CircleCI
- GoCD
- etc.
Machine learning may automate things but it does not automatically generate value.

**What are the challenges?**
- It is hard to map data science investment to business value
- We should be extracting continuous insights from data

**What does the ideal solution look like?**
- KPIs can be visualized and linked to work efforts
- Patterns should emerge
- New questions arise

**What solutions are out there now?**
- BI tools (e.g. Tableau, Chartio, SAS, etc.)
- Work tracking and documentation tools (Github, JIRA, Confluence)
Doing CD with Machine Learning is still a hard problem

WHAT DO WE NEED IN OUR STACK?

- DISCOVERABLE AND ACCESSIBLE DATA
- VERSION CONTROL AND ARTIFACT REPOSITORIES
- CONTINUOUS DELIVERY ORCHESTRATION TO COMBINE PIPELINES
- INFRASTRUCTURE FOR MULTIPLE ENVIRONMENTS AND EXPERIMENTS
- MODEL PERFORMANCE ASSESSMENT TRACKING
- MODEL MONITORING AND OBSERVABILITY
PUTTING EVERYTHING TOGETHER

- Data Science, Model Building
- Model Evaluation
- Production Model
- Integration Testing
- Deployment
- Monitoring

- Discoverable and Accessible Data
- Source Code + Executables
- Model + parameters
- Training Data
- Test Data
- Production Data

CD Tools and Repositories

©ThoughtWorks 2019
Strata Data Conference London, May 1, 2019
THE TOOLS ARE THERE AND THEY ARE FREE

There are many options for tools and technologies to implement CD4ML
YOUR OBJECTIVE FUNCTION MUST LINK TO BUSINESS VALUE
join.thoughtworks.com

THANK YOU!

Emily Gorcenski (egorcens@thoughtworks.com)
Arif Wider (awider@thoughtworks.com)