CSCI E-103

Data Engineering for Analytics to Solve Business Challenges

Good Data -> Good Models

Improving Model Insights

Lecture 09

Anindita Mahapatra & Eric Gieseke

Harvard Extension, Fall 2025

Agenda

- Announcements
- Review Last Lecture
- MLOps
  - Model Creation, Selection, Registration, Deployment
  - From model centric to data centric
  - MLOps Maturity Progression
  - Automation as the tool to control model management
- Lab
  - MLOps
  - `%pip install dbdemos`
  - `import dbdemos`
  - `dbdemos.install('mlops-end2end', catalog='cscie103_catalog', schema='mlops', create_schema=True)`
  -

Announcements

- Case Study #1 (Data Architecture Evolution Research)
- Assignment-4
  - Based on MLFlow
  - Released soon (we will review on Thursday)
- Case Study #2 (Industry Competitive Research)
  - Group
- Final Project
  - Group
- Quiz
  - Quiz#1 - done
  - Quiz#2 - released towards end of Nov

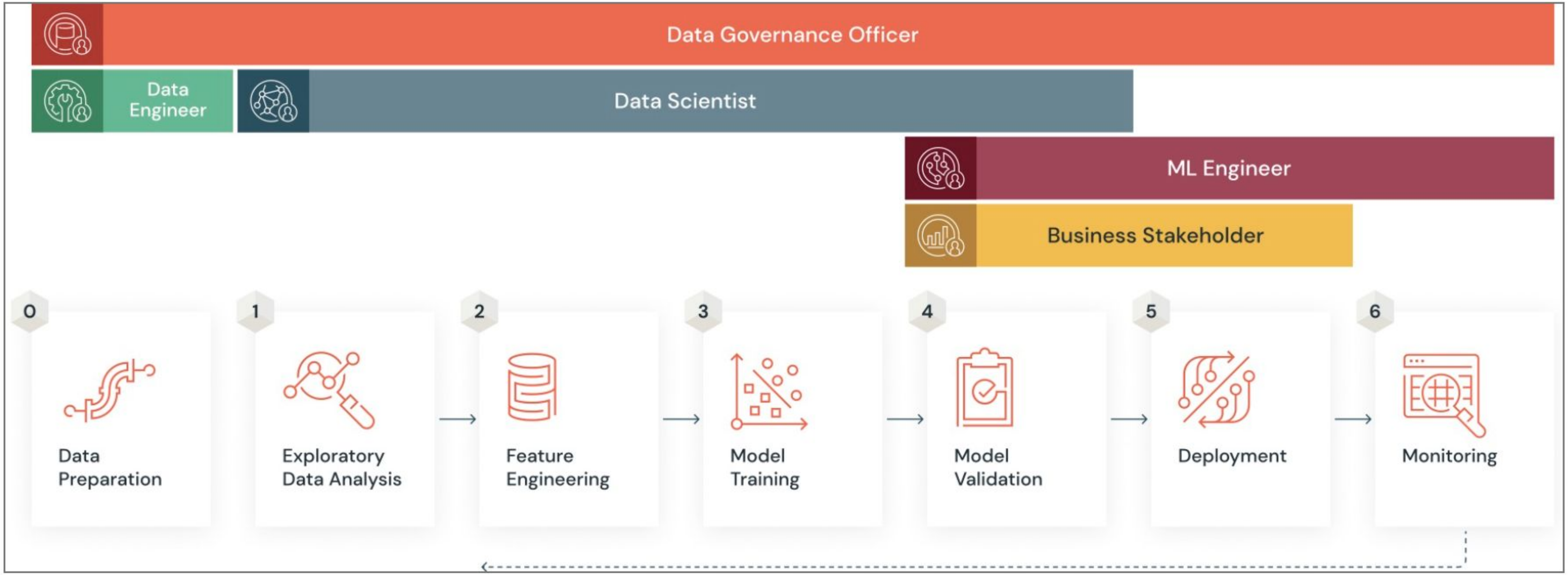
Review Previous Material

Model staleness over time is referred to as	Model Drift
MLFlow is an open source framework for	Model lifecycle management
MLFlow Tracking Server tracks	Parameters, metrics, artifact, code, data, tags, ..
MLFlow Registry helps with model	Discoverability, versioning, promotion
Can MLflow work on-prem	Yes
Process of getting data ready for input to a model	Feature Engineering
Different types of model serving/scoring	Batch, streaming, realtime model serving
Overfit refers to	A statistical model that contains more parameters than can be justified by the data
What aids with feature reuse	Feature Store
AutoML refers to	Automating the tasks of applying machine learning to real-world problems

People and process

- **Business Stakeholder** → Responsible for business value of the ML solution
- **Data Engineer** → Builds data pipelines
- **Data Scientist** → Translates business problem; trains, tunes model
- **ML Engineer** → Deploys ML model to production
- **Data Governance Officer** → Responsible for data governance and compliance

People and process



Business Context: Customer Retention

No data science and ML can start without a business problem at hand.

