DADS7305: MLOPs

Northeastern University

Instructor: Ramin Mohammadi

February 10, 2026

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.

Collecting, Labeling, and Validating Data

Overview

The importance of data

"Data is the hardest part of ML and the most important piece to get right... Broken data is the most common cause of problems in production ML systems"

- Scaling Machine Learning at Uber with Michelangelo - Uber

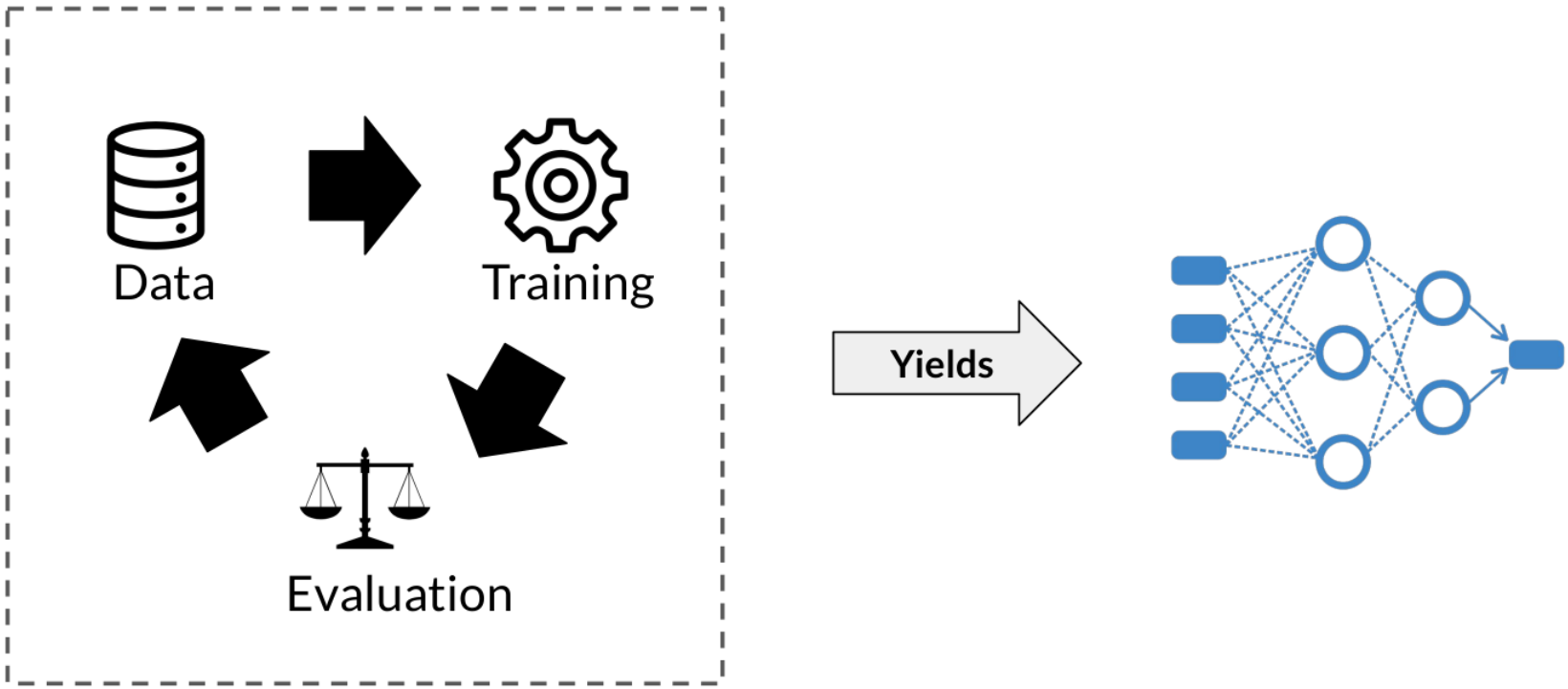
"No other activity in the machine learning life cycle has a higher return on investment than improving the data a model has access to."

- Feast: Bridging ML Models and Data - Gojek

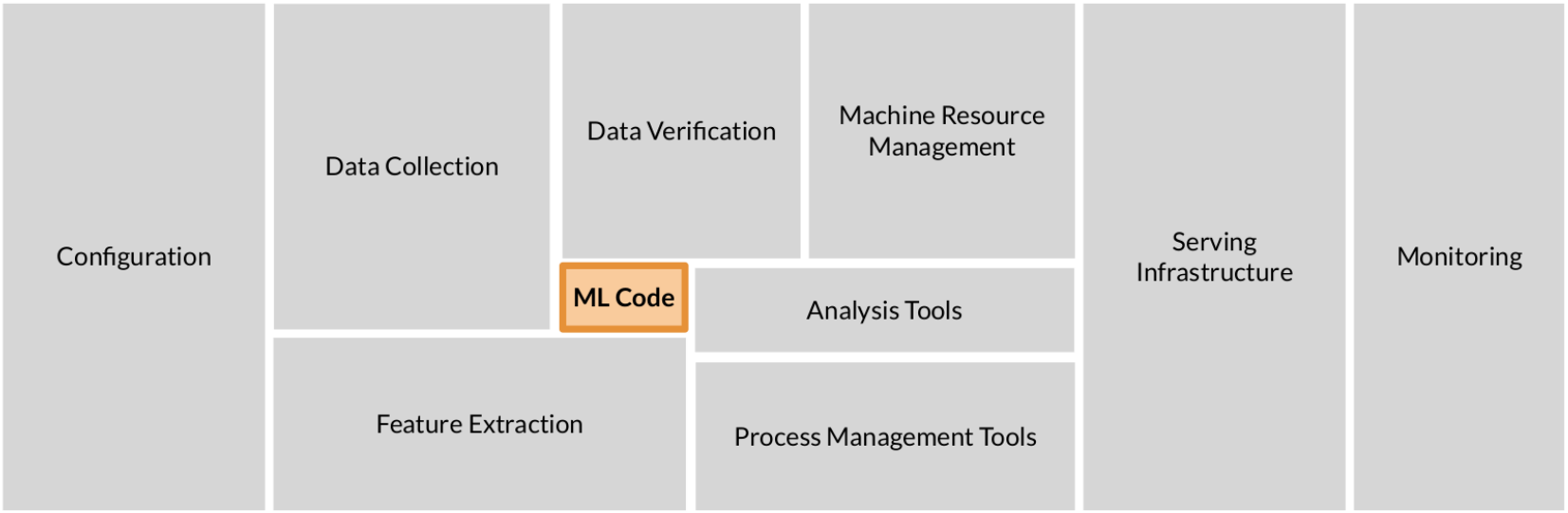
Introduction to Machine Learning Engineering for Production

Overview

Data iteration loop



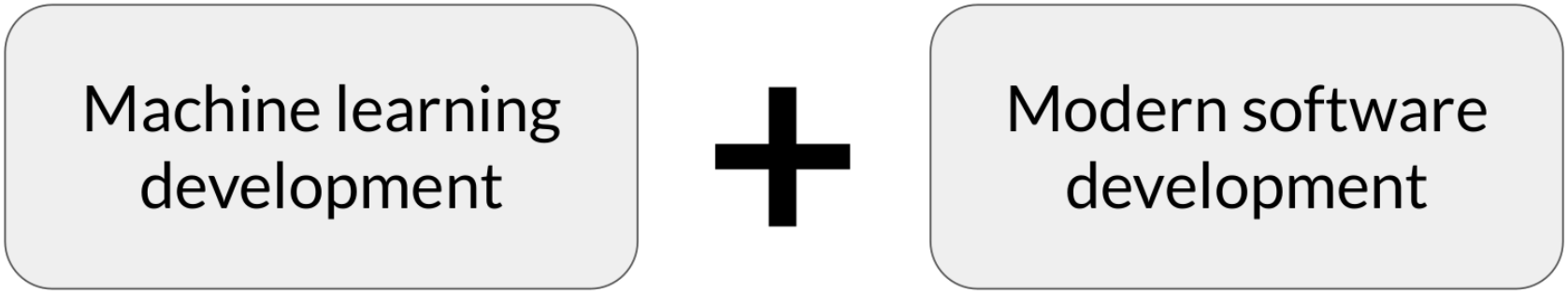
Production ML systems require so much more



