Machine Learning Productization

June 12th, 2019

Sharath Bennur,
Machine Learning Lead / Architect
IQVIA Technologies

Gartner Data Science Team Survey of January 2018:

“Over 60% of models developed with the intention of operationalizing them were never actually operationalized“

Magic Quadrant for Data Science and Machine Learning Platforms -
https://www.gartner.com/doc/reprints?id=1-65OPDHH&ct=190125&st=sb&submissionGuid=7016ef12-38ba-47d8-a435-43c45f572d0d

Process + Platform for ML Productization

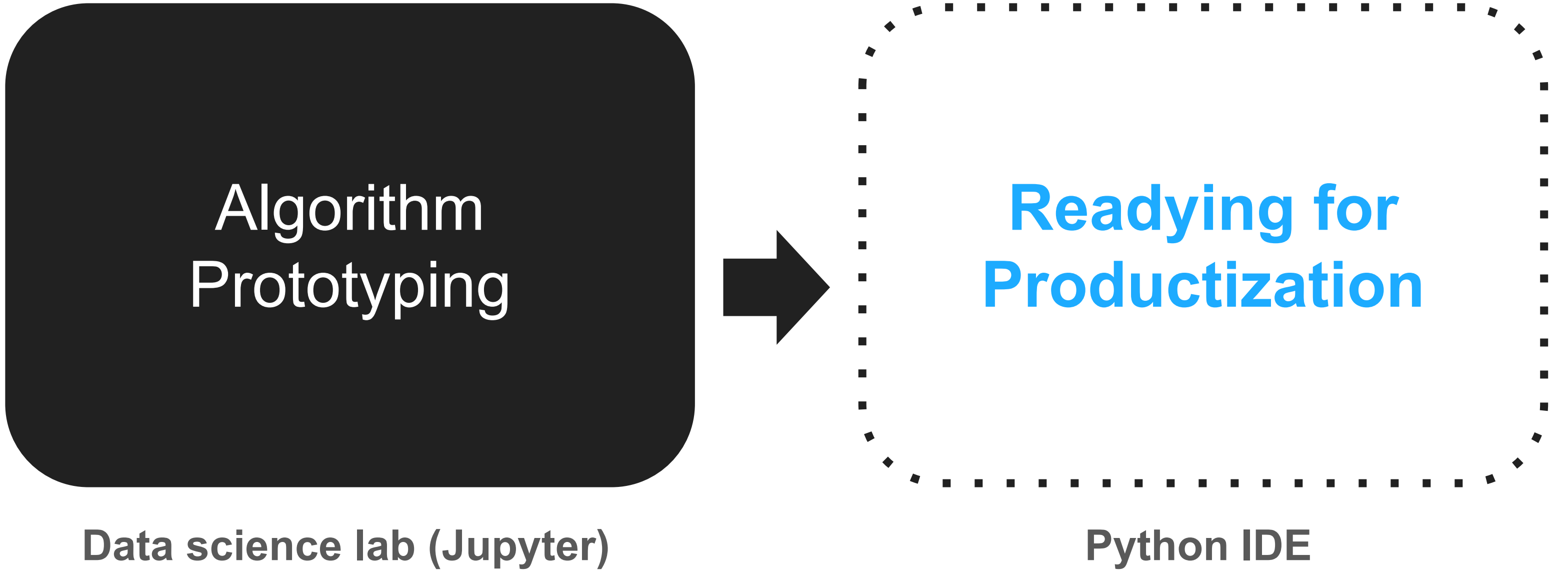
When we build new models / algorithms

Algorithm Prototyping

Data science lab (Jupyter)