You are on a marketing analytics team and you have a lot of **demographic** and **historical service data** on your customers that have **churned**, which has been put into a SQL Analytics dashboard.

The data team has been asked by business stakeholders if you can go further and **predict** which customers are likely to churn. Knowing this will allow the business to take action and **retain revenue**.

Sounds simple enough. What steps do we need to take?
Example

Operational Workflow At a Glance:

- Data Prep & Featurization
- Baseline Model
- Setup Webhooks
- Promote Best Run to Registry
- Testing
- Staging Batch Inference
- Update Dashboard
- Schedule Monthly Retrain Job

Data Engineer   Data Scientist   ML Engineer

Operational Workflow:
Data Engineering

Data Prep *ETL*	Baseline Model	Setup Webhooks	Promote Best Run to Registry	Testing	Staging Batch Inference *Spark UDF*	Update Dashboard Schedule Monthly Retrain Job


Data Engineer


Data Scientist


ML Engineer

Operational Workflow:
Data Scientist

Data Prep & Featurization

EDA

Feature Engineering

Feature Store

Baseline Model

AutoML

MLflow autologging

Hyperopt + Spark

Setup Webhooks

Promote Best Run to Registry

Annotate model

Request transition to staging

Testing

Staging Batch Inference

Update Dashboard

Schedule Monthly Retrain Job

EDA: Exploratory Data Analysis

Operational Workflow:
ML Engineering

Data Prep & Featurization

Baseline Model

Setup Webhooks

Slack notifications

Databricks Jobs (Testing)

Promote Best Run to Registry

Testing

Model schema

Demographic accurary

Docs & artifacts

Set tags

Approve/reject transitition

Staging Batch Inference

Update Dashboard

Schedule Monthly Retrain Job

Data Engineer
Data Scientist
ML Engineer

Operational Workflow:
Data Team

Data Prep & Featurization

ETL + EDA

Feature Engineering

Koalas

Feature Store

Baseline Model

AutoML

MLflow autologging

Hyperopt + Spark

Setup Webhooks

Slack notifications

Databricks Jobs (Testing)