ML modeling vs production ML

	Academic/Research ML	Production ML
Data	Static	Dynamic - Shifting
Priority for design	Highest overall accuracy	Fast inference, good interpretability
Model training	Optimal tuning and training	Continuously assess and retrain
Fairness	Very important	Crucial
Challenge	High accuracy algorithm	Entire system

Production machine learning



Data Quality

- Labeling
- Feature space coverage
- Minimal dimensionality
- Maximum predictive data
- Fairness
- Rare conditions

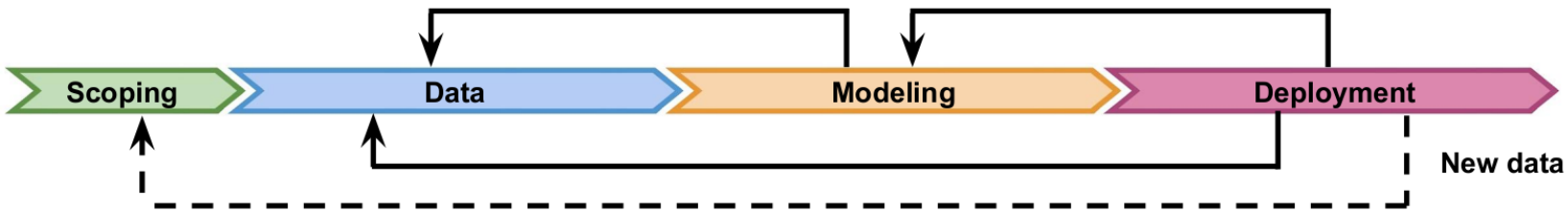
Introduction to Machine Learning Engineering for Production

ML Pipelines

Outline

- ML Pipelines
- Directed Acyclic Graphs and Pipeline Orchestration Frameworks
- Intro to TensorFlow Extended (TFX)

ML pipelines



Infrastructure for
automating, monitoring, and maintaining
model training and deployment

Modern software development

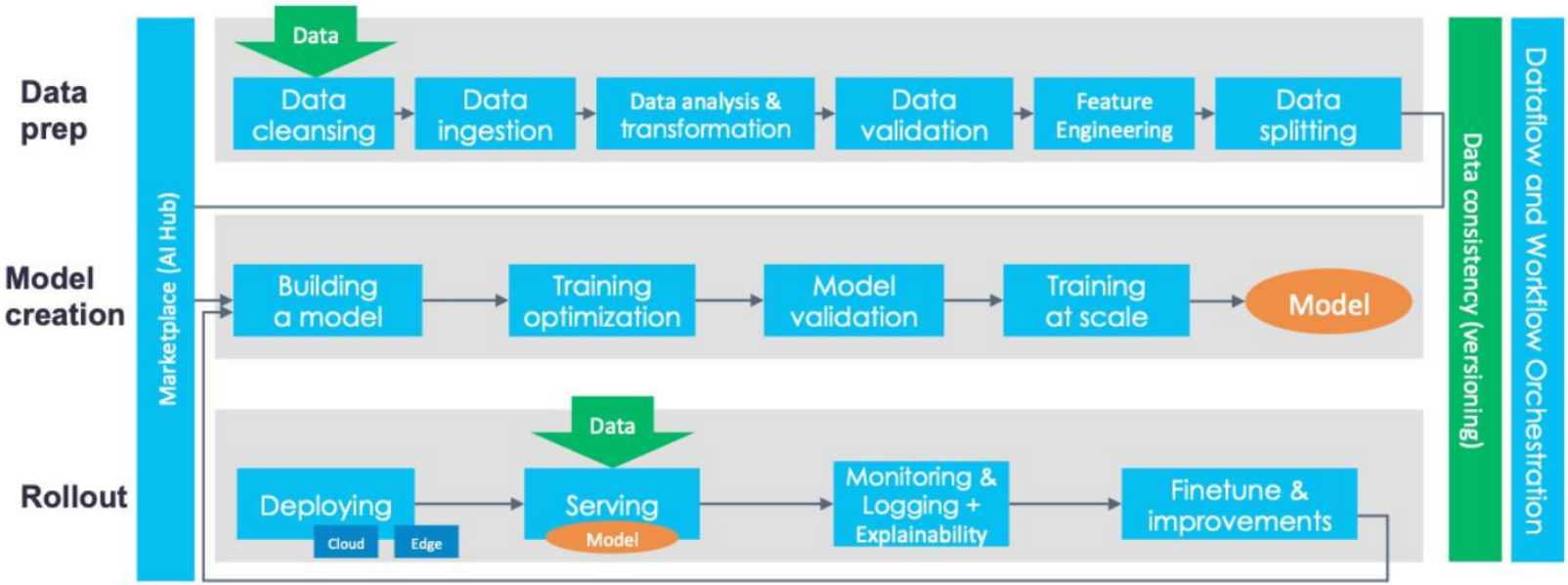
Accounts for:

- Scalability
- Extensibility
- Configuration
- Consistency & reproducibility
- Safety & security
- Modularity
- Testability
- Monitoring
- Best practices

- Build integrated ML systems
- Continuously operate it in production
- Handle continuously changing data
- Optimize compute resource costs

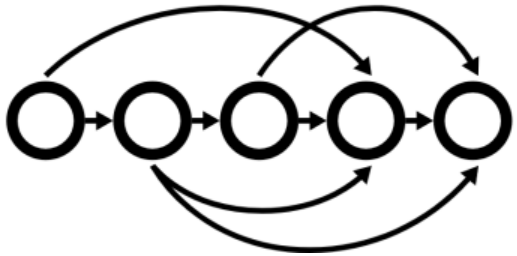
Production ML infrastructure

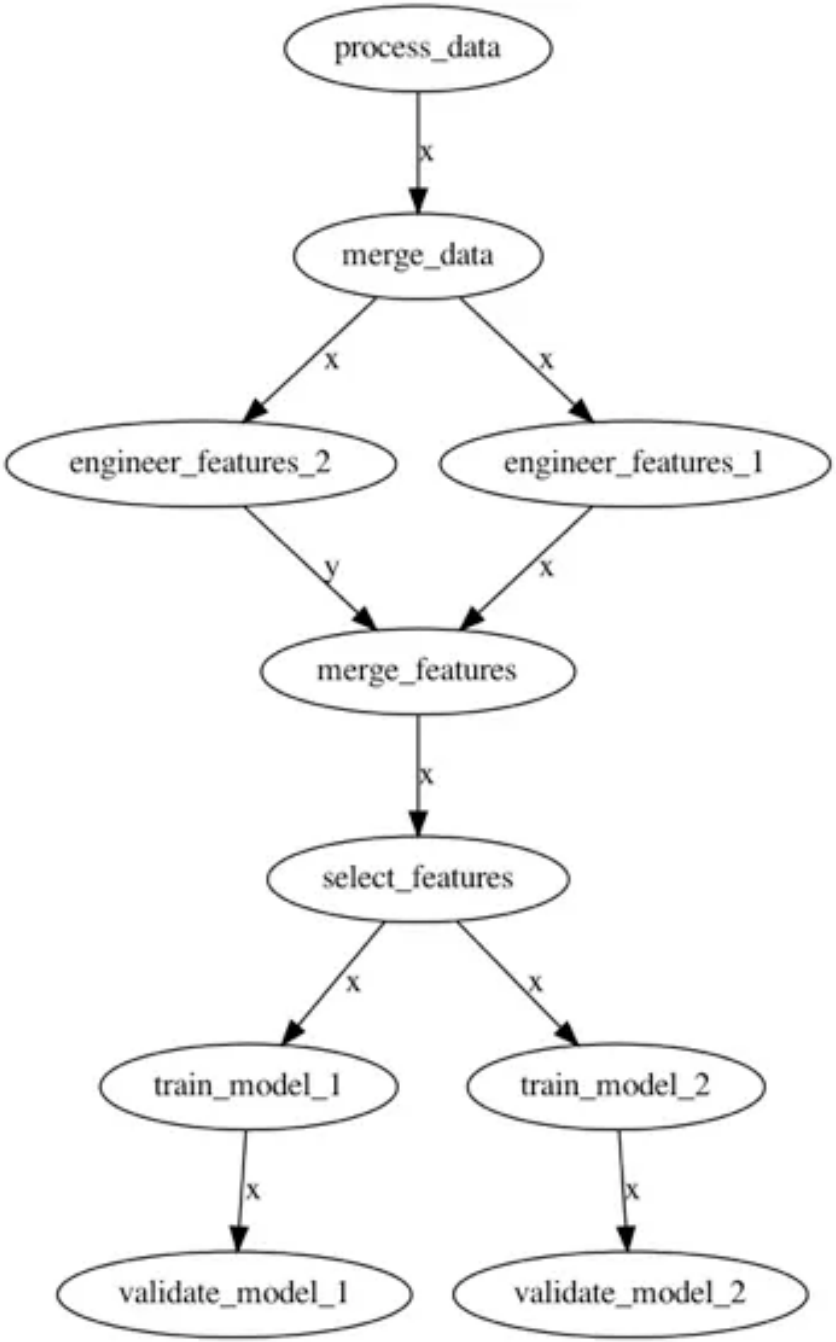
CD Foundation MLOps reference architecture