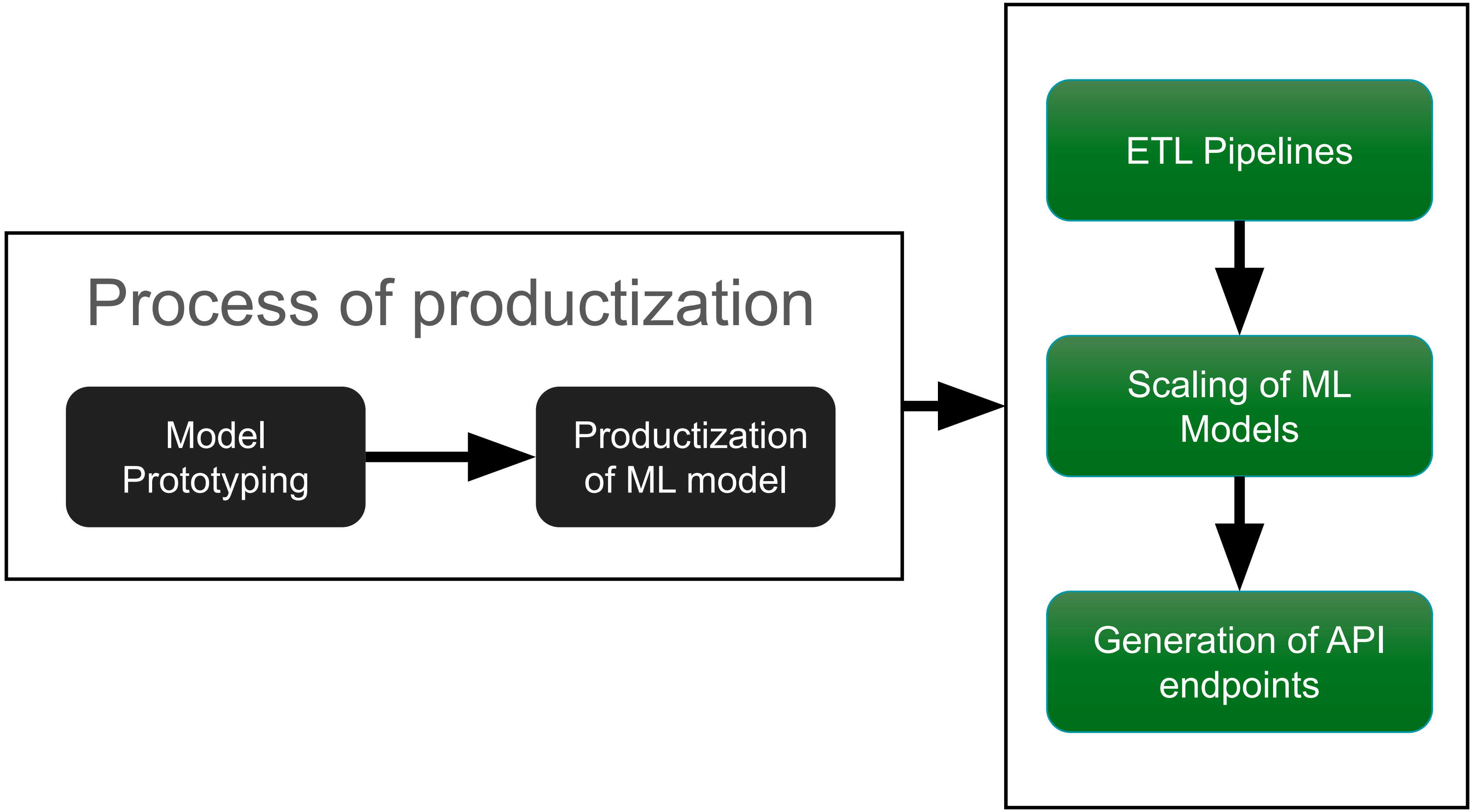
But that's not production ready!

Algorithm code has to be updated for productization



Algorithm code has to be made platform compatible

An ML Platform enables the construction of ML Pipelines



ML Pipeline

An ML platform includes:

- Automation through CI-CD pipelines
- UI interface for non-technical delivery personnel
- Ability to manage multiple ML models and versions
- Feedback loop to measure success
- Risk Management – monitoring health of the platform and models
- Disaster recovery

Function of an ML platform – support ML-Operations

ETL Pipelines (Airflow)

- DAGs to pull data from different sources
- DAGs to create flat files with data model
- Automation, scheduling and logging

Scaling of ML Models (Kubernetes)