Promote Best Run to Registry

Annotate model

Request transition to staging

Testing

Model schema

Demographic accurary

Docs & artifacts

Set tags

Approve/reject transitition

Staging Batch Inference

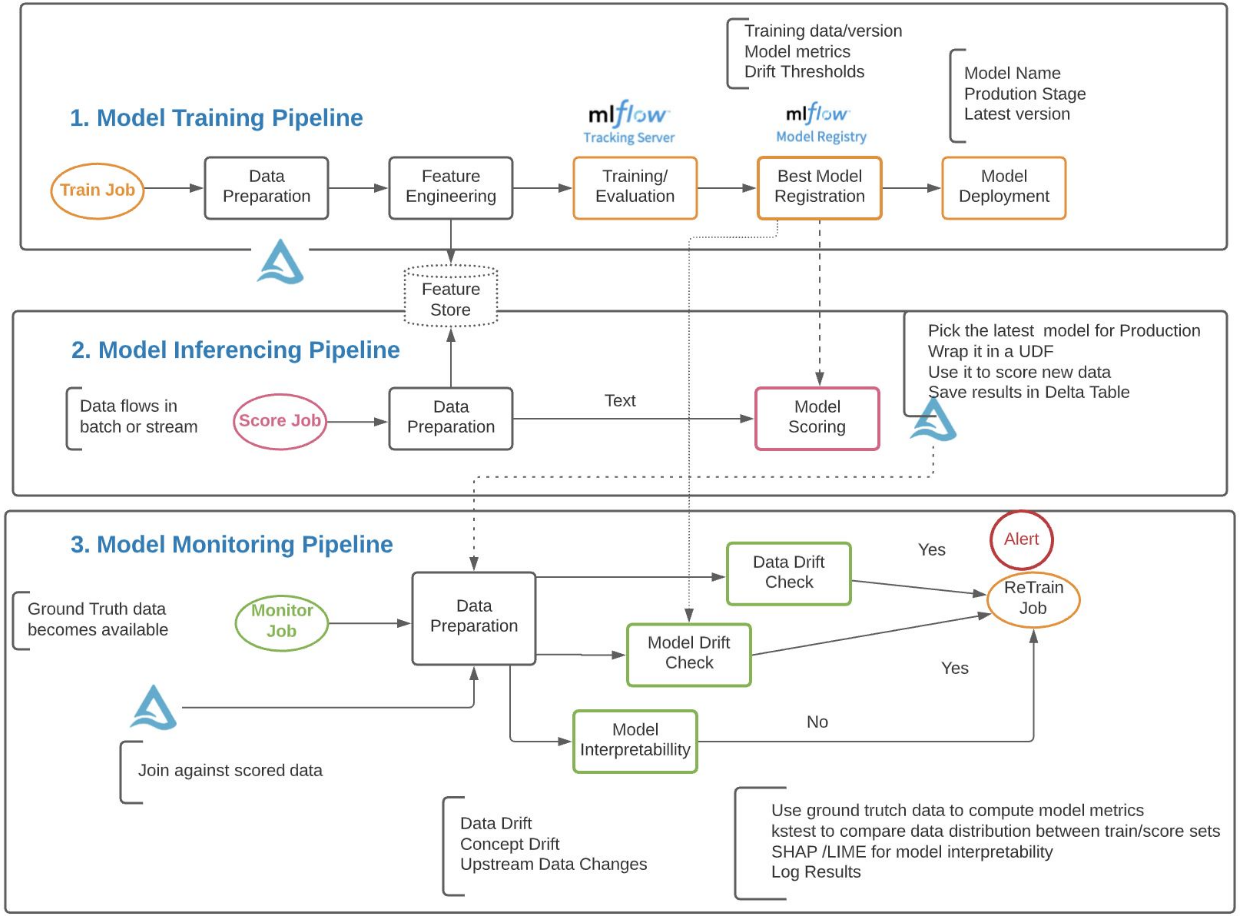
Spark UDF

Update Dashboard

Schedule Monthly Retrain Job

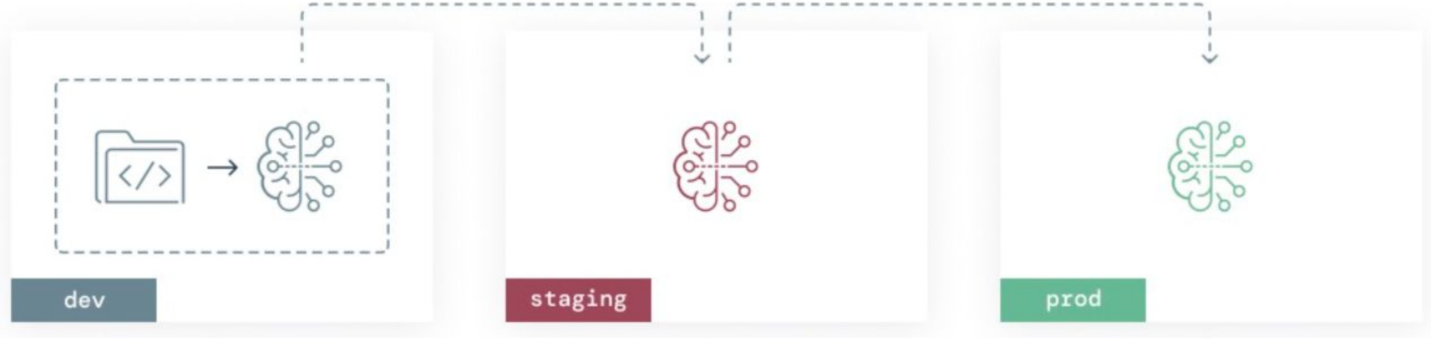
Data Engineer Data Scientist ML Engineer