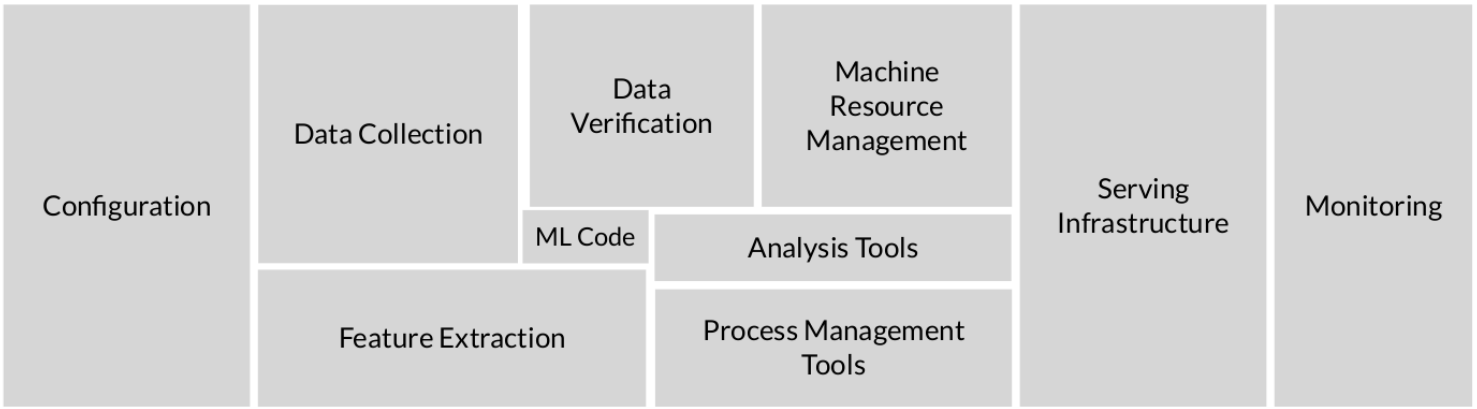
Directed acyclic graphs

- A directed acyclic graph (DAG) is a directed graph that has no cycles
- ML pipeline workflows are usually DAGs
- DAGs define the sequencing of the tasks to be performed, based on their relationships and dependencies.




process_data
x
merge_data
x
x
engineer_features_2
engineer_features_1
y
x
merge_features
x
select_features
x
x
train_model_1
train_model_2
x
x
validate_model_1
validate_model_2

Pipeline orchestration frameworks



- Responsible for scheduling the various components in an ML pipeline DAG dependencies
- Help with pipeline automation
- Examples: Airflow, Argo, Celery, Luigi, Kubeflow

TensorFlow Extended (TFX)

End-to-end platform for deploying production ML pipelines

Sequence of components that are designed for scalable, high-performance machine learning tasks

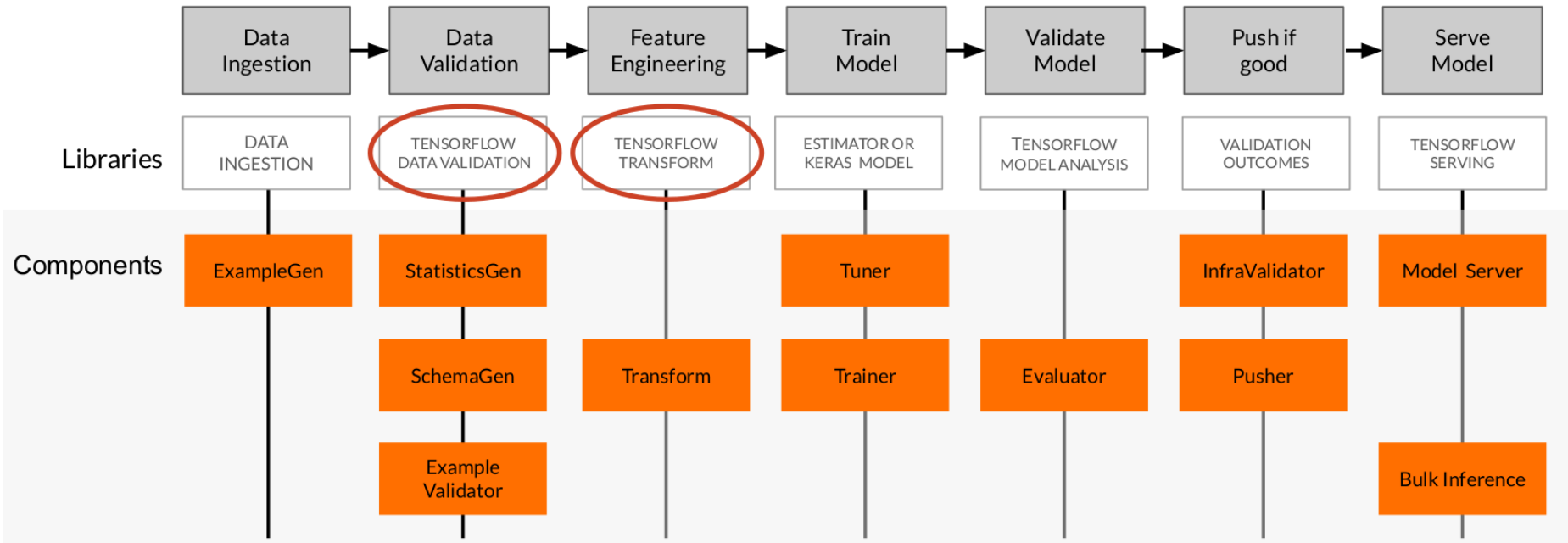
MLOps orchestration tools

Tool	Description	Strengths
Kubeflow Pipelines	Based on Kubernetes, great for orchestration of ML workflows	Cloud-native, powerful with Vertex AI & K8s
MLflow (Databricks)	Tracking, packaging, model registry	Widely adopted in industry, model versioning
Metaflow (Netflix)	Human-centric, Pythonic MLOps	Great for quick iteration and reproducibility
ZenML	MLOps framework with modern plugins (e.g., LangChain, HuggingFace, etc.)	Focused on modern use cases like GenAI, easy plugin system
Airflow + Custom DAGs	Used to manually build MLOps pipelines	Flexible, integrates with everything, not ML-specific

