Data Lifecycle in Production

Part 2: Advanced Labeling

Ramin Toosi

The importance of data

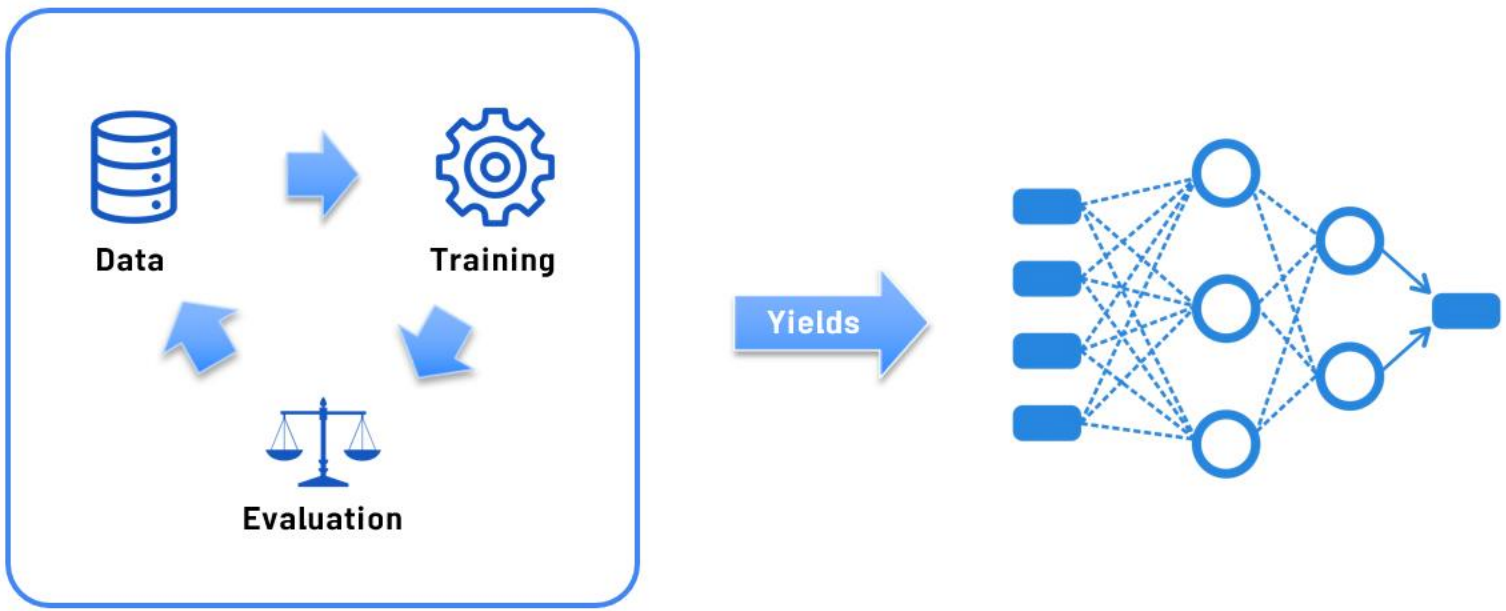
- "Data is the hardest part of ML and the most important piece to get right... Broken data is the most common cause of problems in production ML systems"
  -Scaling Machine Learning at Uber with Michelangelo - Uber

The importance of data

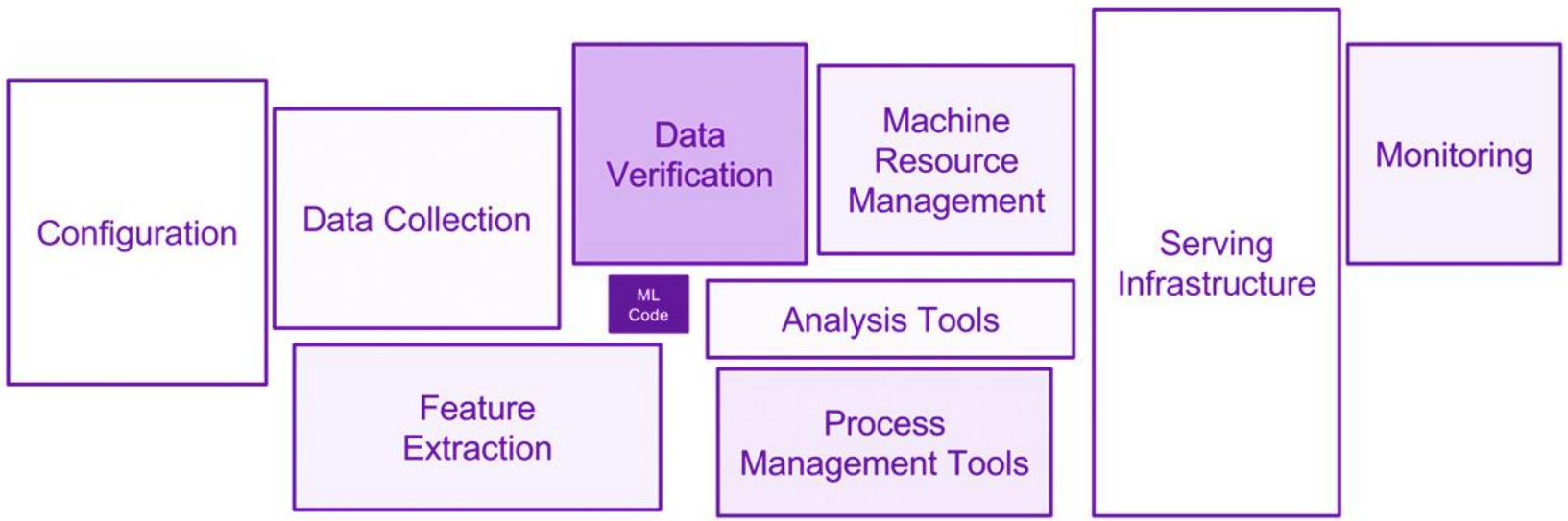
- "Data is the hardest part of ML and the most important piece to get right... Broken data is the most common cause of problems in production ML systems"
  -Scaling Machine Learning at Uber with Michelangelo - Uber

- "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

Traditional ML modeling



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

Machine learning development

+

Modern software development

Managing the entire life cycle of data

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

Modern software development

- **Accounts for:**

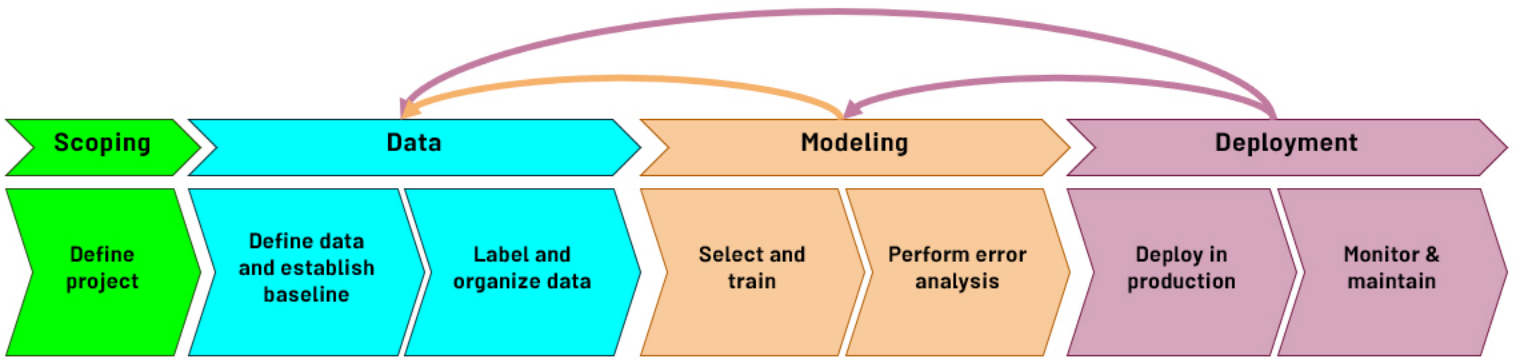
Modern software development

- **Accounts for:**
  - Scalability
  - Extensibility
  - Configuration
  - Consistency & reproducibility
  - Safety & security

Modern software development

- **Accounts for:**
  - Scalability
  - Extensibility
  - Configuration
  - Consistency & reproducibility
  - Safety & security
  - Modularity
  - Testability
  - Monitoring
  - Best practices