Model Training, Inferencing, and Monitoring

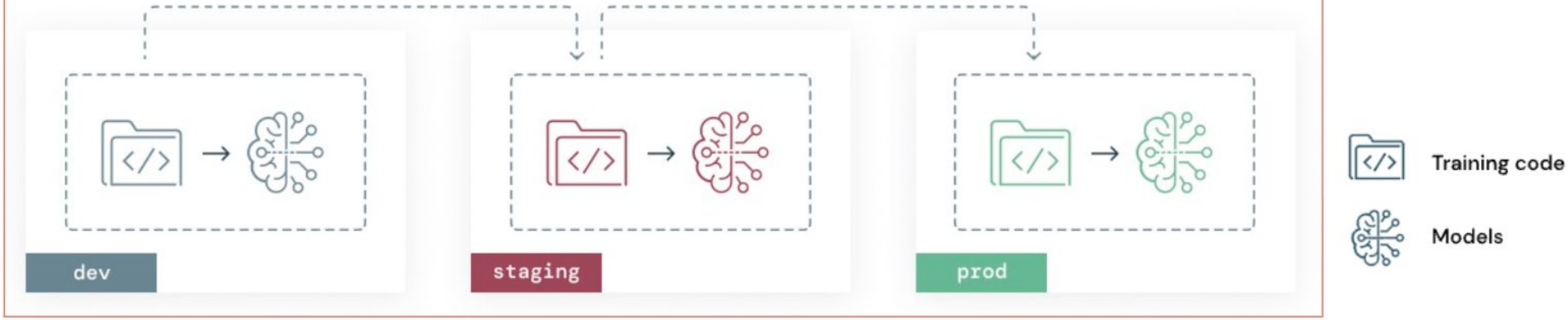


ML deployment patterns

Deploy Models



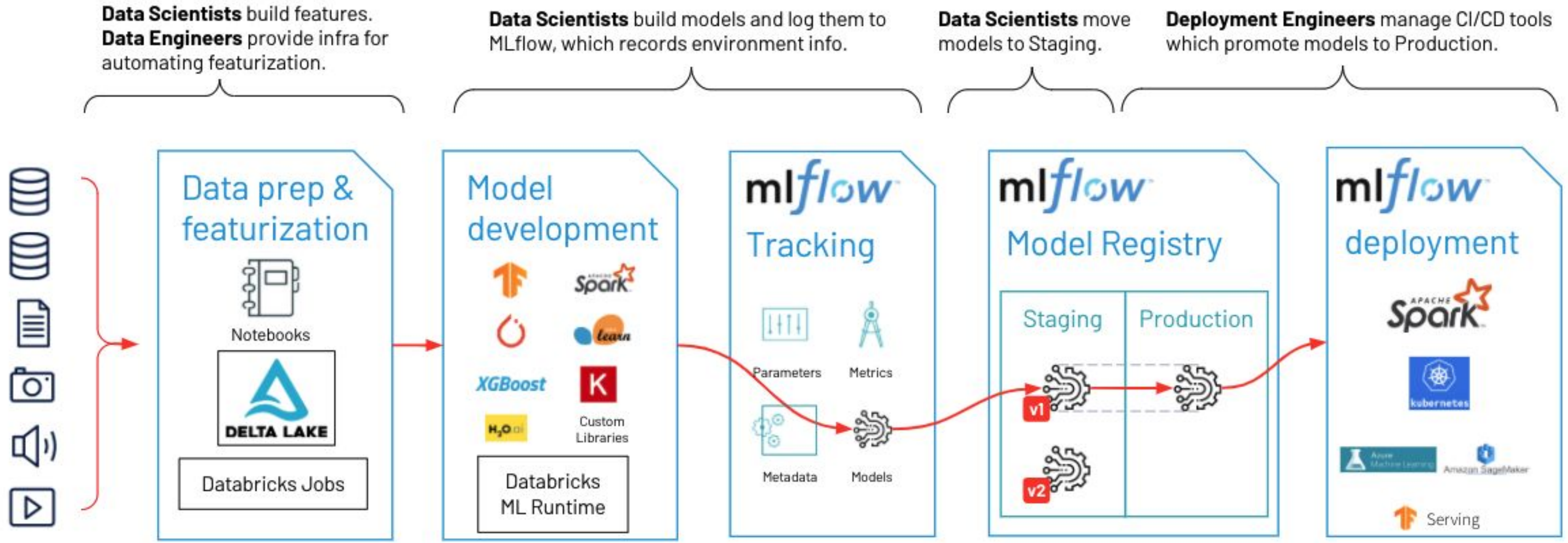
Deploy Code



Benefits of deploy code

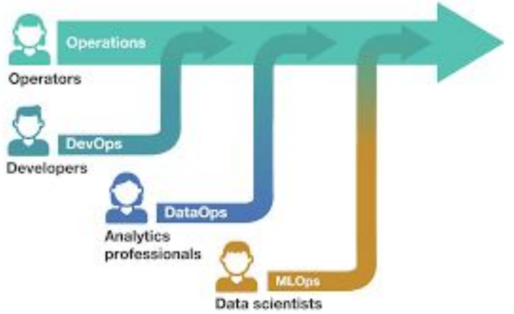
Automation	↑	Supports automated retraining in locked-down env.
Data access control	↑	Only prod env needs read access to prod training data.
Reproducible models	↑	Eng control over training env, which helps to simplify reproducibility.
Support for large projects	↑	This pattern forces the DS team to use modular code and iterative testing, which helps with coordination and development in larger projects.
Data science familiarity	↓	DS team must learn to write & hand off modular code to Eng.
Eng setup & maintenance	↓	Requires CI/CD infra for unit and integration tests, even for one-off models.