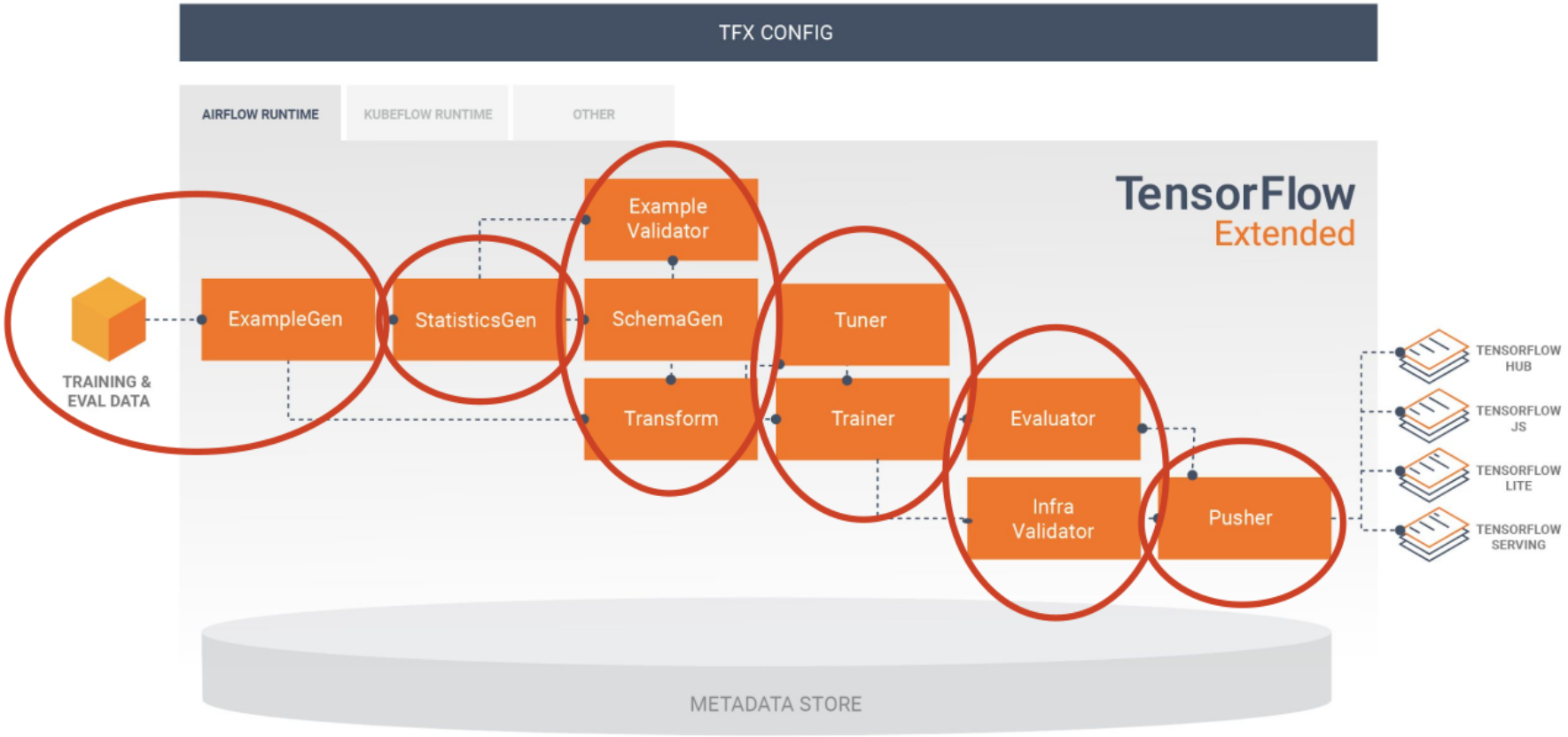
LLM/RAG tooling overview

Tool	Description	Use Cases
LangChain	Framework for building agents and RAG pipelines	Agent orchestration, tool use, chains
LangGraph	State-machine based orchestration for agents (built on LangChain)	Long-lived agents, memory, branching logic
LlamaIndex	Data indexing and retrieval for LLM pipelines	Data preprocessing, vector stores
Haystack	NLP/QA pipeline builder, now LLM-compatible	Modular components, good for RAG
Guardrails AI	Validates and controls LLM outputs using schemas and prompts	Alignment, output control, RAG + eval
DSPy	Programmatic prompt compiler and optimization framework	LLM pipeline optimization, output structure

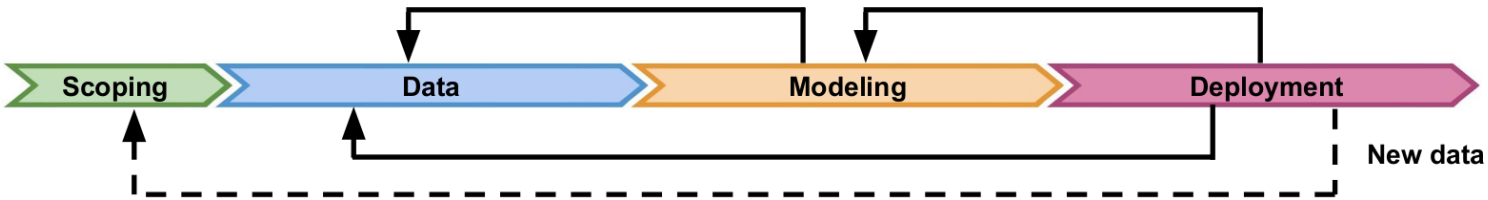
TFX production components



TFX Hello World



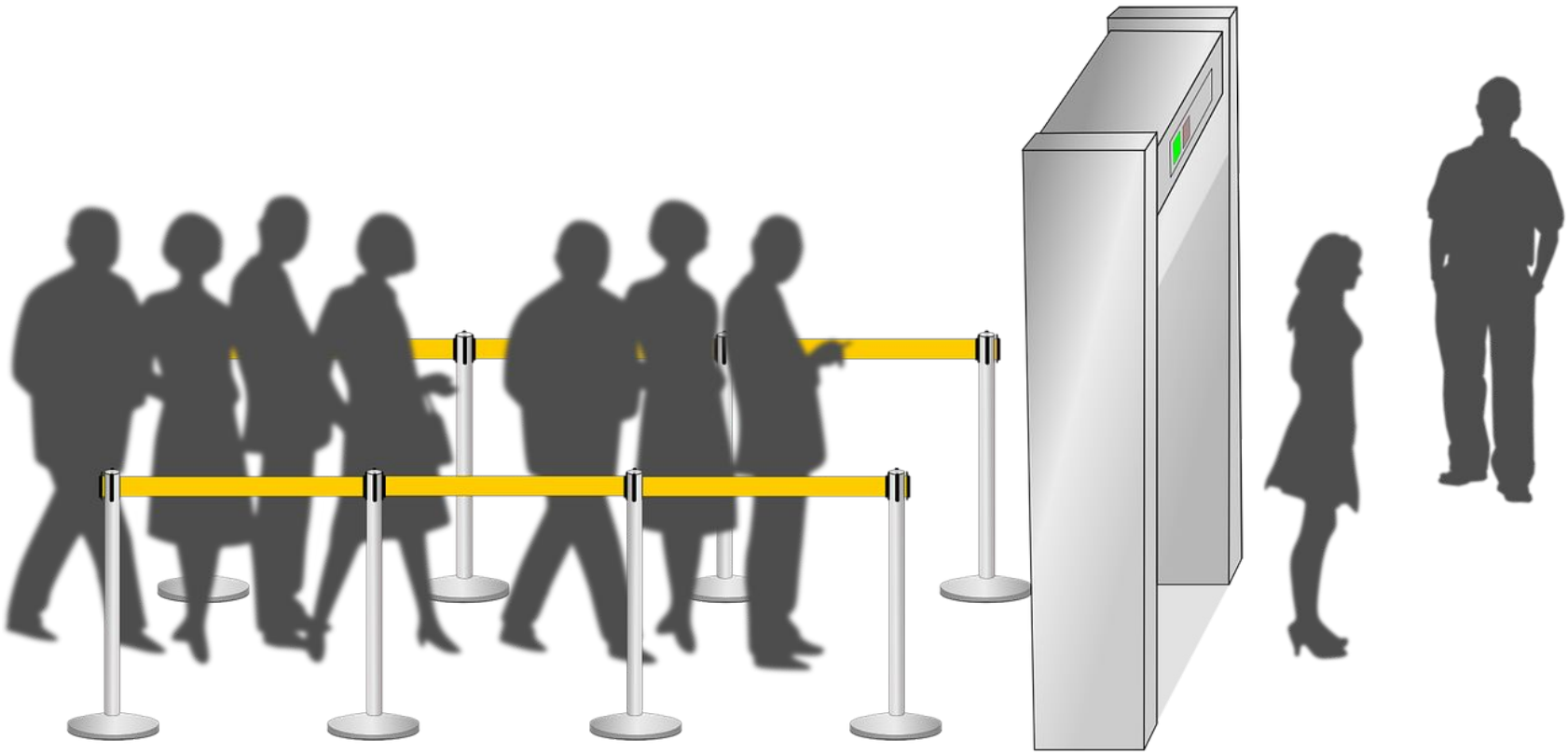
Key points



- Production ML pipelines: automating, monitoring, and maintaining end-to-end processes
- Production ML is much more than just ML code
  - ML development + software development
- TFX is an open-source end-to-end ML platform