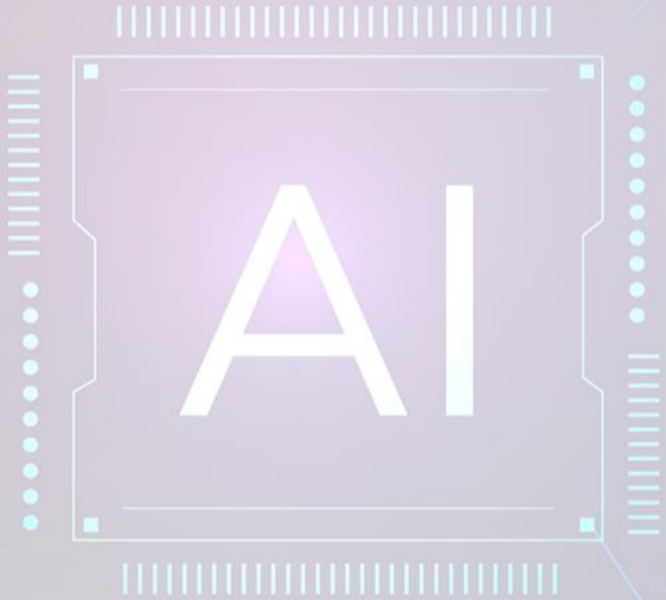
Production machine learning system



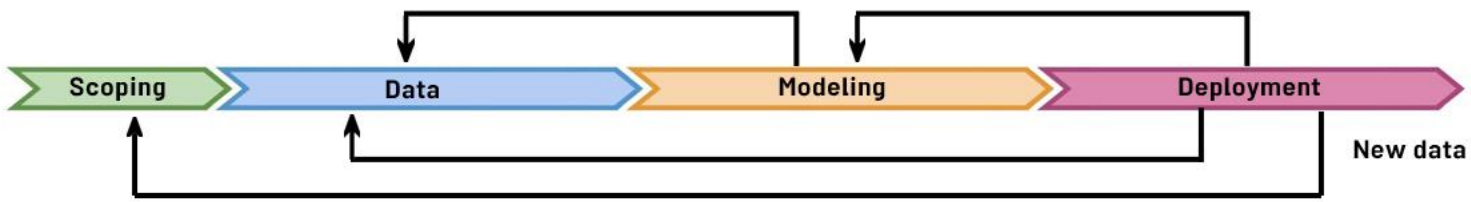
Challenges in production grade ML

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

ML Pipelines

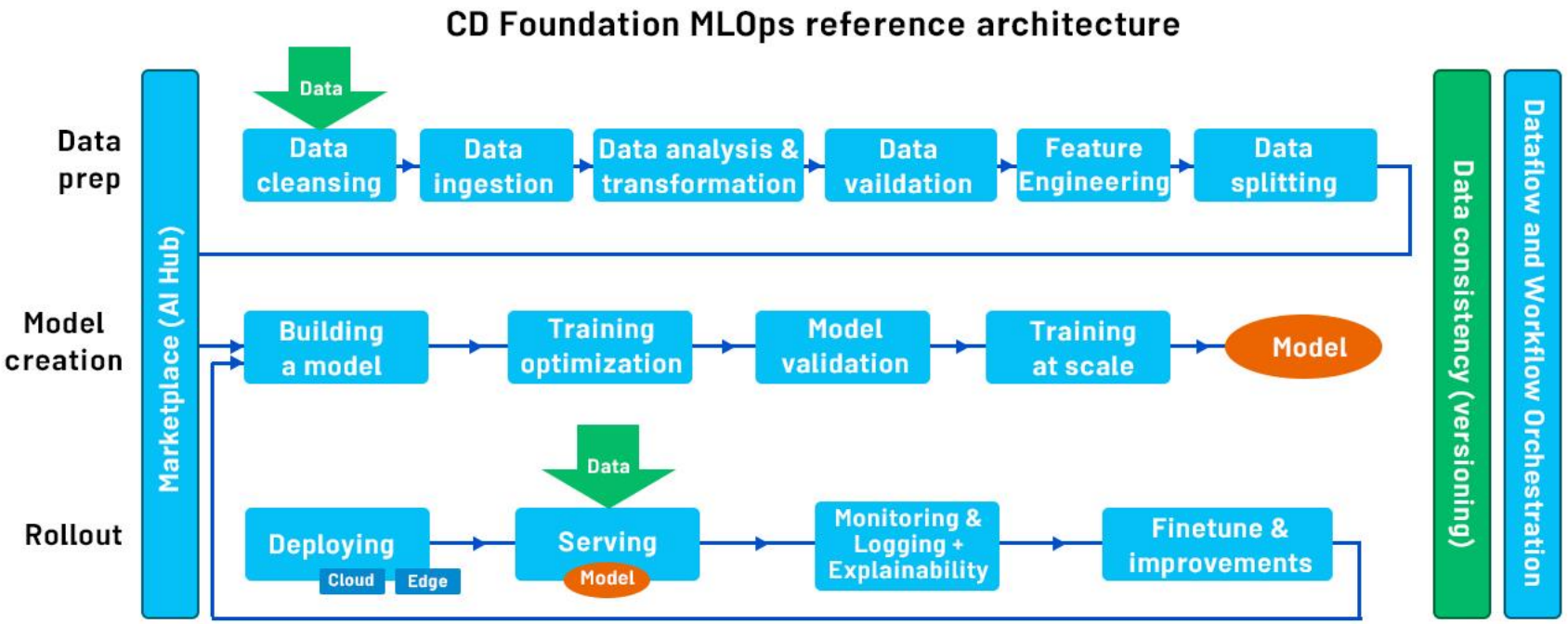


ML Pipelines



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

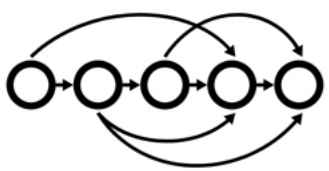
Production ML infrastructure



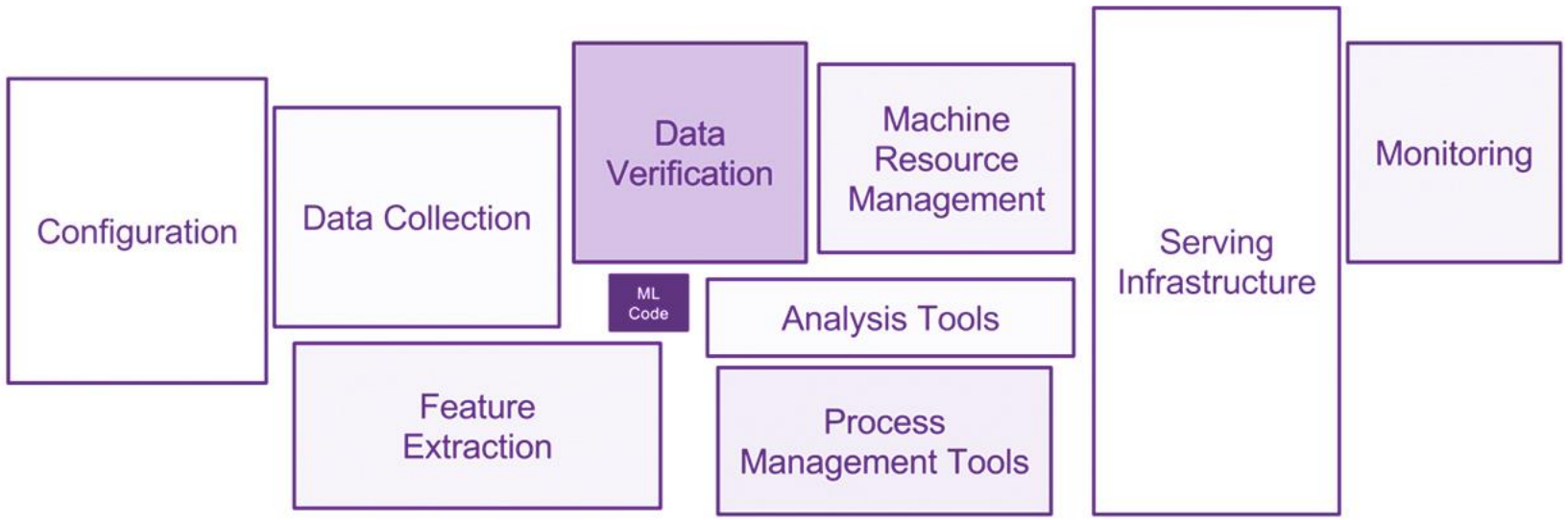
Directed acyclic graphs

Scoping | Data | Modeling | Deployment

- 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.