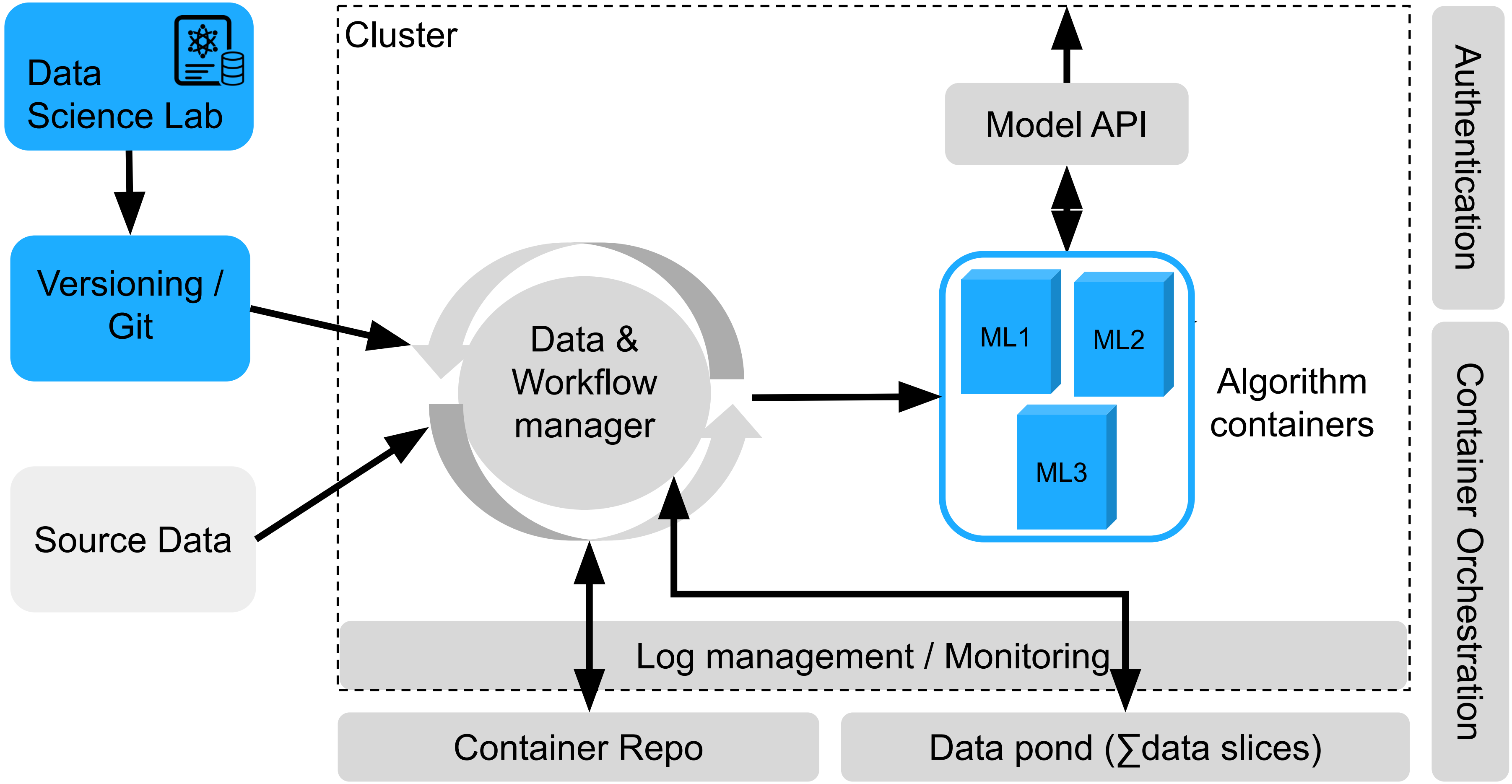
- Auto-scaling based on data load (input or output)
- Use of multiple ML platforms on the same infrastructure
- High performance in response time
- Cloud-agnostic (in theory)

API Endpoint Generation

- Automatic creation of container end-point
- API scaling and logging
- Feedback framework to measure performance
- AB testing framework

What does an ML platform look like?