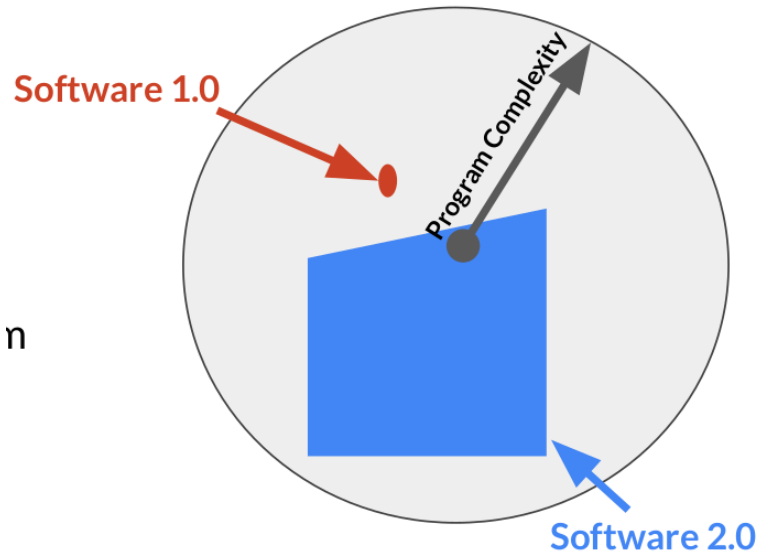
Collecting Data

Importance of Data



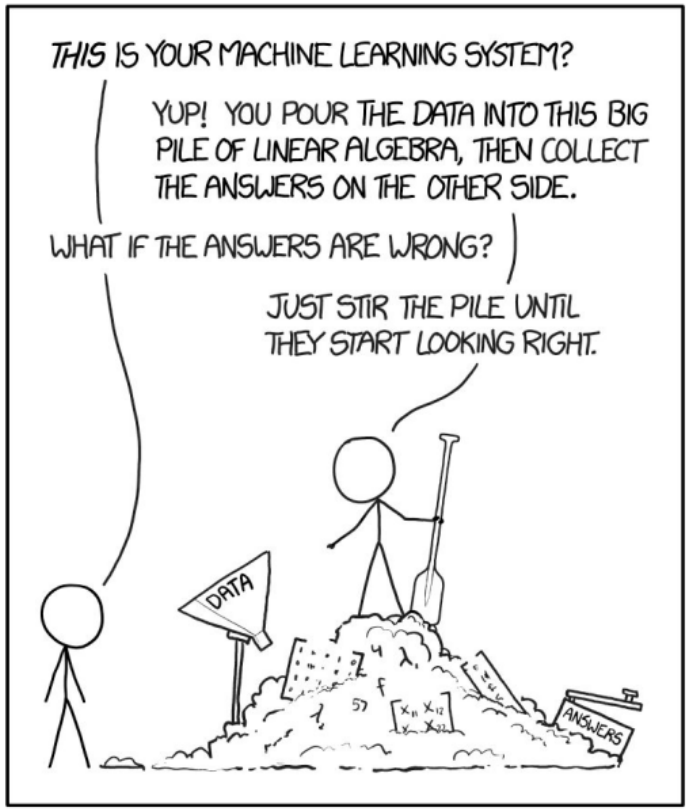
ML: Data is a first class citizen

- **Software 1.0**
  - Explicit instructions to the computer
- **Software 2.0**
  - Specify some goal on the behavior of a program
  - Find solution using optimization techniques
  - Good data is key for success
  - Code in Software = Data in ML

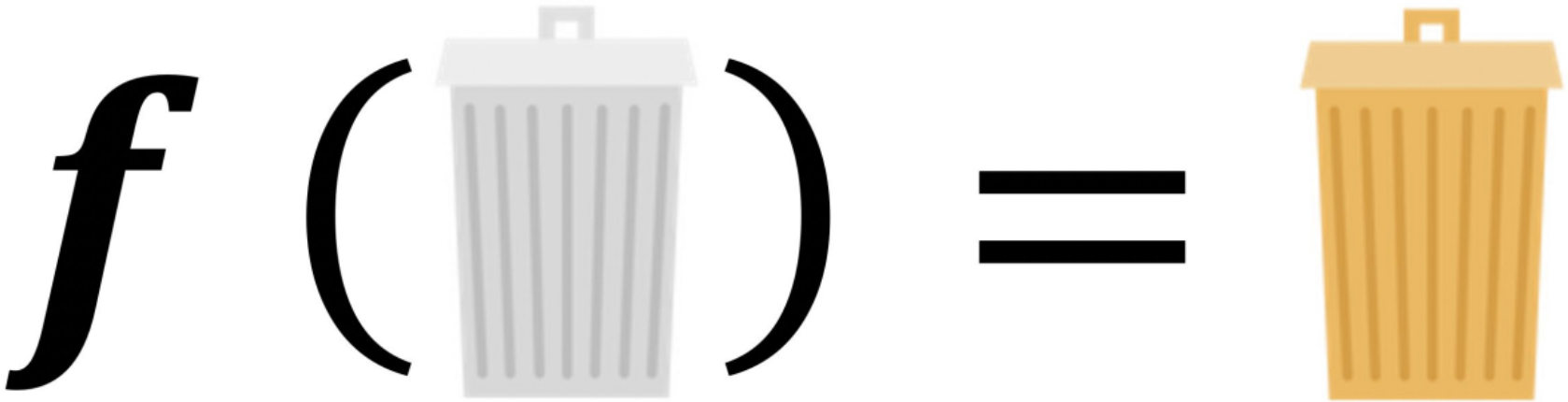


Everything starts with data

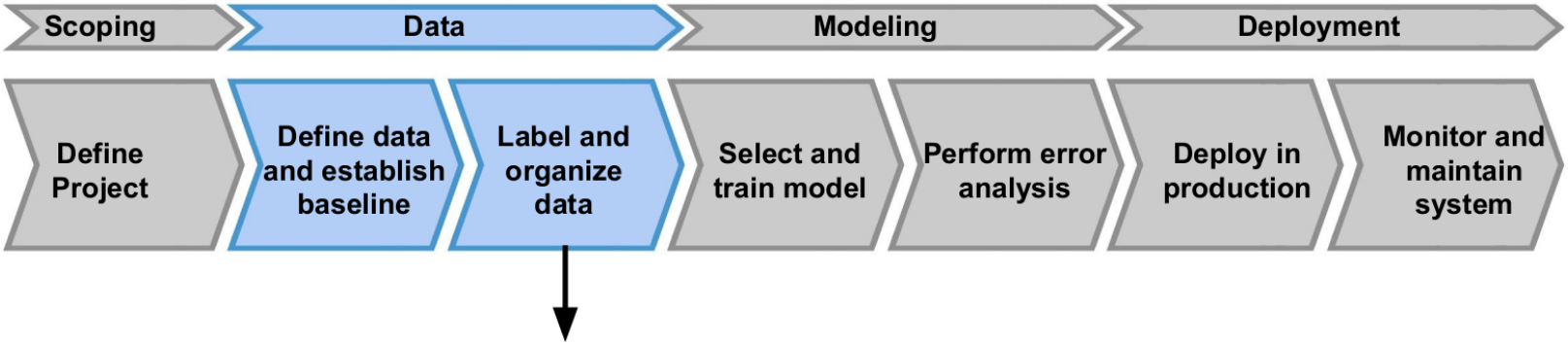
- Models aren't magic
- Meaningful data:
  - maximize predictive content
  - remove non-informative data
  - feature space coverage