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

TensorFlow Extended (TFX)

- End-to-end platform for deploying production ML pipelines

TensorFlow Extended (TFX)

- End-to-end platform for deploying production ML pipelines

TensorFlow Extended (TFX)

- End-to-end platform for deploying production ML pipelines

TensorFlow Extended (TFX)

- End-to-end platform for deploying production ML pipelines

TensorFlow Extended (TFX)

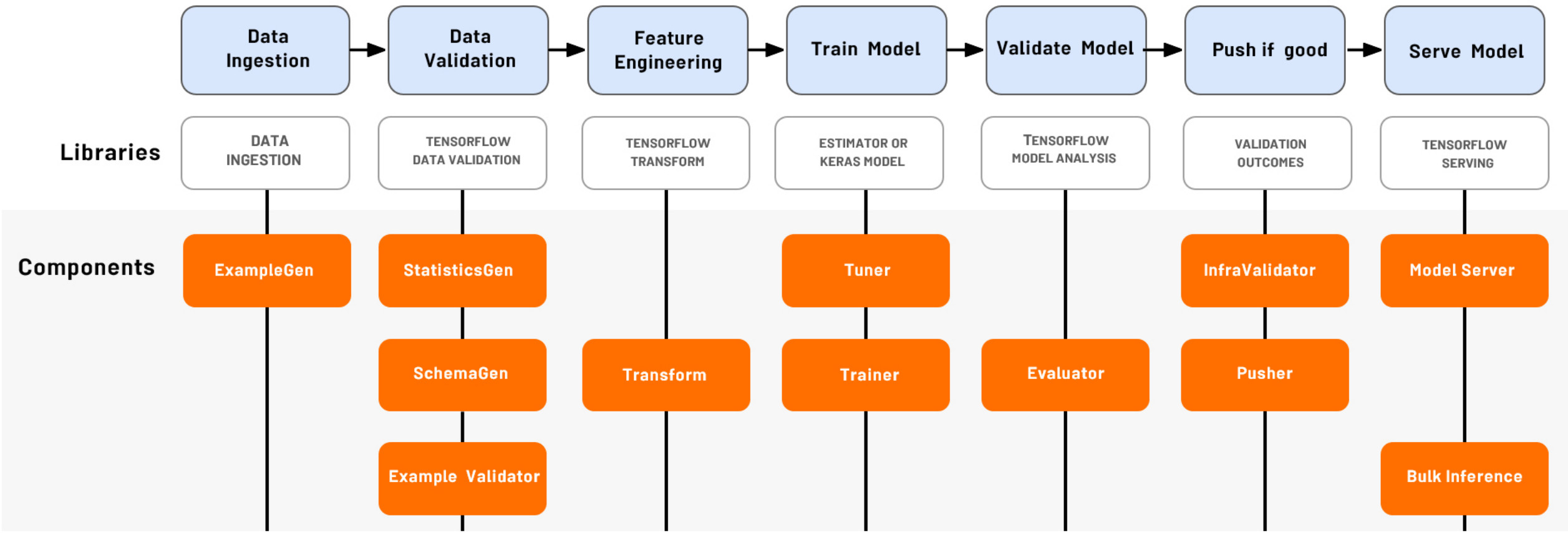
- End-to-end platform for deploying production ML pipelines

TensorFlow Extended (TFX)

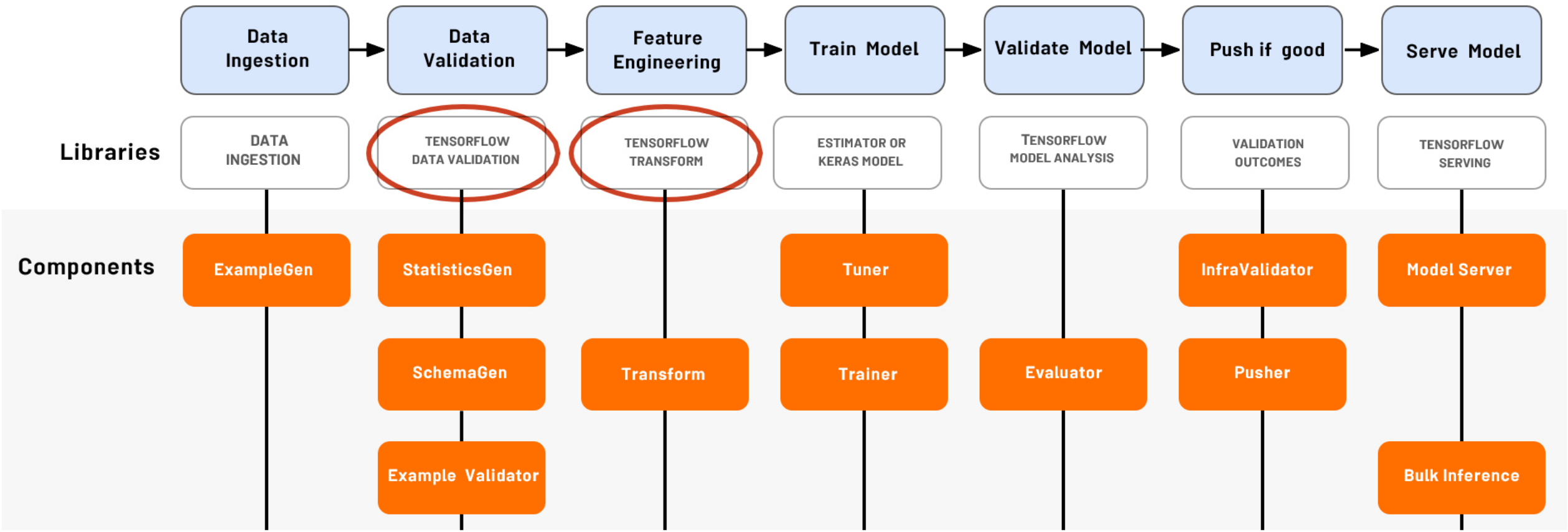
- End-to-end platform for deploying production ML pipelines