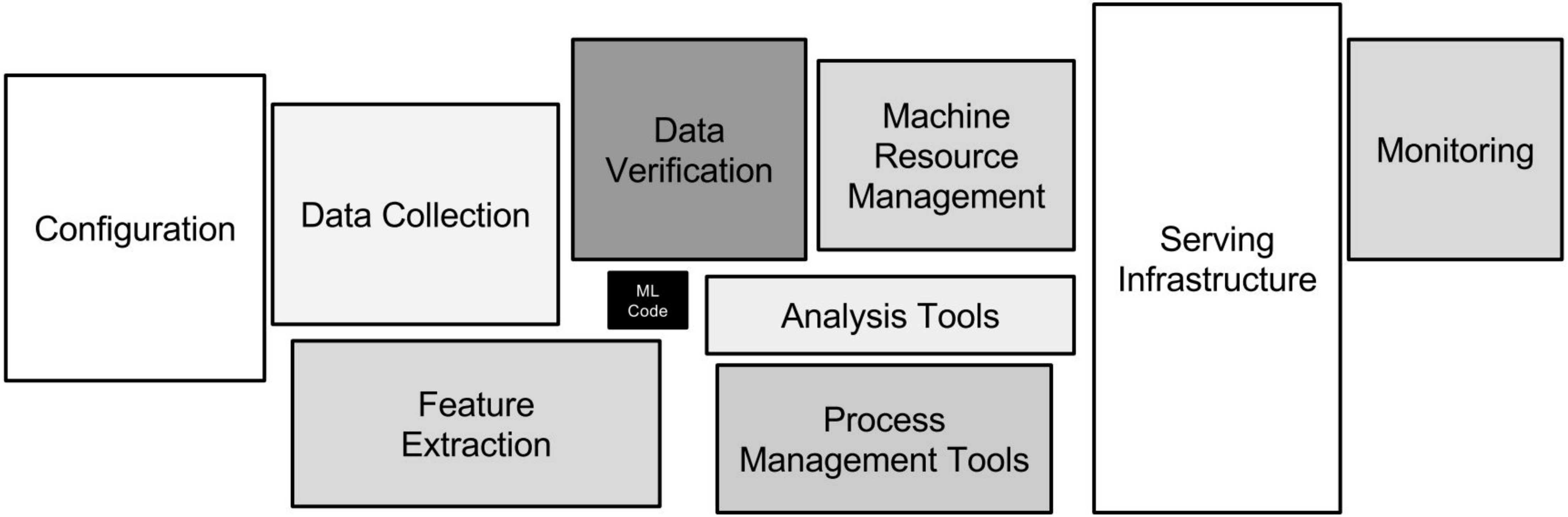
These enable automated building of end to end ML pipelines



Core technology choices:
- Container orchestration
- CI / CD pipelines
- Versioning / Git
- Container repository
- Logging and monitoring
  - Model performance
  - Alerting for failures
  - Data freshness

Lessons learnt

Think Software engineering + Data science



Hidden Technical Debt in Machine Learning Systems
(https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf)

Lessons learned

To deliver intelligence to multiple tenants of multiple products at scale, you need to…

productize your Machine Learning algorithms and publish them on an enterprise-grade platform

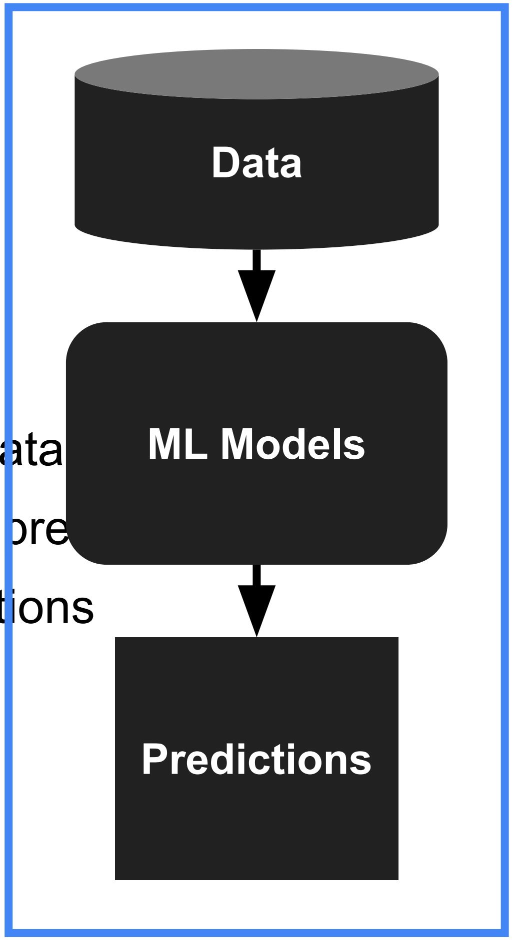
1. Think software engineering first, ML second
2. Think ML Operations
3. Think delivery
4. Combat technical debt

1. Think software engineering

ML in production needs software engineering

Software engineering (+Data science)