Garbage in, garbage out

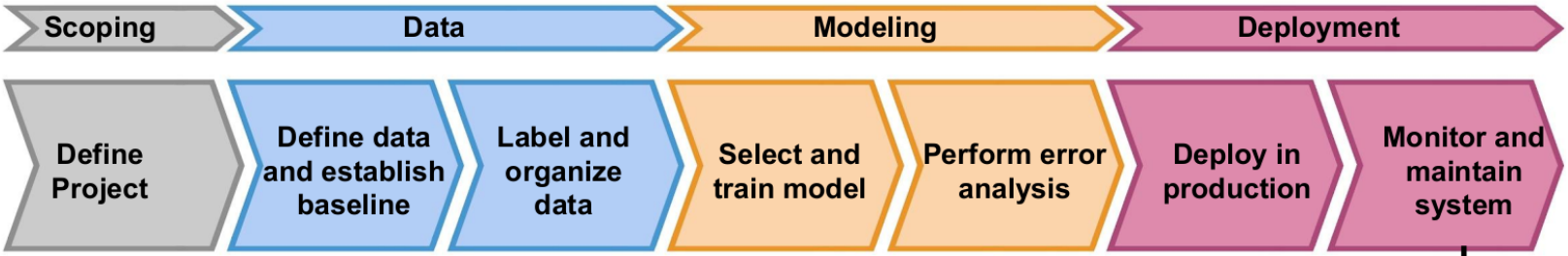


Data pipeline



- Data collection
- Data ingestion
- Data formatting
- Feature engineering
- Feature extraction

Data collection and monitoring



- Downtime
- Errors
- Distributions shifts
- Data failure
- Service failure

Key Points

- Understand users, translate user needs into data problems
- Ensure data coverage and high predictive signal
- Source, store and monitor quality data responsibly

Collecting Data

Example Application: Suggesting Runs

Key considerations

Users	Runners
User Need	Run more often
User Actions	Complete run using the app
ML System Output	• What routes to suggest • When to suggest them
ML System Learning	• Patterns of behaviour around accepting run prompts • Completing runs • Improving consistency

Example dataset

		FEATURES				
	Runner ID	**Run**	**Runner Time**	**Elevation**	**Fun**	
EXAMPLES	AV3DE	Boston Marathon	03:40:32	1,300 ft	Low	**LABELS**
	X8KGF	Seattle Oktoberfest 5k	00:35:40	0 ft	High	
	BH9IU	Houston Half-marathon	02:01:18	200 ft	Medium	

Get to know your data

- Identify data sources
- Check if they are refreshed
- Consistency for values, units, & data types
- Monitor outliers and errors

Dataset issues

- Inconsistent formatting
  - Is zero “0”, “0.0”, or an indicator of a missing measurement
- Compounding errors from other ML Models
- Monitor data sources for system issues and outages

Measure data effectiveness

- Intuition about data value can be misleading
  - Which features have predictive value and which ones do not?
- Feature engineering helps to maximize the predictive signals
- Feature selection helps to measure the predictive signals

Translate user needs into data needs

Data Needed	▶ Running data from the app ▶ Demographic data ▶ Local geographic data

Translate user needs into data needs

Features Needed	
	▶ Runner demographics ▶ Time of day ▶ Run completion rate ▶ Pace ▶ Distance ran ▶ Elevation gained ▶ Heart rate

Translate user needs into data needs

Labels Needed	▶ Runner acceptance or rejection of app suggestions ▶ User generated feedback regarding why suggestion was rejected ▶ User rating of enjoyment of recommended runs

key Points

- Understand your user, translate their needs into data problems
  - What kind of / how much data is available
  - What are the details and issues of your data
  - What are your predictive features
  - What are the labels you are tracking
  - What are your metrics

Collecting Data

Data for Your LLM Work

Before using an LLM

- Identify and collect the right data for your task (fine-tuning dataset or domain-specific documents/context for prompting).
- Considerations include domain relevance, sources, and quality vs. quantity.

Domain Relevance

- Use data that is relevant to your use-case.
- Example: if building an assistant for legal questions, gather legal text documents.
- High-quality, domain-specific data will yield more accurate and useful model outputs.

Data Sources

- Plain text files, PDFs, websites, databases, APIs, etc.
- LLM applications often involve unstructured data (free-form text) that may require extraction from various formats (PDFs, Word docs, HTML pages).

Quantity vs Quality

- It's not just about having a lot of data-quality matters more.
- A smaller high-quality dataset often outperforms a larger noisy dataset. Invest time in cleaning and curating the data.

Synthetic Data

- You might generate data (e.g., using an LLM to create Q&A pairs) to augment your dataset.
- Validate carefully; synthetic data can introduce biases or errors.

After Data Collection: Preprocessing

- Clean and convert raw data into a format suitable for the model.
- Typical tasks: remove unwanted content, split text into segments, transform formats (e.g., PDF to text).
- If data is already plain text (`.txt`), preprocess by:
  - Cleaning whitespace, correcting encoding issues.
  - Normalizing text (lowercasing, removing special symbols) as appropriate for the use case.

PDFs and Documents

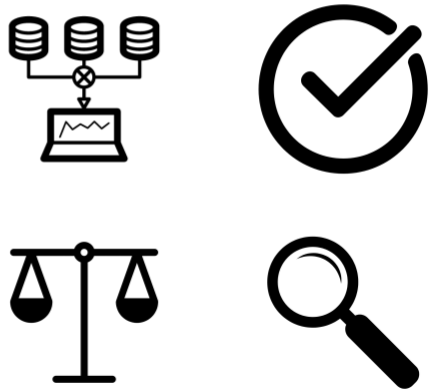
- PDFs contain layout-oriented text and can be tricky to extract.
- Use libraries like PyMuPDF or `pdfminer.six`; or higher-level tools.
- LangChain provides convenient document loaders (e.g., `PyPDFLoader`) to extract text into a standard format.

Collecting Data

Responsible Data: Security, Privacy and Fairness

Outline

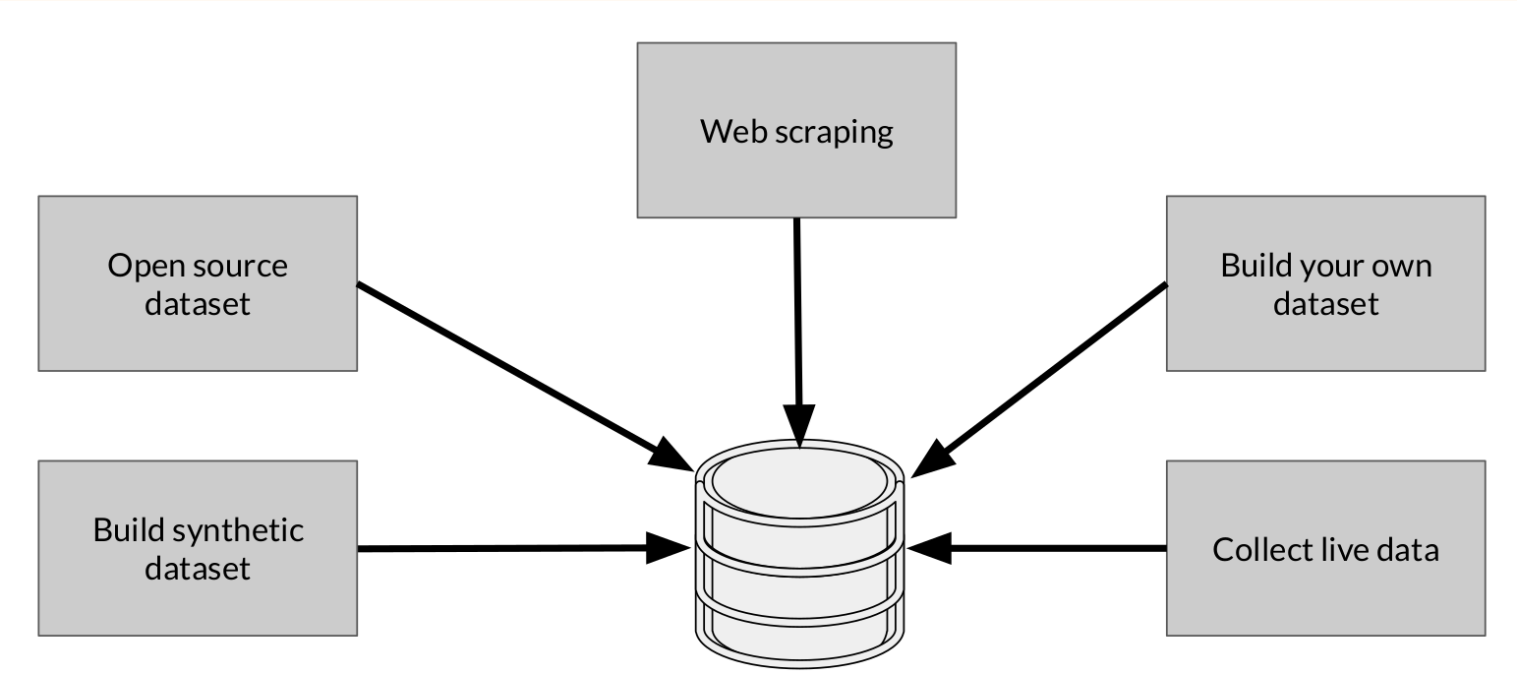
- Data Sourcing
- Data Security and User Privacy
- Bias and Fairness