The Full ML Lifecycle



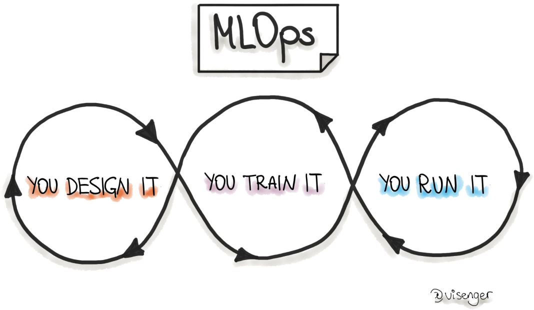
MLOps

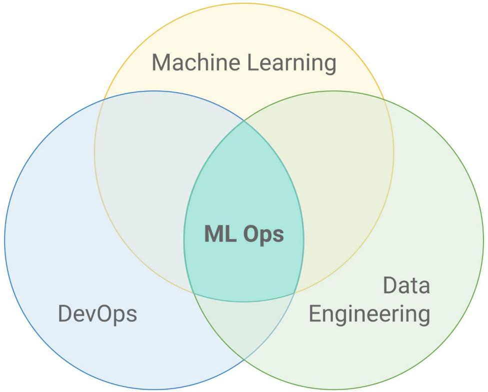
Making ML Systems Systematic, Reliable, and Repeatable

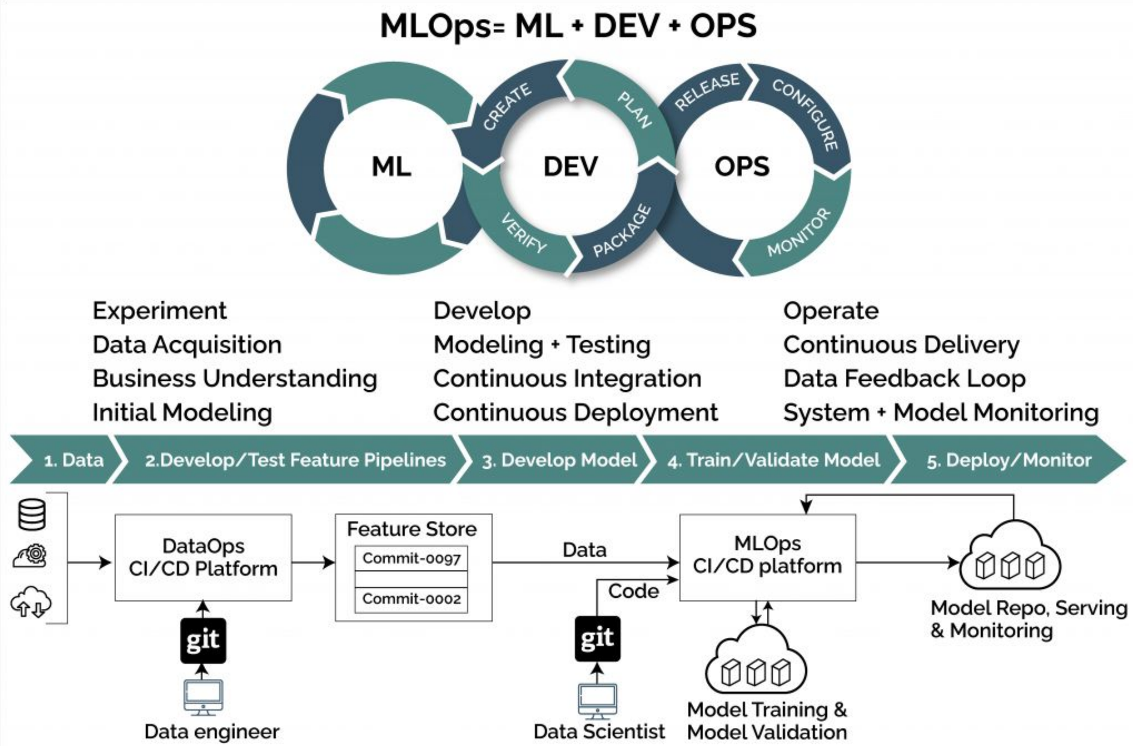


[ModelOps from Model Centric to Data Centric AI]

AI System = Code (Model/Algorithm) + Data







From Big Data to Good Data

- Good data is defined as
  - Unambiguous labels
  - Good cover of important use cases
  - Timely feedback from production data (distribution covers both data drift & concept drift)
  - Sized appropriately

Model-centric AI

How can you change the model (code) to improve performance?