- Iterative development and delivery
- Efficient code management (versioning, frequent commits, etc)
- Tests, lots of tests – not just on *local* but also on *cluster*
  - › *Data tests* – test for errors in data, staleness of data and distribution of data
  - › *Model tests* - Sanity checks to test model predictions + Latency tests for pre
  - › *ML-specific integration tests* - Test entire ML pipeline from data to predictions

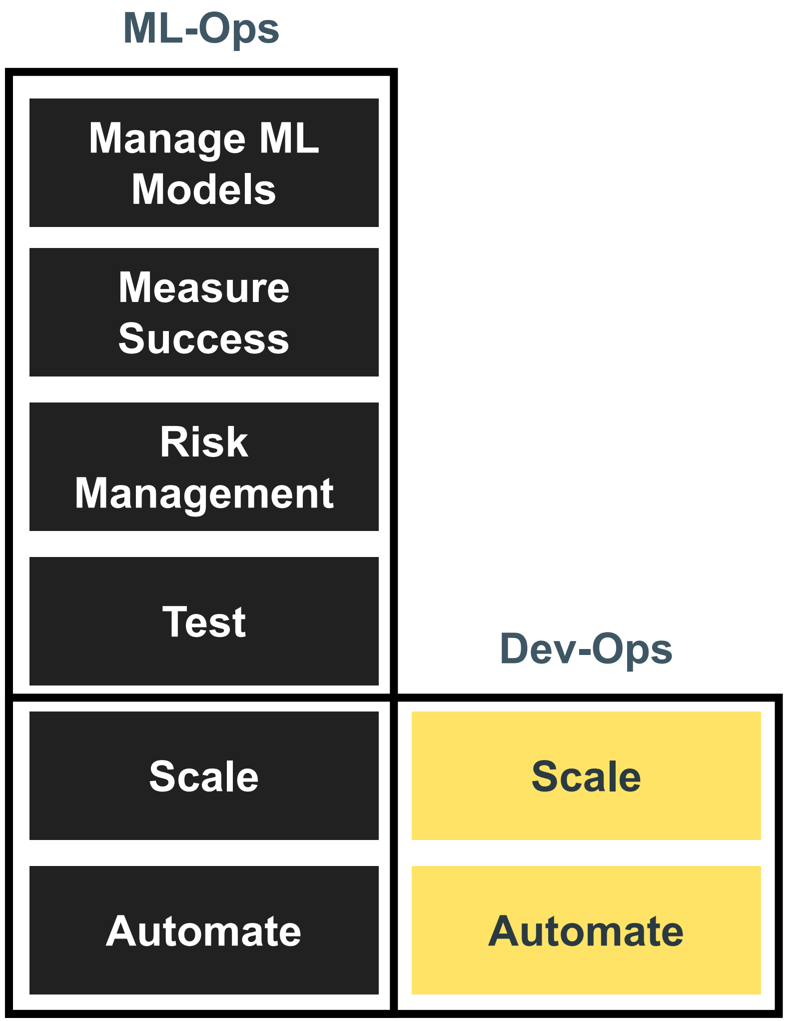


ML Integration

What's your ML Test Score? A rubric for ML production systems (https://ai.google/research/pubs/pub45742)

2. Think operations, ML operations

- The platform may have to serve many tenants/teams for 10+ different products
  - Automated workflow – deploy your ML pipeline to production automatically
  - Easy/automated management & monitoring for ProdOps team
- To build an enterprise-grade SaaS Machine Learning platform, you need flawless operations
  - Easy/automated deployment of new tenants and new algorithms for existing tenants
  - Continuous health monitoring for models
  - Focus on Non Functional requirements: scalability, high-availability, disaster recovery



