DADS7305: MLOPs

Northeastern University

Instructor: Ramin Mohammadi

September 7, 2025

These materials have been prepared and sourced for the course **MLOPs** at Northeastern University. Every effort has been made to provide proper citations and credit for all referenced works.

If you believe any material has been inadequately cited or requires correction, please contact me at:

`r.mohammadi@northeastern.edu`

Thank you for your understanding and collaboration.

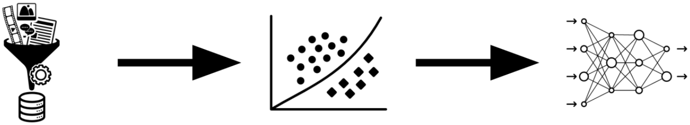
Data Journey and Data Storage

Data Journey

Outline

- The data journey
- Accounting for data and model evolution
- Intro to ML metadata
- Using ML metadata to track changes

The data journey



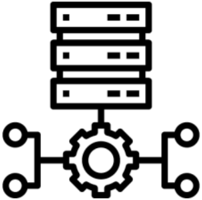
Raw features and labels

Input-output map

ML model to learn mapping

Data Transformation

- Data transforms as it flows through the process
- Interpreting model results requires understanding data transformation



Artifacts and the ML pipeline

- Artifacts are created as the components of the ML pipeline execute
- Artifacts include all of the data and objects which are produced by the pipeline components
- This includes the data, in different stages of transformation, the schema, the model itself, metrics, etc.

Data provenance and lineage

- The chain of transformations that led to the creation of a particular artifact.
- Important for debugging and reproducibility.

Data provenance: Why it matters

Helps with debugging and understanding the ML pipeline:

Inspect artifacts at each point in the training process

Trace back through a training run

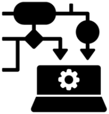
Compare training runs

Data lineage: data protection regulation

- Organizations must closely track and organize personal data
- Data lineage is extremely important for regulatory compliance

Data provenance: Interpreting results

Data transformations sequence leading to predictions



Understanding the model as it evolves through runs

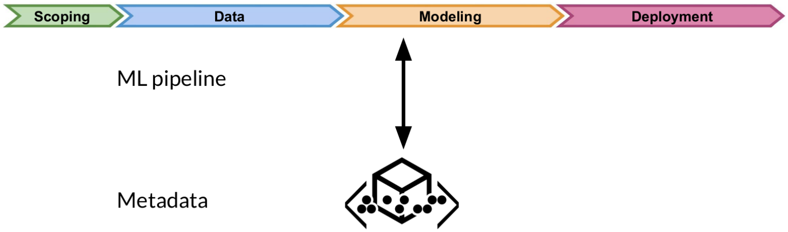
Data versioning

- Data pipeline management is a major challenge
- Machine learning requires reproducibility
- Code versioning: GitHub and similar code repositories
- Environment versioning: Docker, Terraform, and similar
- Data versioning:
  - Version control of datasets
  - Examples: DVC, Git-LFS

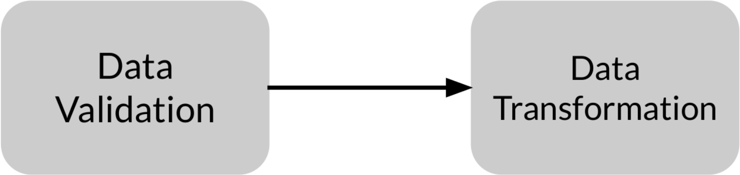
Data Journey and Data Storage

Intro to ML Metadata

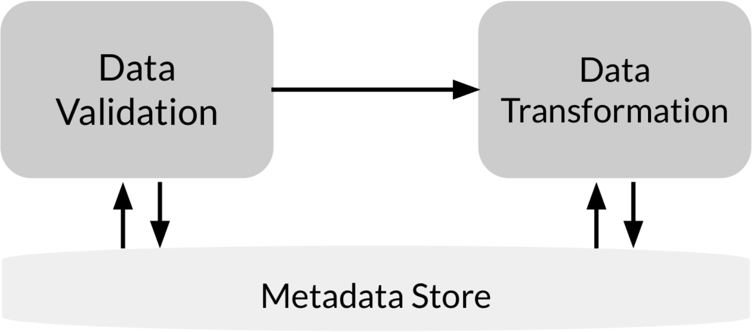
Metadata: Tracking artifacts and pipeline changes