Data-centric AI

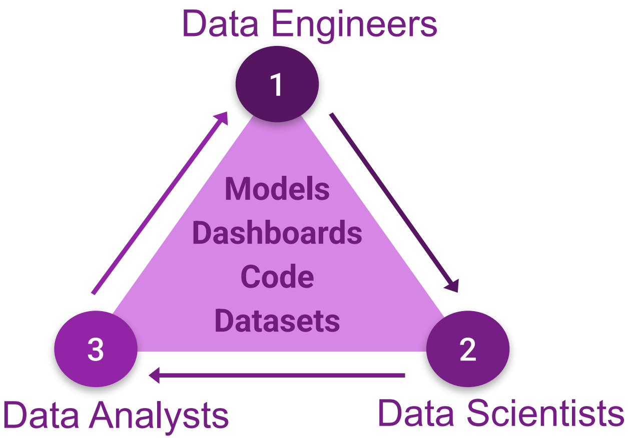
How can you systematically change your data (inputs x or labels y) to improve performance?

- Data is food for AI
  - Improving the code Vs the data

PREP	ACTION
Source and prepare high quality ingredients	Cook a meal
Source and prepare high quality data	Train a model

	Steel defect detection	Solar panel	Surface inspection
Baseline	76.2%	75.68%	85.05%
Model-centric	+0% (76.2%)	+0.04% (75.72%)	+0.00% (85.05%)
Data-centric	+16.9% (93.1%)	+3.06% (78.74%)	+0.4% (85.45%)

MLOps Progression



Beginner

- Use familiar tools
- Increase productivity
- Plan for advanced and more complex workloads

Intermediate

- Scale data & workloads
- Automate workloads & focus on reproducibility
- Unify data teams & Improve collaboration across all data & AI workloads

Advanced

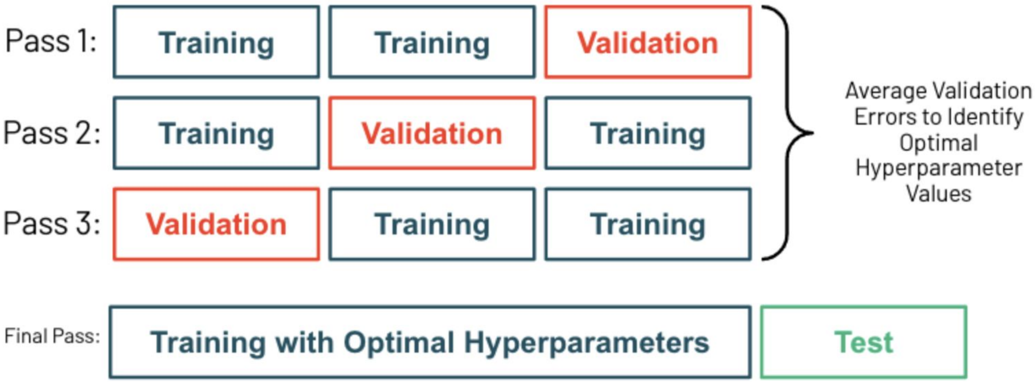
- Faster production cycles
- CI/CD & well defined processes
- E2E automation & reproducibility across multiple workloads

Model Tuning

Spark MLlib supports **grid search for hyperparameter tuning**.