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

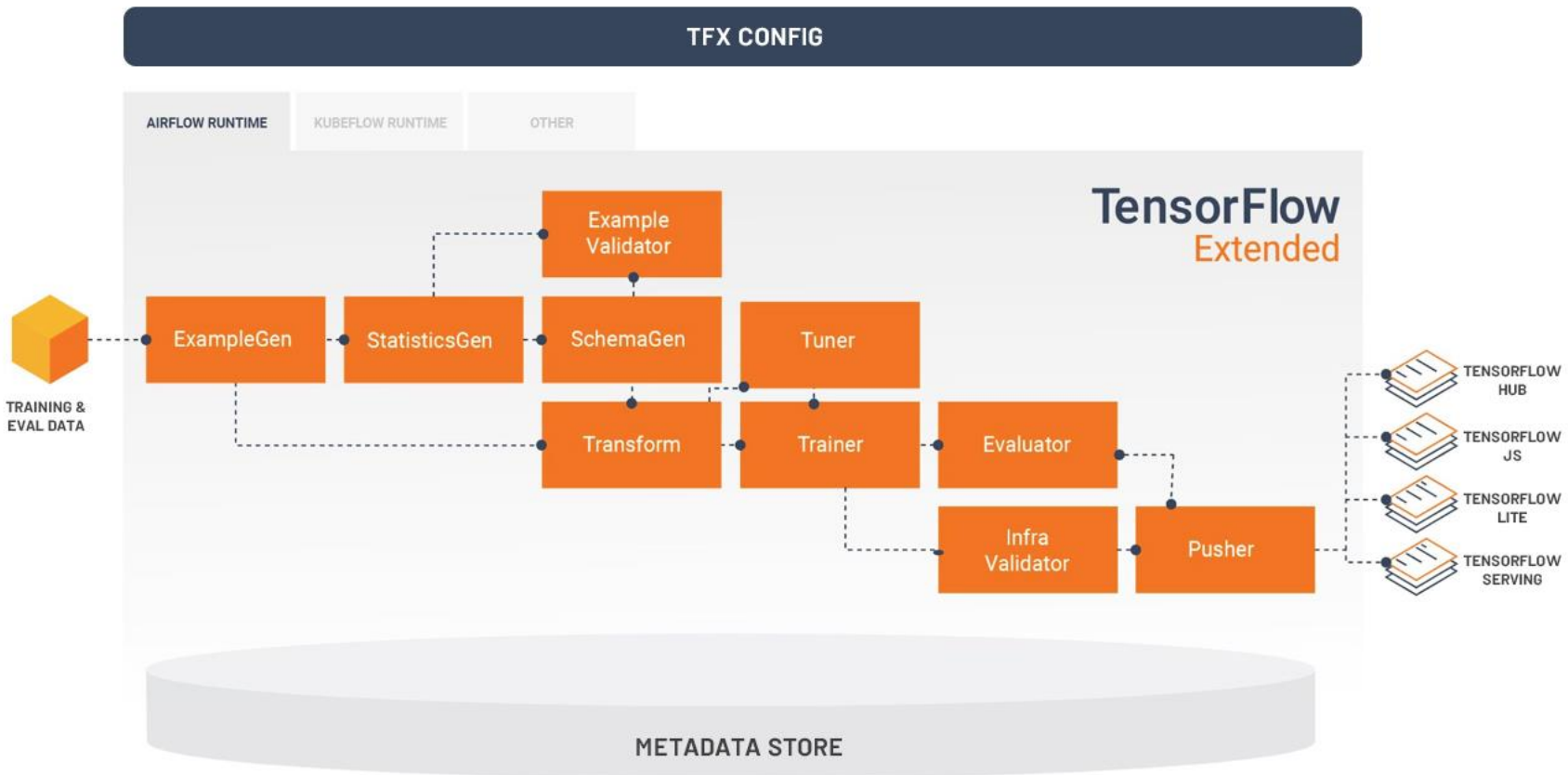
TFX production components



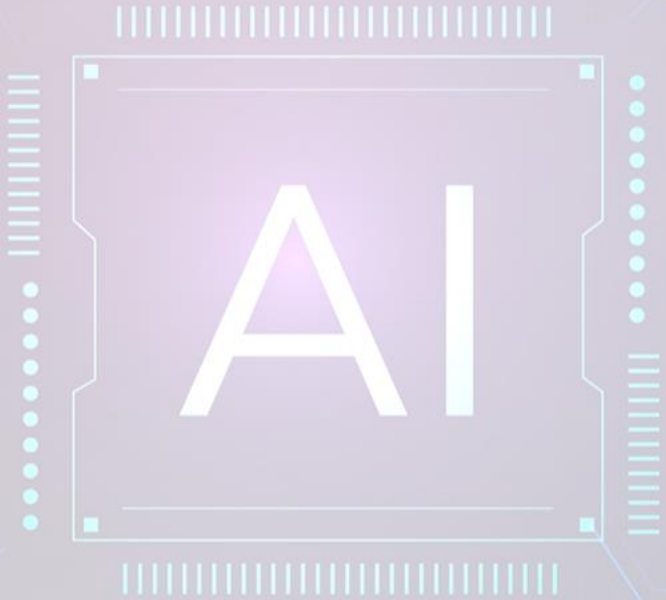
TFX production components



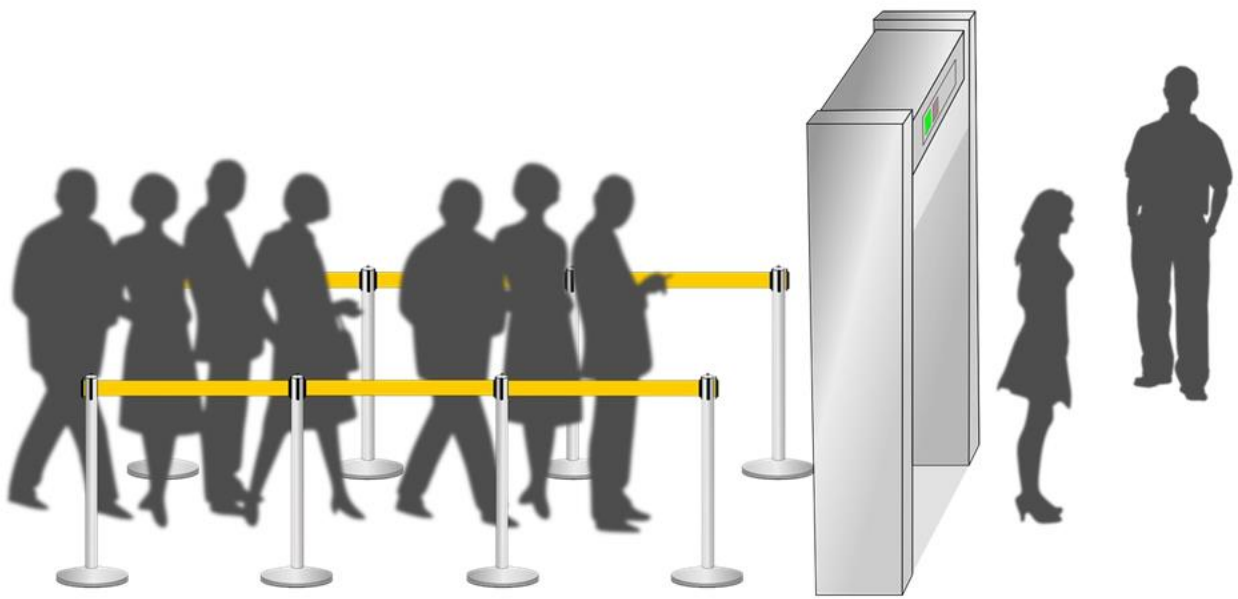
TFX


TFX CONFIG
AIRFLOW RUNTIME
KUBEFLOW RUNTIME
OTHER
TensorFlow
Extended
Example Validator
ExampleGen
StatisticsGen
SchemaGen
Tuner
Transform
Trainer
Evaluator
Infra Validator
Pusher
TRAINING & EVAL DATA
TENSORFLOW HUB
TENSORFLOW JS
TENSORFLOW LITE
TENSORFLOW SERVING
METADATA STORE

Data



Data

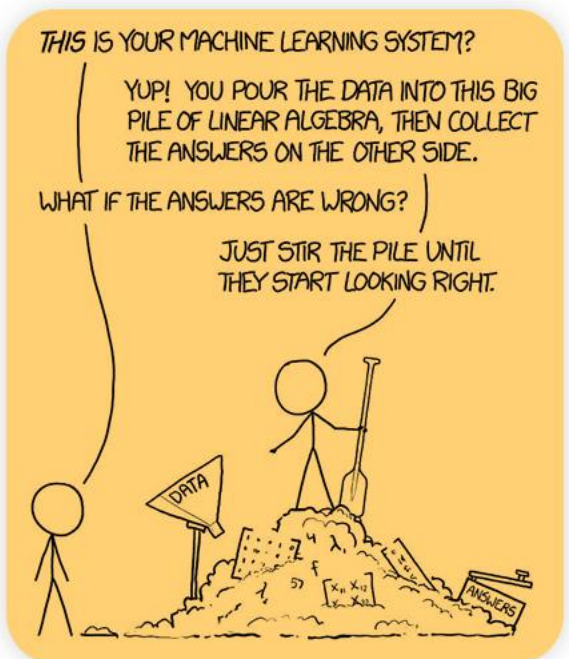


ML: Data is a first class citizen

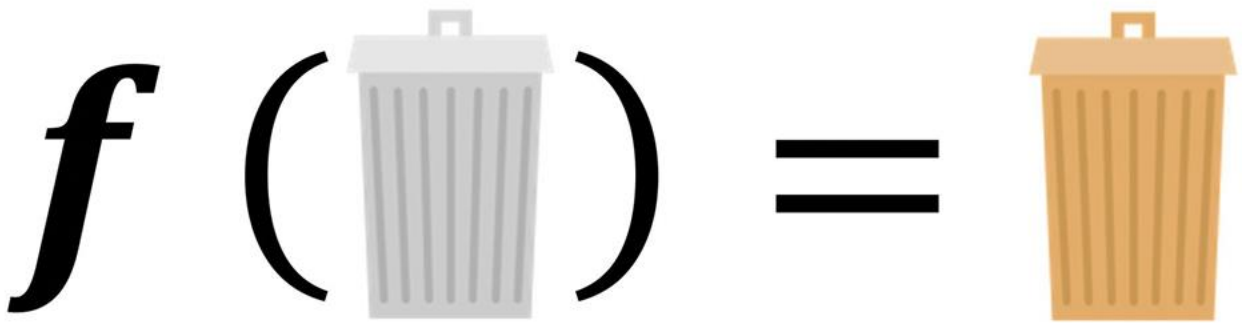
- **Software 1.0**
  - Explicit instructions to the computer
- **Software 2.0**
  - Specify some goal on the behavior of a program
  - Find solution using optimization techniques
  - Good data is key for success
  - Code in Software = Data in ML

Everything starts with data

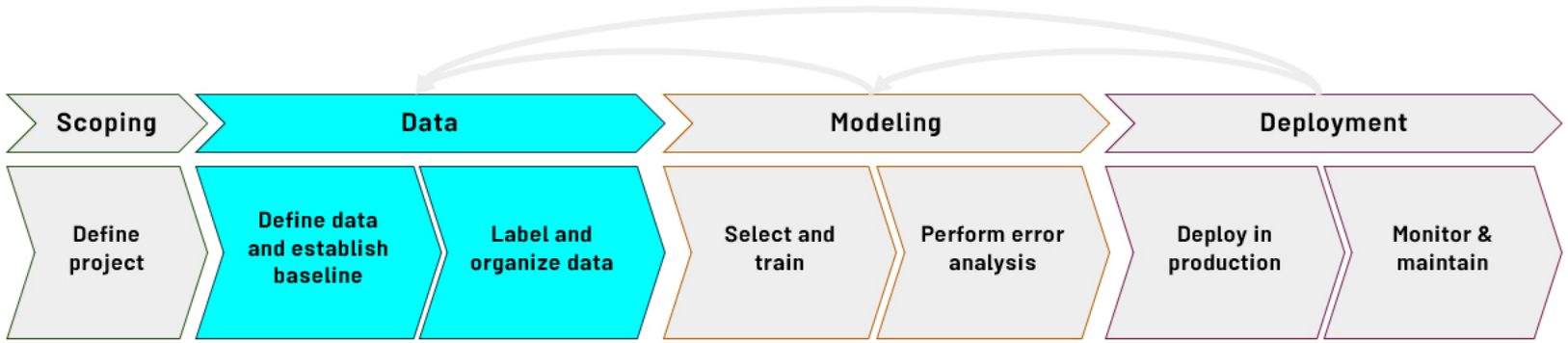
- Models aren't magic
- Meaningful data:
  - maximize predictive content
  - remove non-informative data
  - feature space coverage