Ordinary ML data pipeline

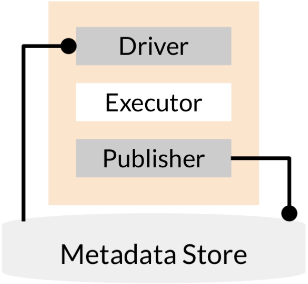


Metadata: Tracking progress



Metadata: TFX component architecture

- **Driver:**
  - Supplies required metadata to executor
- **Executor:**
  - Place to code the functionality of component
- **Publisher:**
  - Stores result into metadata



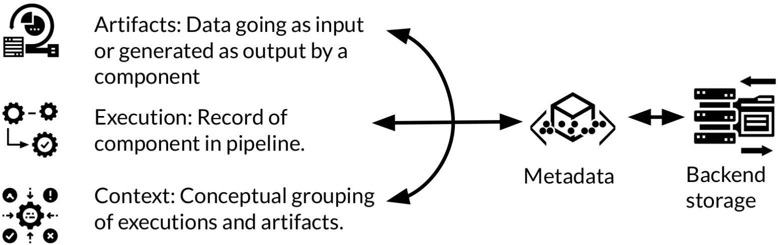
ML Metadata library

- Tracks metadata flowing between components in pipeline
- Supports multiple storage backends

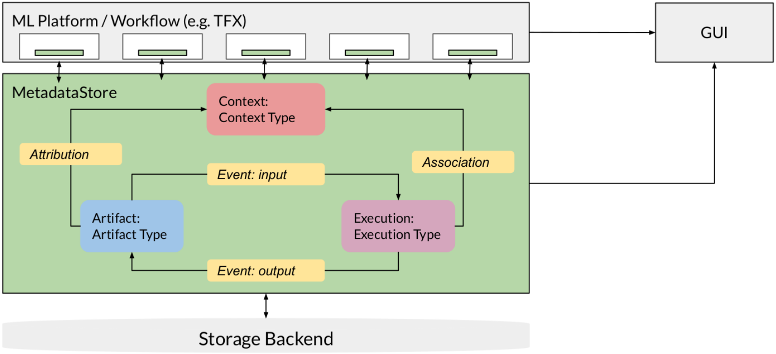
ML Metadata terminology

Units	Types	Relationships
Artifact	ArtifactType	Event
Execution	ExecutionType	Attribution
Context	ContextType	Association

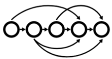
Metadata stored



Inside MetadataStore



Other benefits of ML Metadata



Produce DAG of pipelines

Verify the inputs used in an execution

List all artifacts

Compare artifacts

Key points

- ML metadata:
  - Architecture and nomenclature
  - Tracking metadata flowing between components in pipeline

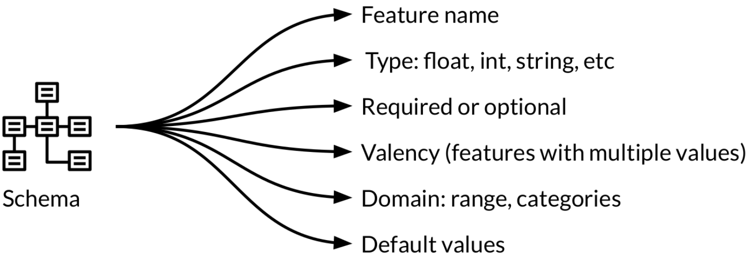
Evolving Data

Schema Development

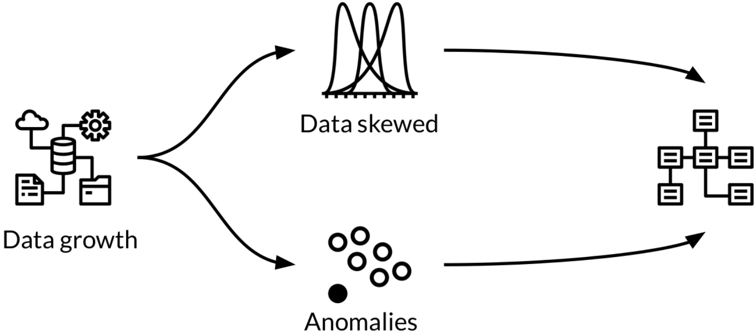
Outline

- Develop enterprise schema environments
- Iteratively generate and maintain enterprise data schemas

Review: Recall Schema



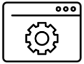
Iterative schema development and evolution



Reliability during data evolution

Platform needs to be resilient to disruptions from:

Inconsistent data



Software

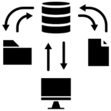
User configurations

Execution environments

Scalability during data evolution

Platform must scale during:

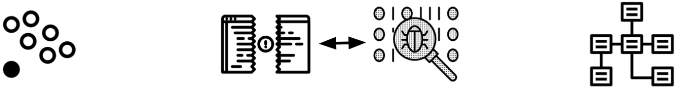
High data volume during training



Variable request traffic during serving

Anomaly detection during data evolution

Platform designed with these principles:

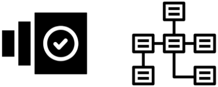


Easy to detect anomalies

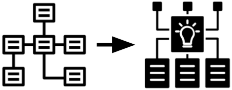
Data errors treated same as code bugs

Update data schema

Schema inspection during data evolution



Looking at schema versions to
track data evolution

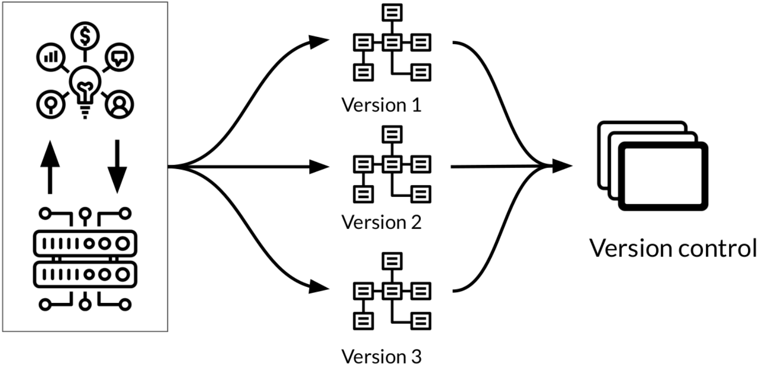


Schema can drive other
automated processes

Evolving Data

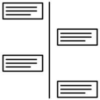
Schema Environments

Multiple schema versions

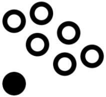


Maintaining varieties of schema

Business use-case needs to support data from different sources.



Data evolves rapidly



Is anomaly part of accepted type of data?

Anomaly: No labels in serving dataset

Feature name	Anomaly short description	Anomaly long description
'Cover_Type'	Out-of-range values	Unexpectedly small value: 0.

Schema environments

- Customize the schema for each environment
- Example: Add or remove label in schema based on type of dataset

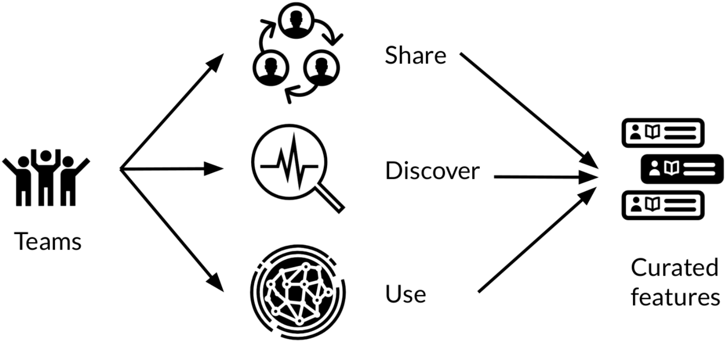
Key points

- Iteratively update and fine-tune schema to adapt to evolving data
- How to deal with scalability and anomalies
- Set schema environments to detect anomalies in serving requests

Enterprise Data Storage

Feature Stores

Feature stores



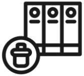
Feature stores

Many modeling problems use identical or similar features

Feature stores

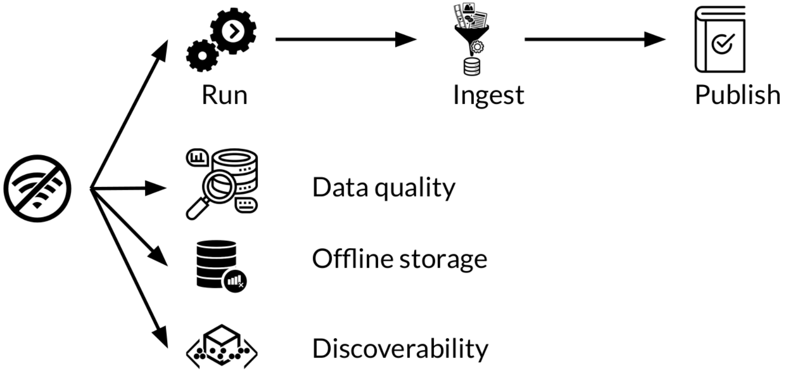
Avoid duplication

Control access

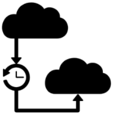


Purge

Offline feature processing



Offline feature usage



Low latency access
to features

Features difficult
to compute online

Precompute and
store for low
latency access

Features for online serving - Batch

Batch precomputing

Loading history

- Simple and efficient
- Works well for features to only be updated every few hours or once a day
- Same data is used for training and serving