Tree Depth	Number of Trees
5	2
8	4

Tree Depth	Number of Trees
5	2
5	4
8	2
8	4



K-fold Cross Validation

Avoid overfit/underfit

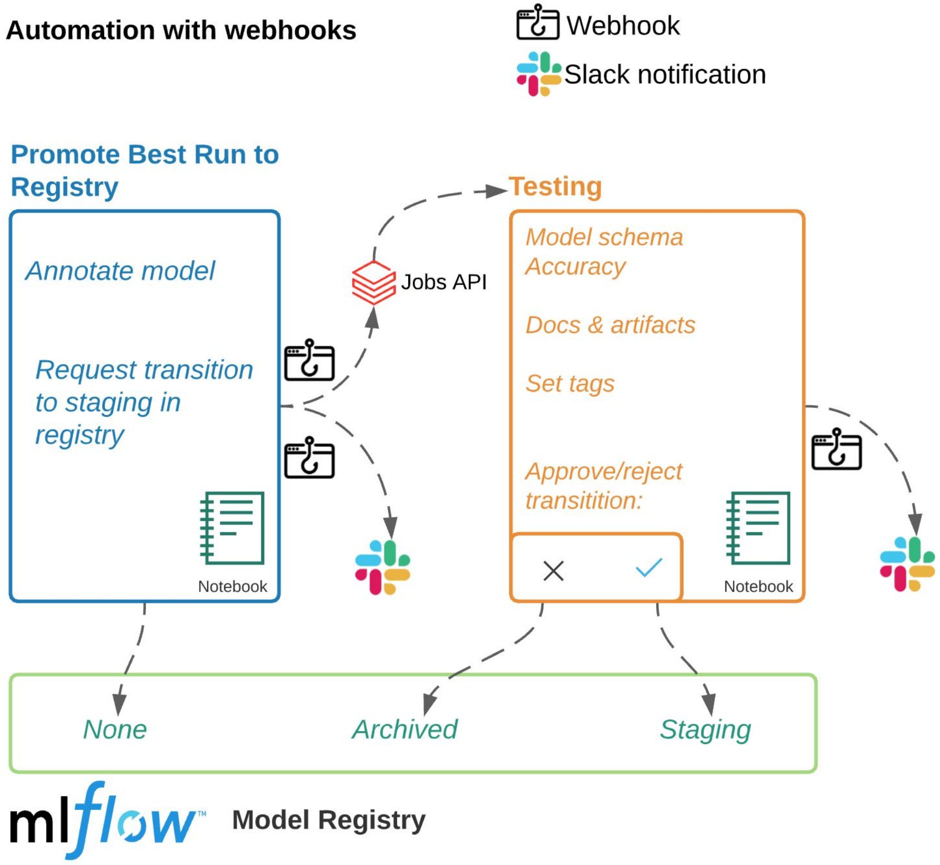
Train: Validation: Test

We need a **validation set** to assess model performance within our optimization process

This keeps us from optimizing on our **test** or holdout set.

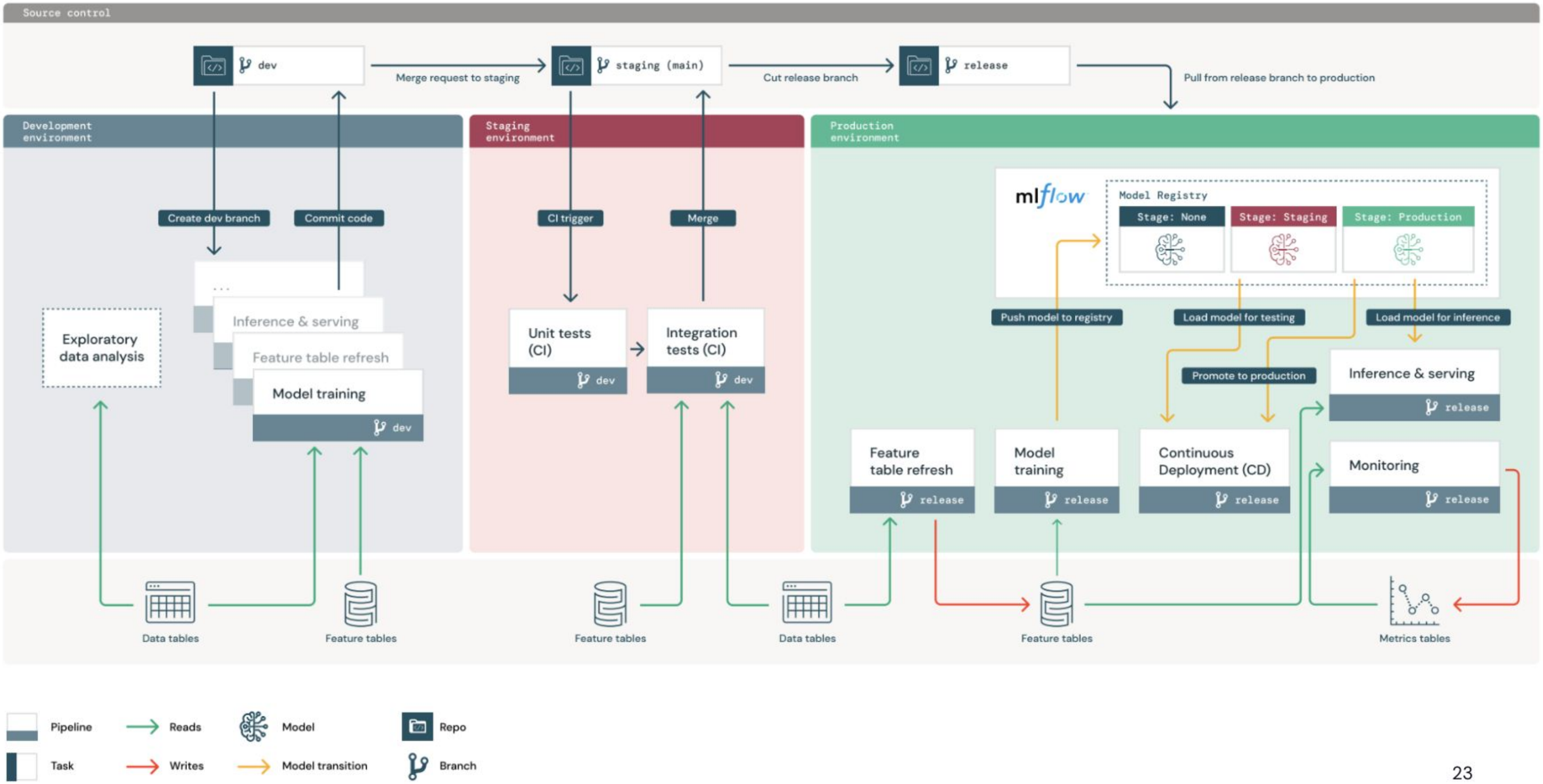
Model Selection (aka Hyperparameter Tuning)