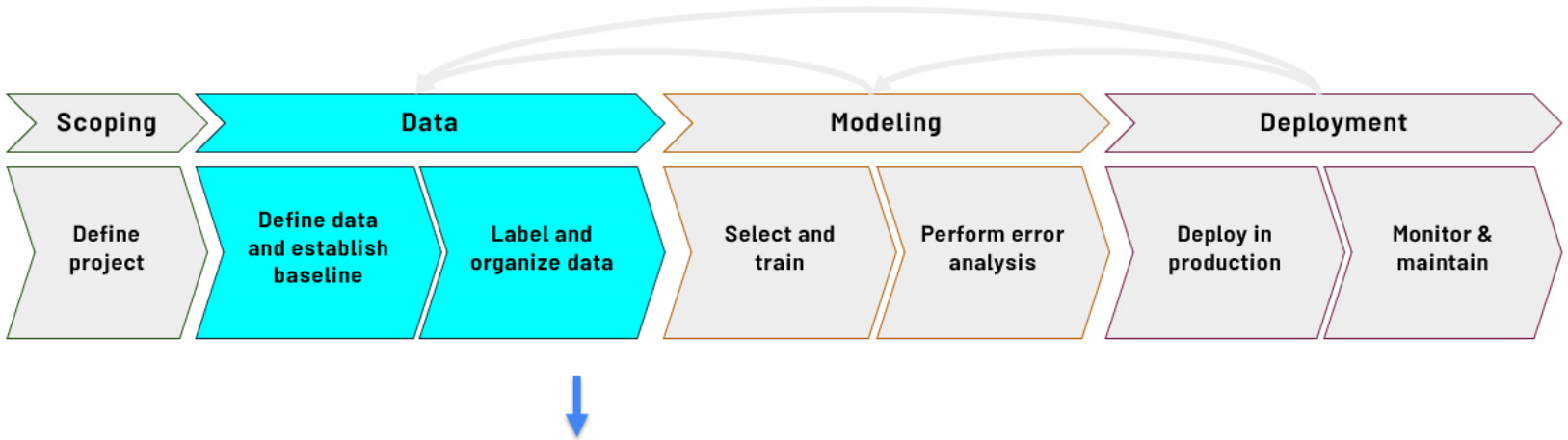
Garbage in, garbage out



Data Pipeline

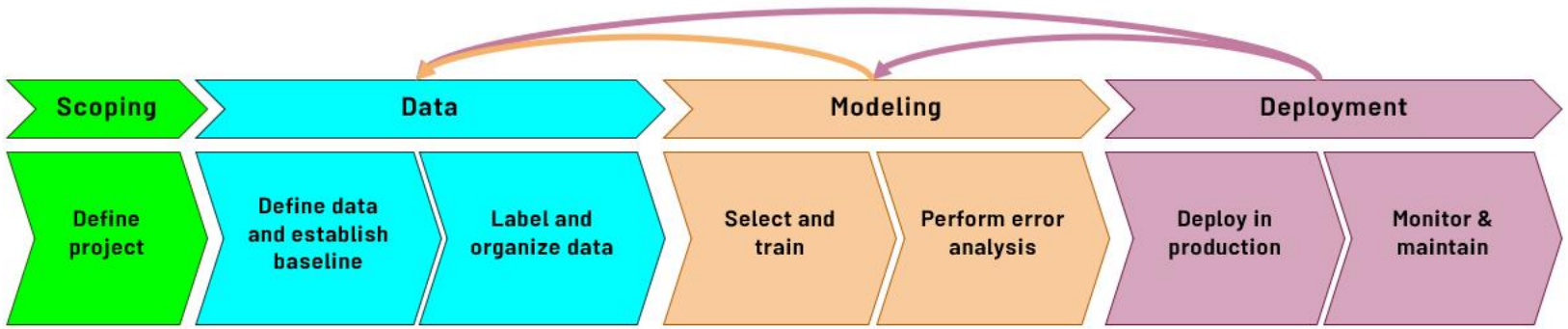


Data Pipeline

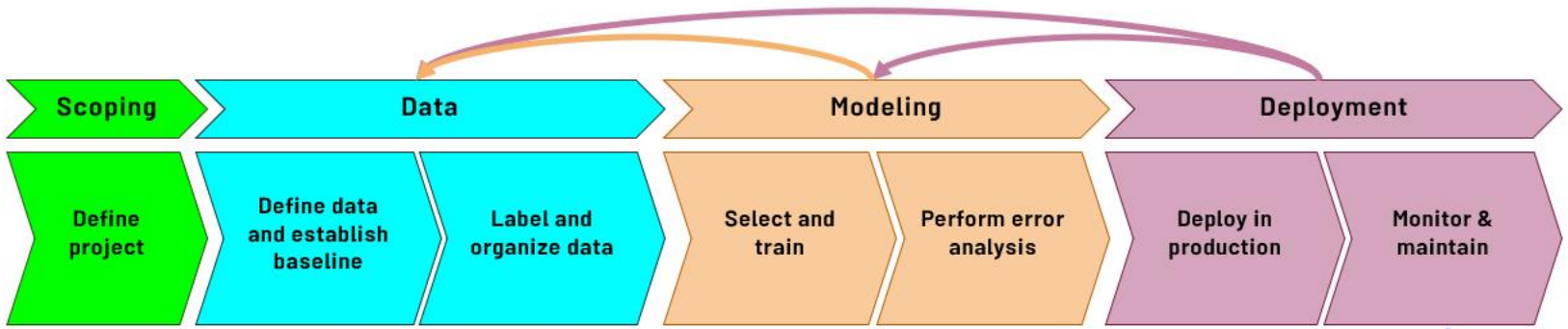


- Data collection
- Data ingestion
- Data formatting
- Feature engineering
- Feature extraction

Data collection and monitoring



Data collection and monitoring



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

Example application: Suggesting runs

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

Key considerations

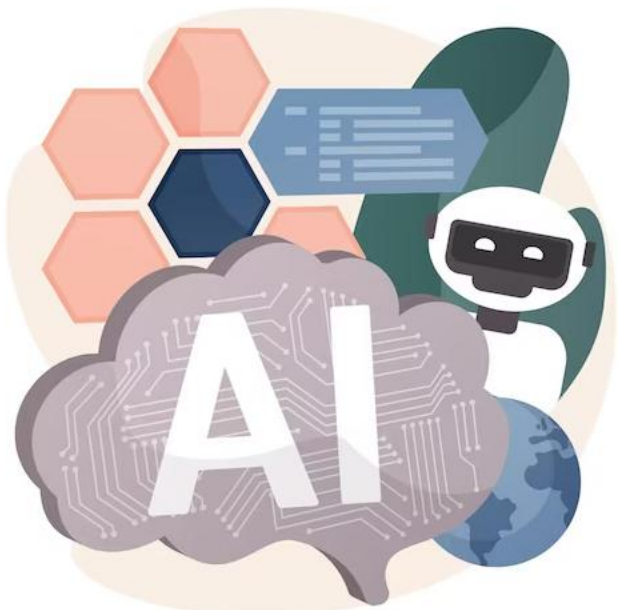
- **Data availability and collection**
  - What kind of/how much data is available?
  - How often does the new data come in?
  - Is it annotated?
    - If not, how hard/expensive is it to get it labeled?
- **Translate user needs into data needs**
  - Data needed
  - Features needed
  - Labels needed

Example dataset

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

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

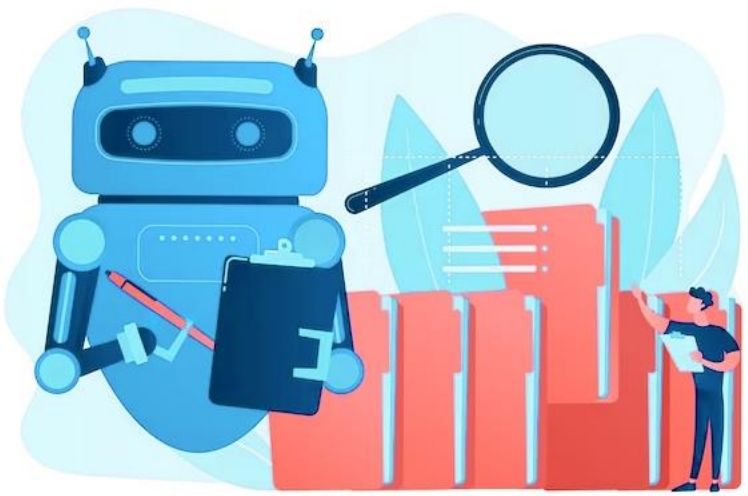
Measure data effectiveness

- 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

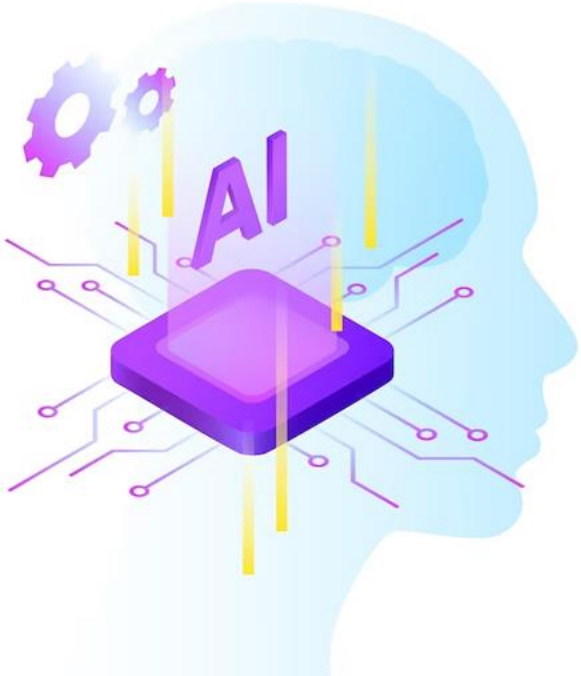
Features Needed

- Runner demographics
- Time of day
- Run completion rate
- Pace
- Distance ran
- Elevation gained
- Heart rate