Avoiding problematic biases in datasets

Example: classifier trained on the Open Images dataset

Data Sources



Data security and privacy

- Data collection and management isn't just about your model
  - Give users control of what data can be collected
  - Is there a risk of inadvertently revealing user data?
- Compliance with regulations and policies (e.g., GDPR)

Users privacy

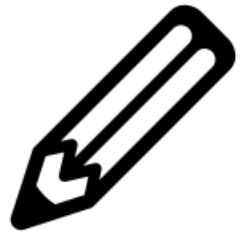
- Protect personally identifiable information
  - Aggregation – replace unique values with summary value
  - Redaction – remove some data to create a less complete picture

How ML systems can fail users

Fair

Accountable

Transparent



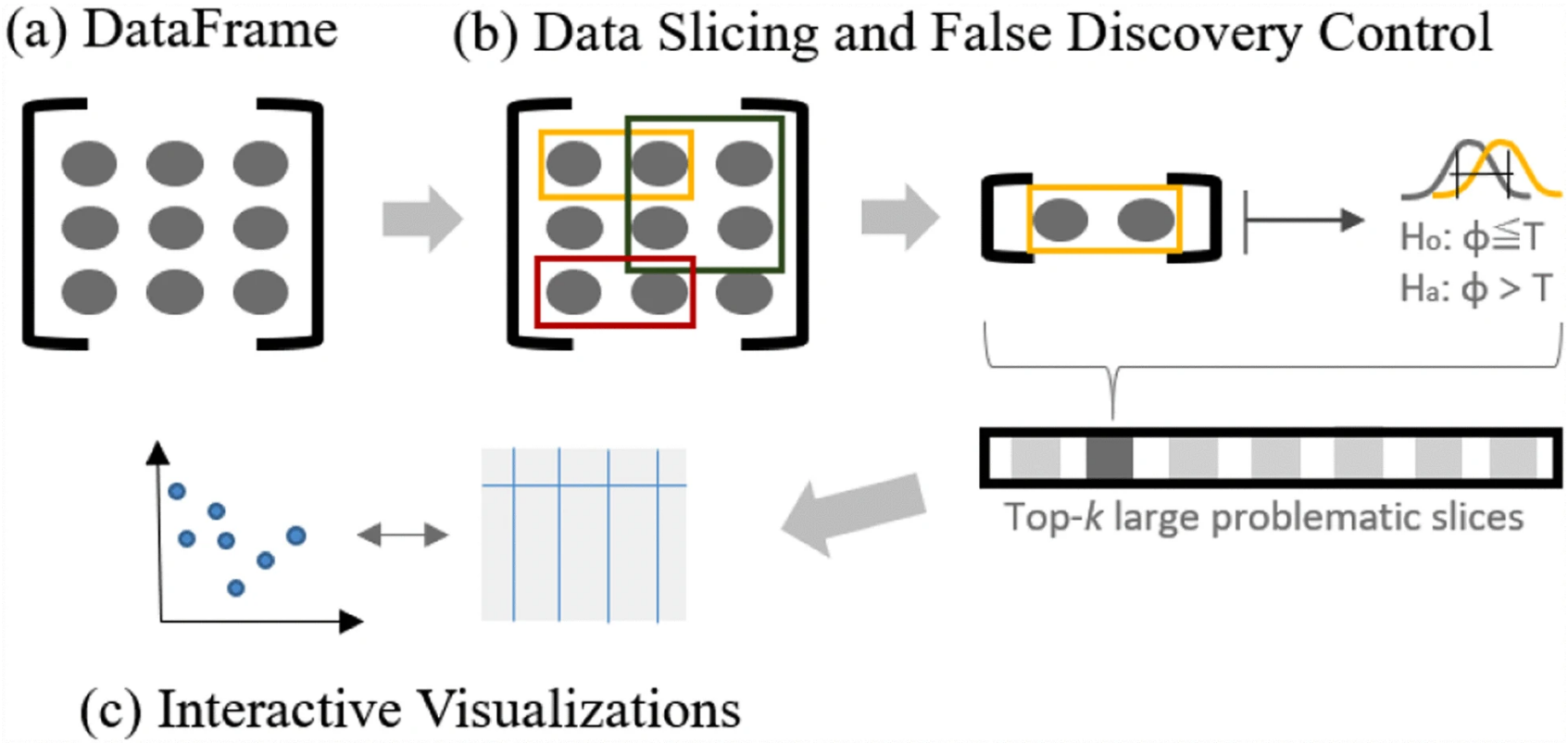
Explainable

- Representational harm
- Opportunity denial
- Disproportionate product failure
- Harm by disadvantage

Commit to fairness

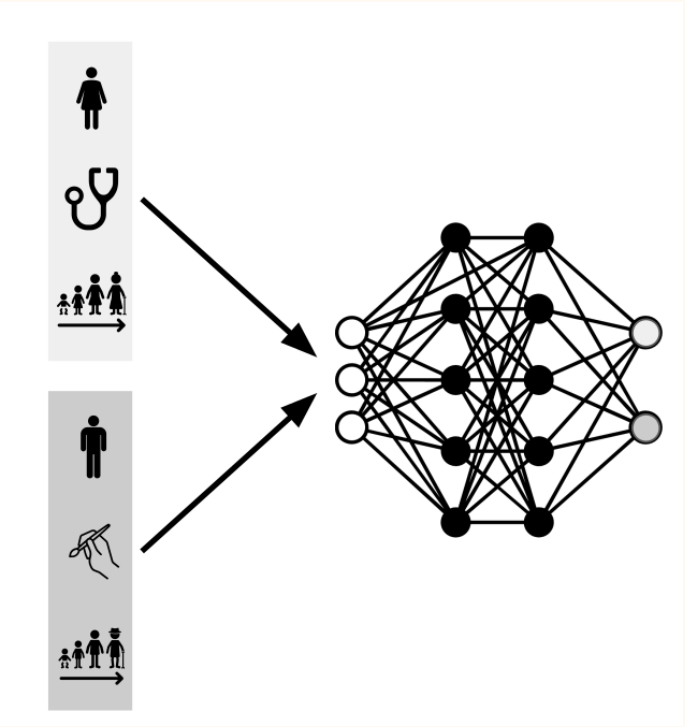
- Make sure your models are fair
  - Group fairness, equal accuracy
- Bias in human-labeled and/or collected data
- ML models can amplify biases