2. Think operations, ML operations

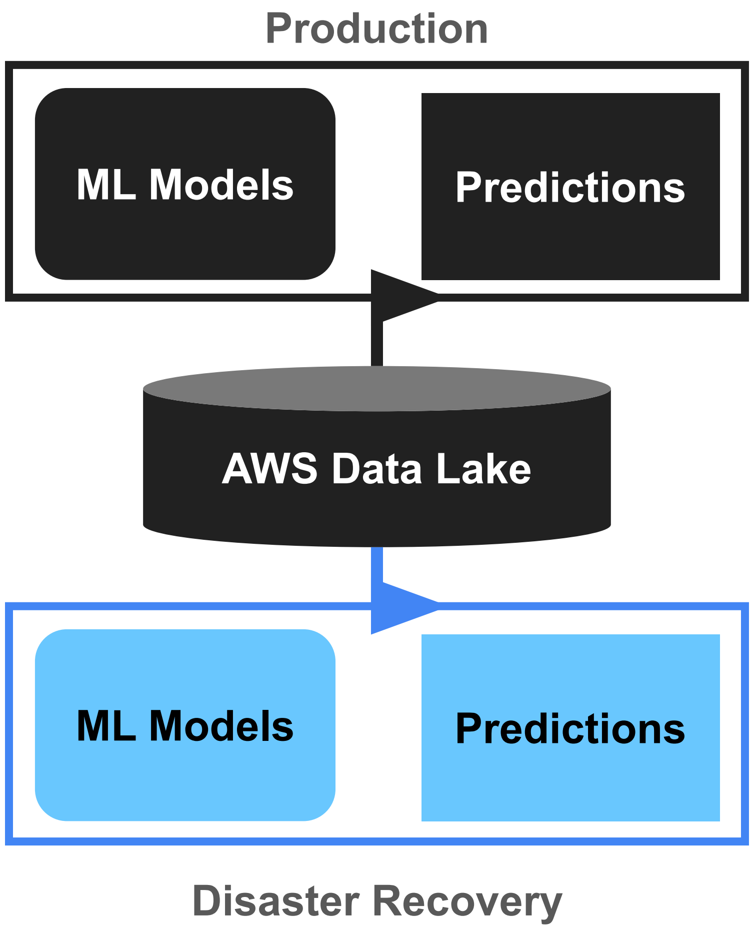
Disaster recovery for ML

Recovering from a loss of an ML platform can be problematic, there are ML specific challenges like:

- Loss of ML models and data
- Potential loss of ML model/data state for models that are updated in real/near-real time

Our approach to DR:

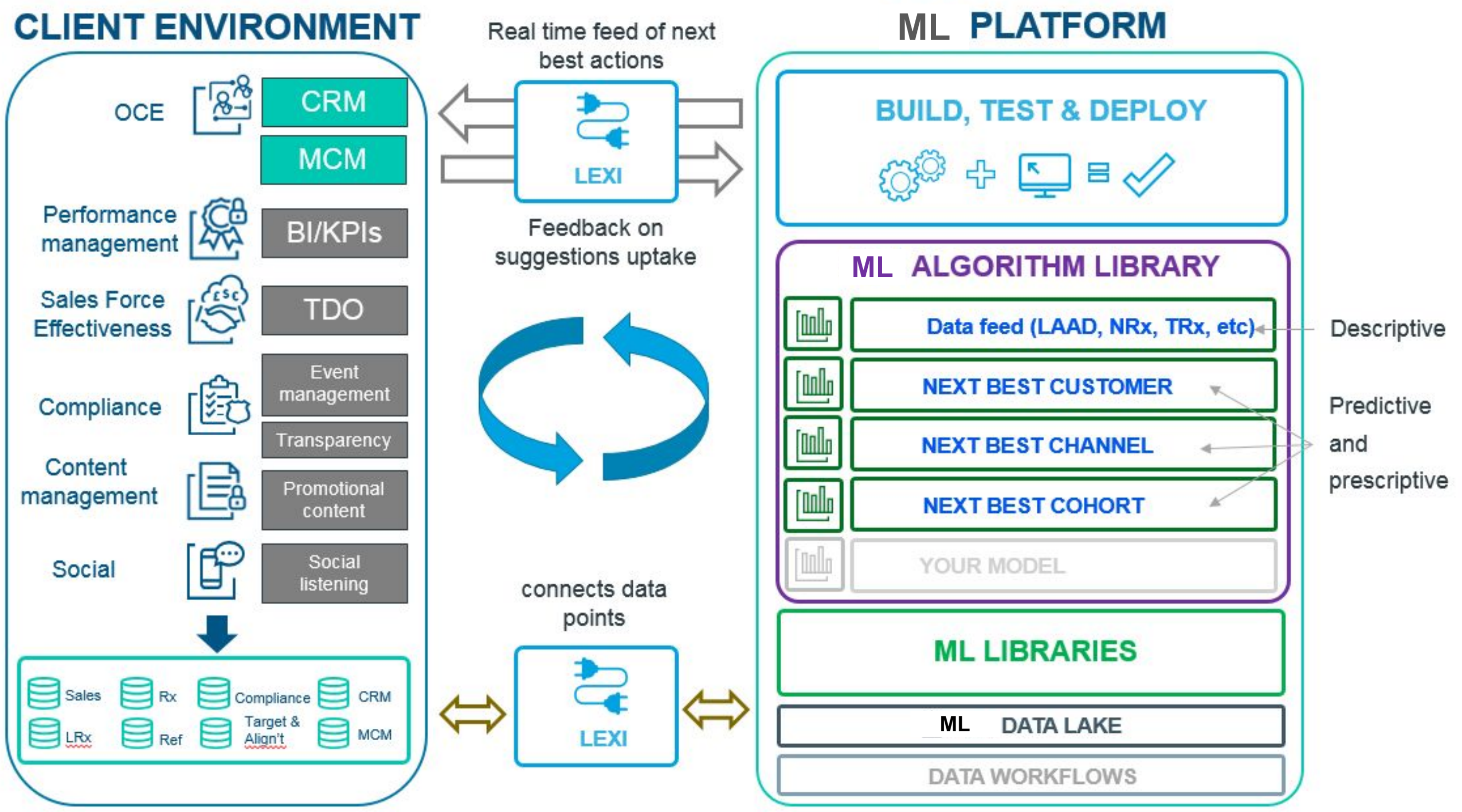
- Use cloud object database (AWS S3) with multi-region replication
- Recovery of models + predictions (through backup of ML containers on ECR)