Translate user needs into data needs

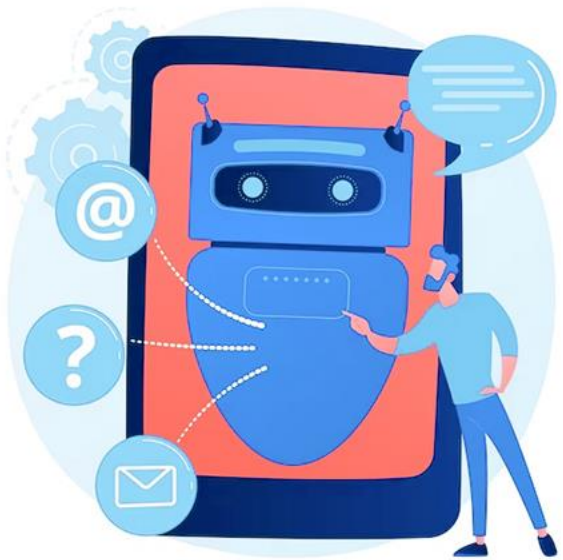
Labels Needed

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



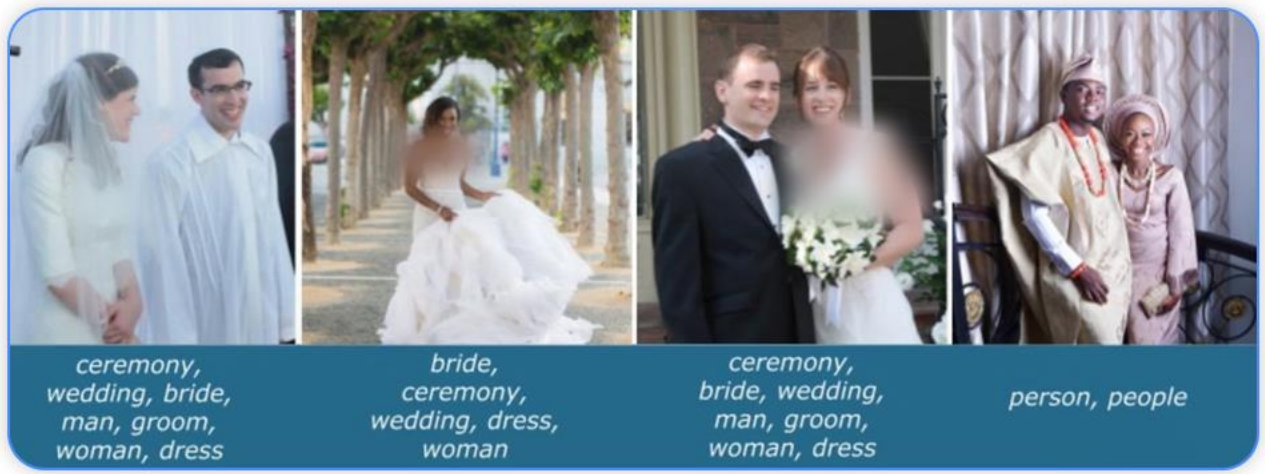
Key points

- **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

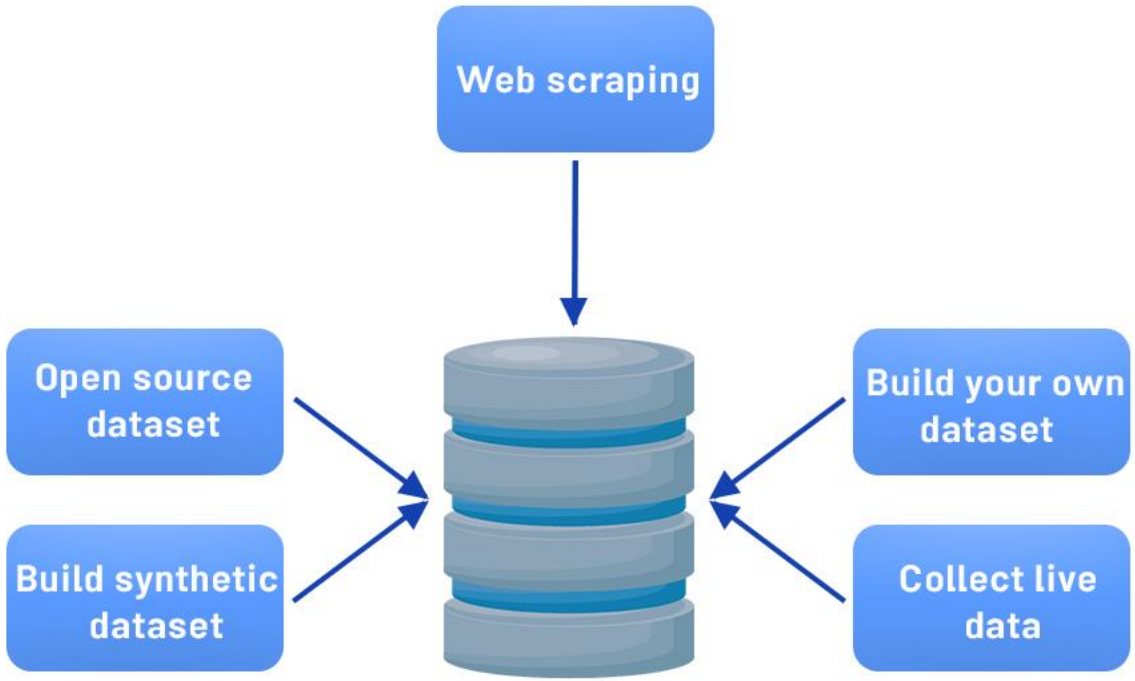


Avoiding problematic biases in datasets

- Example: classifier trained on the Open Images dataset



Source Data Responsibly



Data security and privacy

- Data collection and management isn't just about your model
  - Give user 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 less complete picture