- Manual:
    - Grid Search
- Automated:
    - Random Search
    - Bayesian
    - Genetic



Automation with webhooks
Webhook
Slack notification
Promote Best Run to Registry
Annotate model
Request transition to staging in registry
Notebook
Jobs API
Testing
Model schema
Accuracy
Docs & artifacts
Set tags
Approve/reject transitition:
Notebook
None
Archived
Staging
mlflow
Model Registry

Overview



Source control
dev
Merge request to staging
staging (main)
Cut release branch
release
Pull from release branch to production
Development environment
Create dev branch
Commit code
...
Inference & serving
Feature table refresh
Model training
dev
Exploratory data analysis
Staging environment
CI trigger
Merge
Unit tests (CI)
dev
Integration tests (CI)
dev
Production environment
mlflow
Model Registry
Stage: None
Stage: Staging
Stage: Production
Push model to registry
Load model for testing
Load model for inference
Promote to production
Inference & serving
release
Feature table refresh
release
Model training
release
Continuous Deployment (CD)
release
Monitoring
release
Data tables
Feature tables
Feature tables
Data tables
Feature tables
Metrics tables
Pipeline
Task
Reads
Writes
Model
Model transition
Repo
Branch

Solution Accelerators

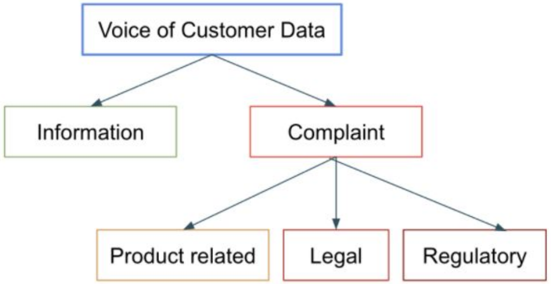
E2E ML Example

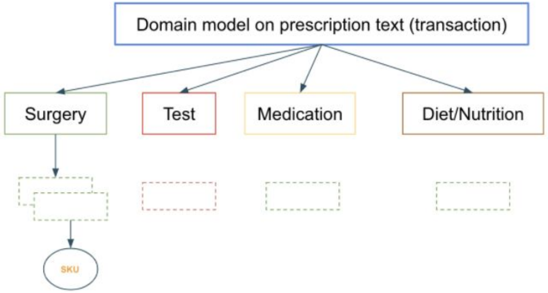
https://databricks.com/solutions#by-industry

https://databricks.com/solutions/accelerators

The Big Book of Machine Learning Use Case.pdf
https://databricks.com/p/ebook/big-book-of-machine-learning-use-cases

Model Ensembles





Single Best Model Vs Ensemble of Models
Champion & Challenger models

Link

Multiple models do not necessarily mean an ensemble!

A layering approach where moderately performant un-correlated models are combined to produce a supermodel that improves accuracy while improving stability and is often a divide and conquer strategy used in large, diverse datasets.

- Fetch the "best" models of each architecture type
- Build a custom pyfunc model class that encapsulates the best models
- Provide a predict function for the ensemble
- **Provide a voting function**
- Package and log the model in MLflow as a custom pyfunc model
- Scoring