Data Slicing

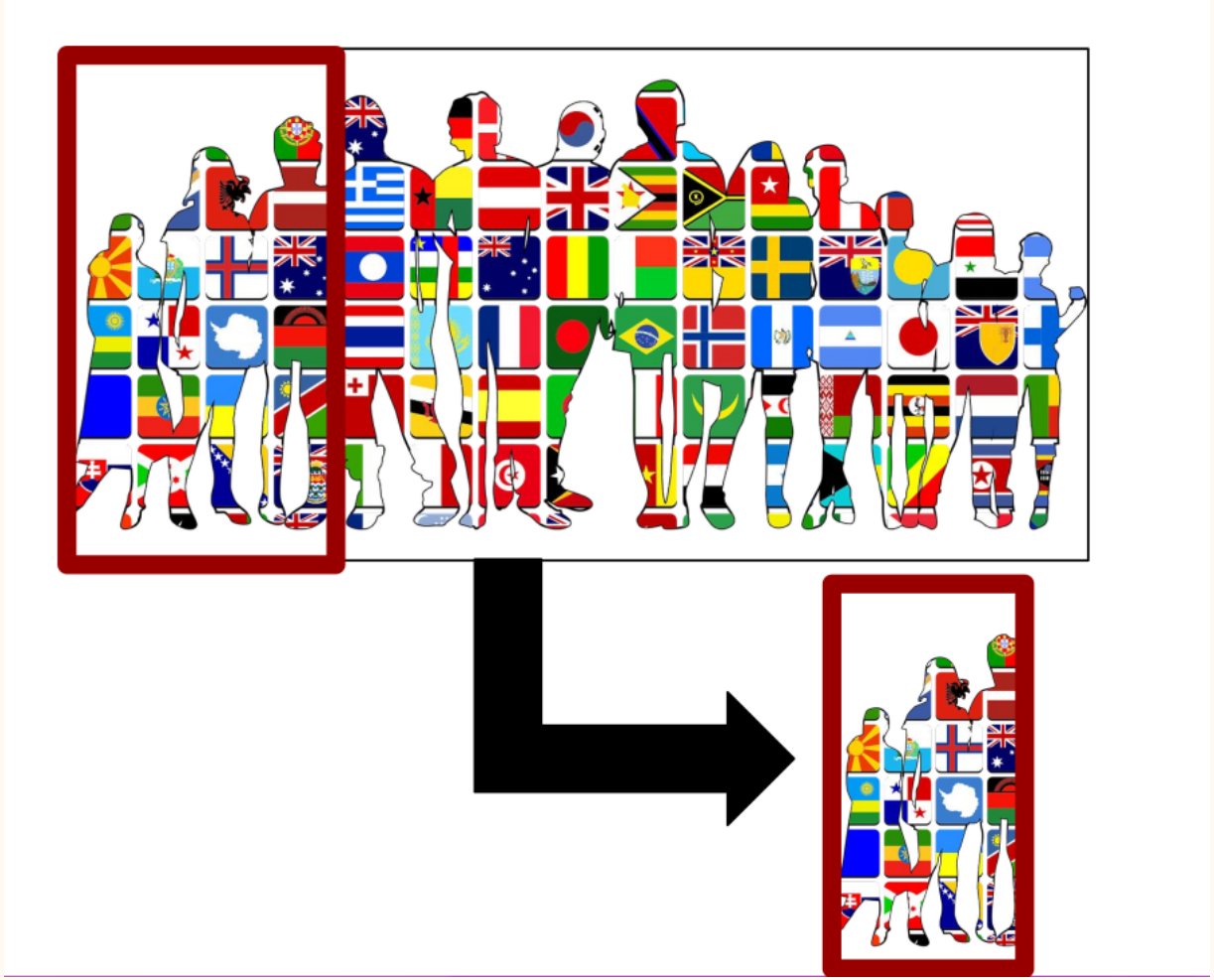


Commit to fairness

- Make sure your models are fair
- Bias in human-labeled data
- ML models can amplify biases



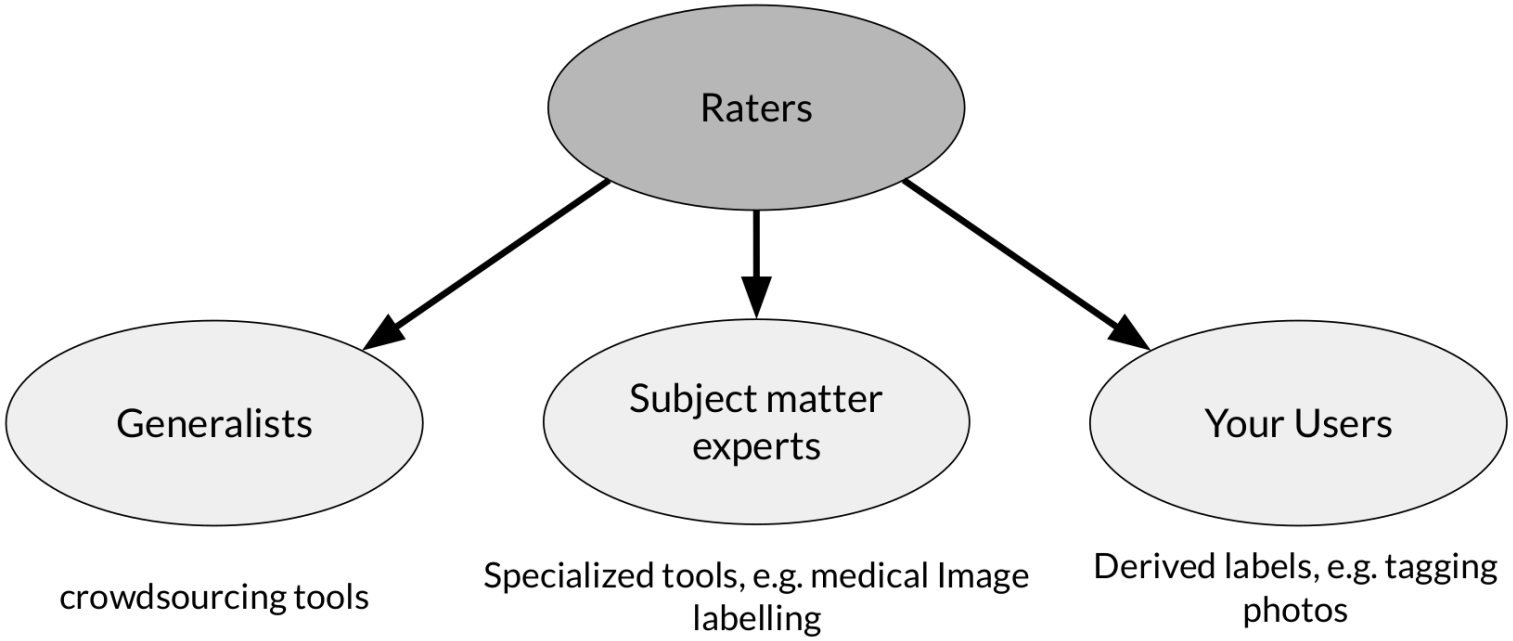
Biased data representation



Reducing bias: Design fair labeling systems

- Accurate labels are necessary for supervised learning
- Labeling can be done by:
  - Automation (logging or weak supervision)
  - Humans (aka "Raters", often semi-supervised)

Types of human raters



Key points

- Ensure rater pool diversity
- Investigate rater context and incentives
- Evaluate rater tools
- Manage cost
- Determine freshness requirements

Labs for This Week

Objective

Data Pipelines, Data Labeling, Data Version Control.

Lab Activities:

- Lab 1: [Snorkel] - [Snorkel Tutorial]
- Lab 2: [AirFlow] - [AirFlow Tutorial 2]

Lab Activities:

- Lab 1: [AirFlow] , [AirFlow Tutorial]
- Lab 2: [DVC] , [DVC Tutorial]
- Lab 3: [Data Pipeline] , [Torch Data pipeline Tutorial]

Submission Deadline: [Before the next class]

- Assignment 4: [AirFlow] , [Create a Airflow pipeline of your choice]

Submission Deadline: [Before the next class]

- Assignment 3: [Snorkel] - [Create a labeling 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:

- [What is Weak Supervision and How Does Weak Supervision Work?]

Be prepared to discuss highlights and open questions in class.

Sources

 DeepLearning.AI

 The People + AI Guidebook