How ML systems can fail users

Fair

Accountable

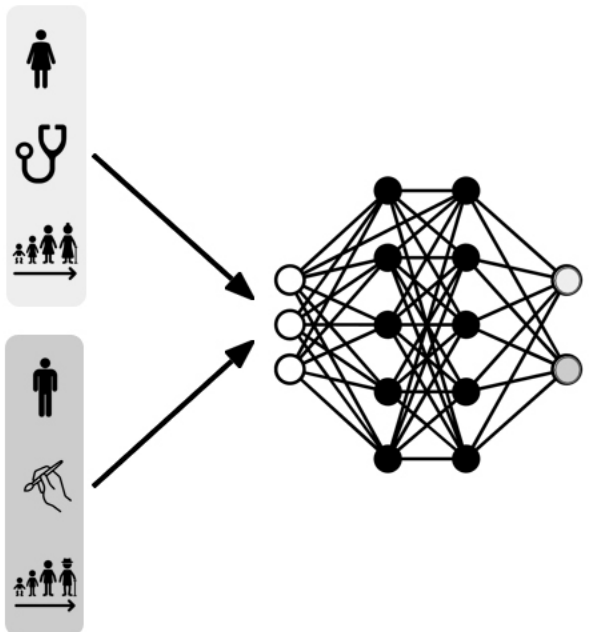
Transparent

Explainable

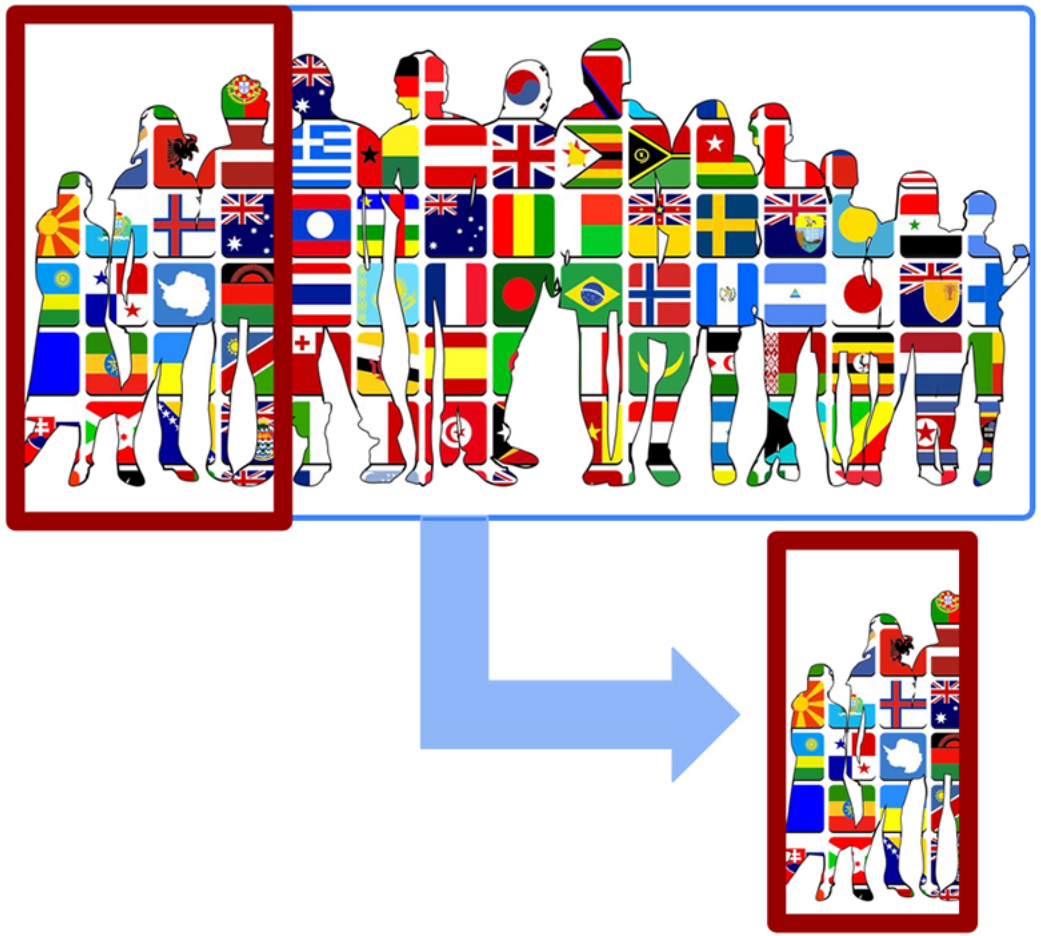
- Representational harm
- Opportunity denial
- Disproportionate product failure
- Harm by disadvantage

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



Biased data representation



Reducing bias: Design fair labeling systems

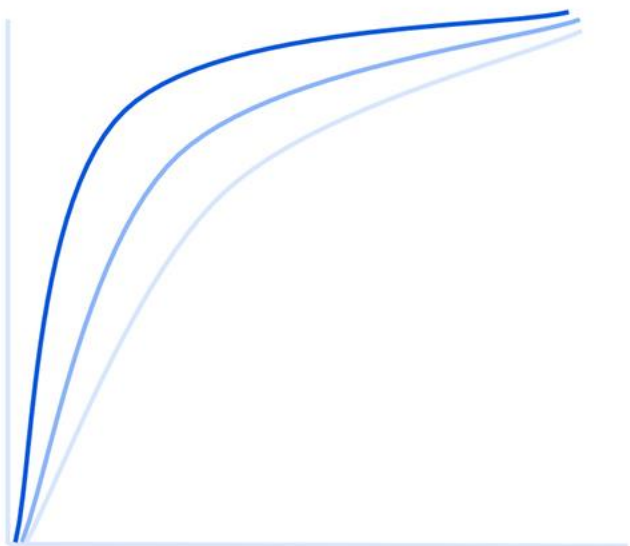
- Accurate labels are necessary for supervised learning
- Labeling can be done by:
  - Automation (logging or weak supervision)
  - Humans (aka "Raters", often semi-supervised)

Degraded Model Performance

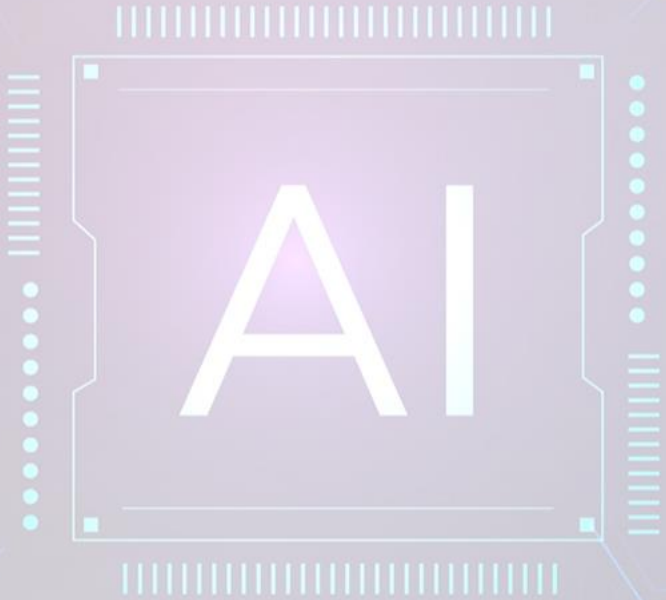
- You're an Online Retailer Selling Shoes ...
- Your model predicts **click-through rates (CTR)**, helping you decide how much inventory to order

When suddenly

- Your AUC and prediction accuracy have dropped on men's dress shoes!



How do we know that we have a problem?



What causes problems?

- **Kinds of problems:**
  - Slow - example: drift
  - Fast - example: bad sensor, bad software update

Gradual problems

- **Data changes**
  - Trend and seasonality
  - Distribution of features changes
  - Relative importance of features changes
- **World changes**
  - Styles change
  - Scope and processes change
  - Competitors change
  - Business expands to other geos

Sudden problems

- **Data collection problem**
  - Bad sensor/camera
  - Bad log data
  - Moved or disabled sensors/cameras
- **Systems problem**
  - Bad software update
  - Loss of network connectivity
  - System down
  - Bad credentials

Why "Understand" the model?

- Mispredictions do not have uniform **cost** to your business
- The **data you have** is rarely the data you wish you had
- Model objective is nearly always a **proxy** for your business objectives
- Some percentage of your customers may have a **bad experience**

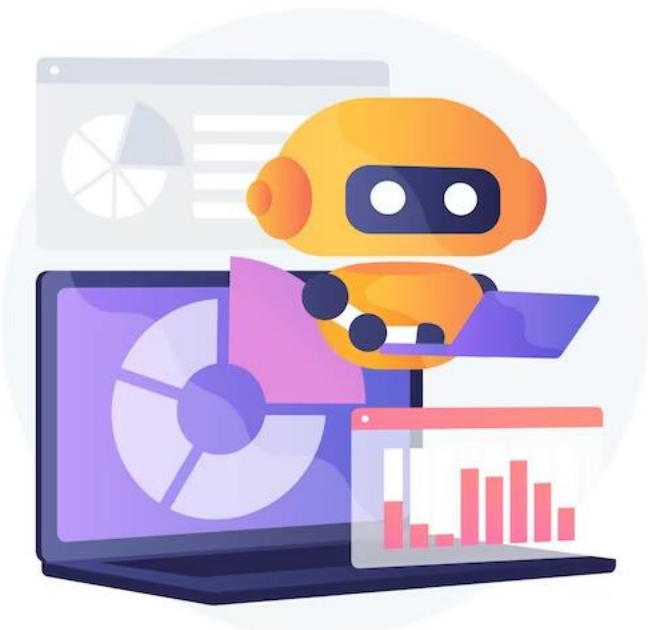
Why "Understand" the model?

- Mispredictions do not have uniform **cost** to your business
- The **data you have** is rarely the data you wish you had
- Model objective is nearly always a **proxy** for your business objectives
- Some percentage of your customers may have a **bad experience**

The real world does not stand still!

Detecting problems with deployed models

- Data and scope changes
- Monitor models and validate data to find problems early
- Changing ground truth: **label** new training data



Easy problems

- Ground truth changes slowly (months, years)
- Model retraining driven by:
  - Model improvements, better data
  - Changes in software and/or systems
- Labeling
  - Curated datasets
  - Crowd-based

Harder problems

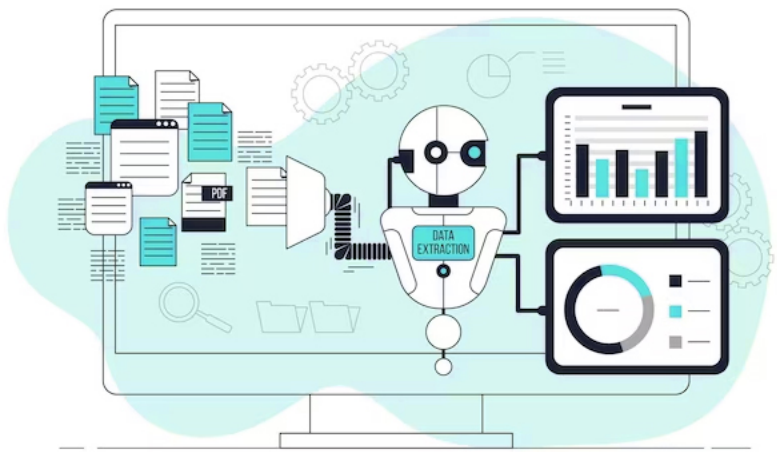
- Ground truth changes faster (weeks)
- Model retraining driven by:
  - **Declining model performance**
  - Model improvements, better data
  - Changes in software and/or system
- Labeling
  - Direct feedback
  - Crowd-based