3. Think delivery

Define integration requirements at the onset of the project

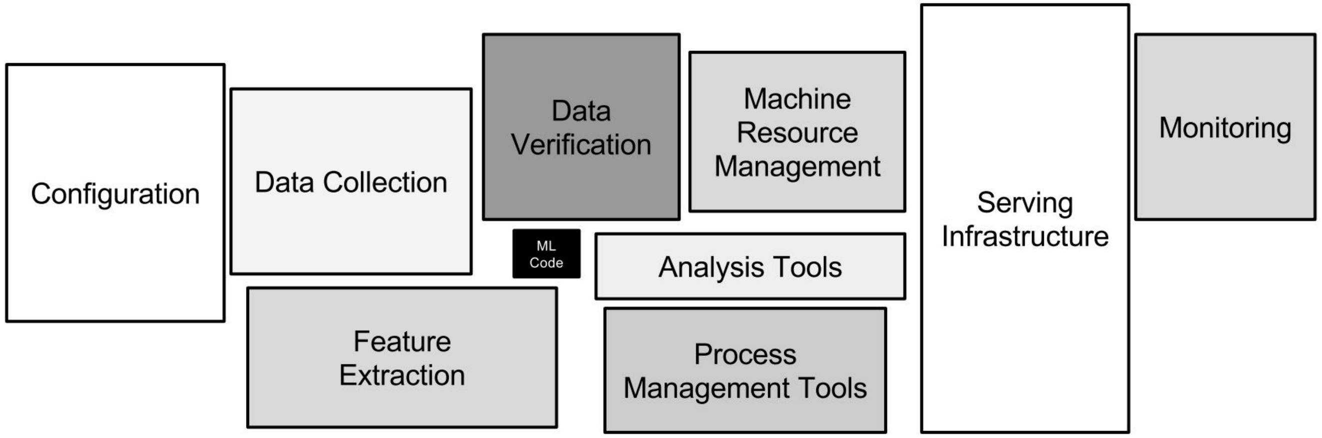
- What will the user experience be?
  - Push or pull
  - Which step in the workflow
  - Frequency of data syncs
- How is the product going to consume
  - Real-time or batch?
  - Exchange format?
- Design solid requirements
  - Happy path
  - Error management
  - Edge cases (what if?)



4. Think about technical debt

ML Ops reduces technical debt

- By Improving ML-Ops - mature ML systems can have hundreds of models running
  - Combine ML and engineering when possible
  - Visualize and detect blockages or errors
  - Understand data dependencies and integration
- Having clear process and standards for productiza
  - Move towards isolated models and ensembles v
  - Reducing model complexity
  - Managing resources & configurations safely for
  - Create or use a standardized template



Questions?