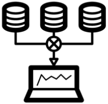
Feature store: key aspects

- Managing feature data from a single person to large enterprises
- Scalable and performant access to feature data in training and serving
- Provide consistent and point-in-time correct access to feature data
- Enable discovery, documentation, and insights into your features

Enterprise Data Storage

Data Warehouse

Data Warehouse

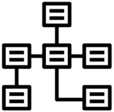


Aggregates
data sources

Processed
and analyzed

Read
optimized

Not
real time

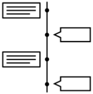


Follows
schema

Key features of data warehouse

Subject oriented

Integrated



Non volatile

Time variant

Data Warehouse

Enhanced ability to analyze data

Timely access to data

Enhanced data quality and consistency

High return on investment

Increased query and system performance

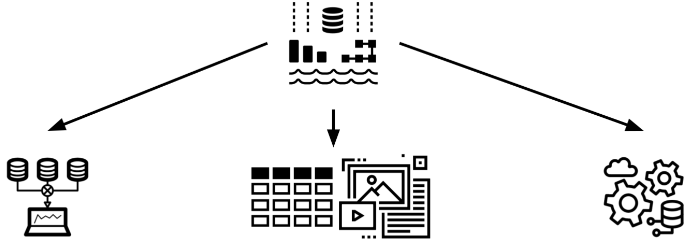
Comparison with databases

Data warehouse	Database
Online analytical processing (OLAP)	Online transactional processing (OLTP)
Data is refreshed from source systems	Data is available real-time
Stores historical and current data	Stores only current data
Data size can scale to ¿= terabytes	Data size can scale to gigabytes
Queries are complex, used for analysis	Queries are simple, used for transactions
Queries are long running jobs	Queries executed almost in real-time
Tables need not be normalized	Tables normalized for efficiency

Enterprise Data Storage

Data Lakes

Data Lakes



Aggregates raw data from one or more sources

Data can be structured or unstructured

Doesn't involve any processing before writing data

Comparison with data warehouse

	Data warehouses	Data lakes
Data Structure	Processed	Raw
Purpose of data	Currently in use	Not yet determined
Users	Business professionals	Data scientists
Accessibility	More complicated and costly to make changes	Highly accessible and quick to update

LLM Data Storage

Vector Stores

Vector stores: foundation of retrieval

- Vector stores are specialized databases optimized for similarity search over high-dimensional vectors
- Used in Retrieval-Augmented Generation (RAG) to find contextually relevant documents or chunks
- Store embeddings derived from text, images, or other modalities
- Enable fast, approximate nearest-neighbor (ANN) search

How vector stores are used in LLM pipelines

- Input documents are chunked and embedded using a model (e.g., SentenceTransformers, OpenAI, LLaMA)
- Embeddings are stored in a vector database alongside metadata
- At query time, the user input is embedded and matched against stored vectors
- Matched chunks are retrieved and passed to the LLM for context-aware generation

Popular vector databases

- **FAISS** – Facebook's open-source library for efficient similarity search; supports CPU/GPU
- **Pinecone** – Fully managed vector DB with metadata filtering and scaling
- **Weaviate** – Open-source, schema-aware, supports hybrid search (text + vector)
- **Chroma** – Lightweight, fast local vector store; ideal for prototyping

Key considerations for vector store design

- **Scalability** – Can it handle millions of vectors efficiently?
- **Index type** – Flat, HNSW, IVF, PQ — tradeoff between speed and accuracy
- **Filtering** – Support for metadata filtering alongside vector similarity
- **Latency** – Query performance matters in production LLM applications
- **Freshness** – How easily can new documents be added or deleted?

Labs for This Week

Objective

Briefly describe the learning goal for this week's lab(s).

Lab Activities:

- Lab 1: [MLMD] — [MLMD Tutorial]
- Lab 2: [TFX] — [TFX Tutorial]

Submission Deadline: [Before the next class]

- Assignment 5: [MLMD] — [Create a metadata store of your choice]
- Assignment 5: [TFX] — [Create a TFX pipeline of your choice]

Reading Materials

This Week's Theme

Topic focus: [People + AI Guidebook - Data Collection + Evaluation.pdf]
You should use the worksheet related to this pdf to your project and submit it when its requested.

Required Readings:

- [On the Reliable Detection of Concept Drift from Streaming Unlabeled Data]

Be prepared to discuss highlights and open questions in class.

Sources

- DeepLearning.AI
- The People + AI Guidebook