Really hard problems

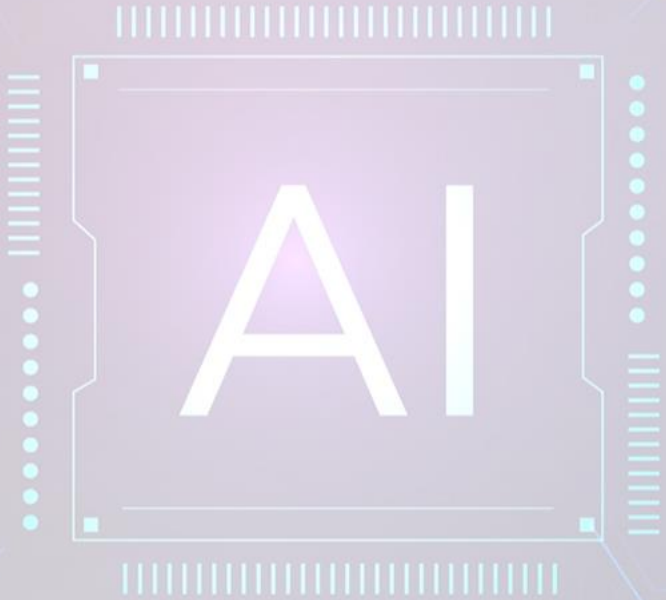
- Ground truth changes very fast (days, hours, min)
- Model retraining driven by:
  - **Declining model performance**
  - Model improvements, better data
  - Changes in software and/or system
- Labeling
  - Direct feedback
  - Weak supervision

Key Points

- Model performance decays over time
  - Data and Concept Drift
- Model retraining helps to improve performance
  - Data labeling for changing ground truth and scarce labels



Process Feedback and Human Labeling

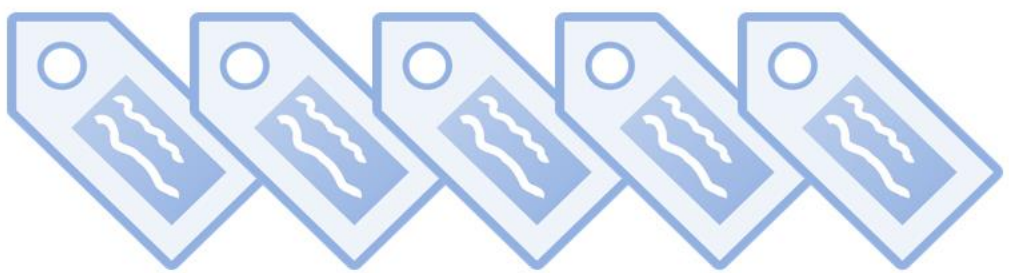


Data labeling

- Process Feedback (Direct Labeling)
- Human Labeling
- Semi-Supervised Labeling
- Active Learning
- Weak Supervision

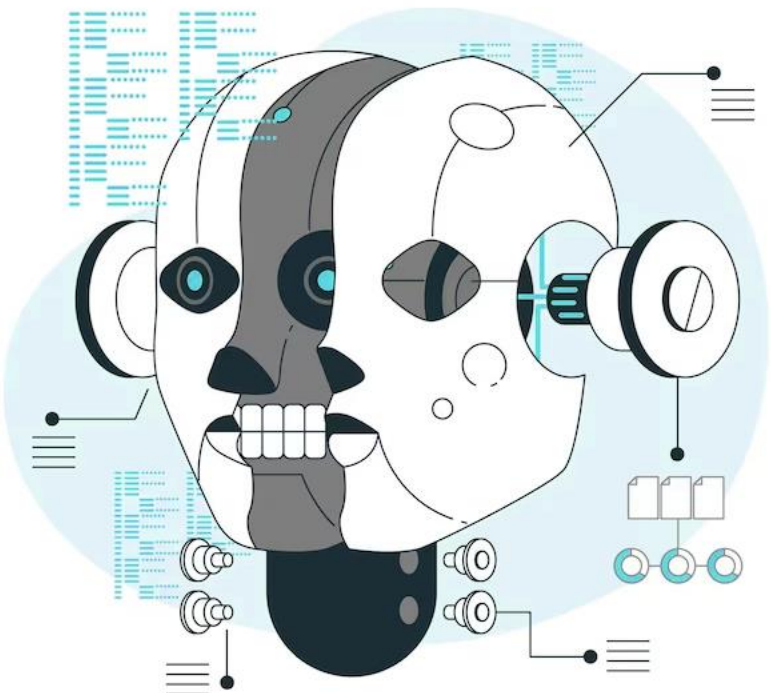
Data labeling

- **Process Feedback** — Example: Actual vs predicted click-through
- **Human Labeling** — Example: Cardiologists labeling MRI images



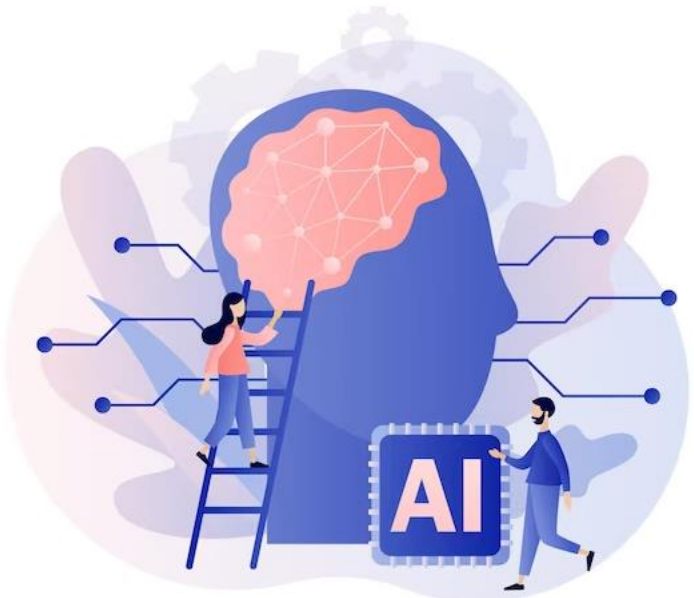
Why is labeling important in production ML?

- Using business/organisation available data
- Frequent model retraining
- Labeling ongoing and critical process
- Creating a training datasets requires labels



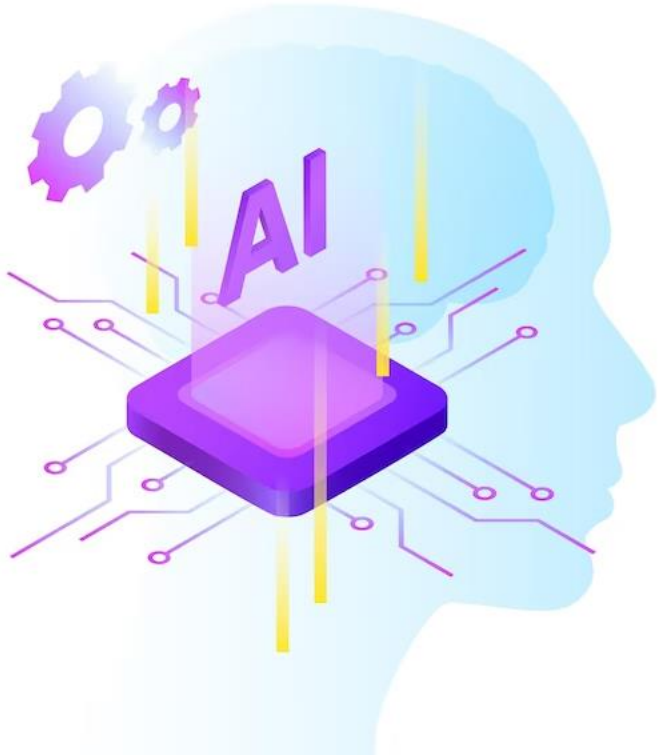
Process feedback - advantages

- Training dataset continuous creation
- Labels evolve quickly
- Captures strong label signals



Process feedback - disadvantages

- Hindered by inherent nature of the problem
- Failure to capture ground truth
- Largely bespoke design



Process feedback - Open-Source log analysis tools

- **Logstash**
  - Free and open source data processing pipeline
    - Ingests data from a multitude of sources
    - Transforms it
    - Sends it to your favorite "stash."

Process feedback - Open-Source log analysis tools

- **Logstash**
  - Free and open source data processing pipeline
    - Ingests data from a multitude of sources
    - Transforms it
    - Sends it to your favorite "stash."

- **Fluentd**
  - Open source data collector
  - Unify the data collection and consumption

Process feedback - Cloud log analytics

Cloud Log Analysis

- Google Cloud Logging
  - Data and events from Google Cloud and AWS
  - BindPlane. Logging: application components, on-premise and hybrid cloud systems
  - Sends it to your favorite "stash"
- AWS ElasticSearch
- Azure Monitor

Human labeling

- People (“raters”) to examine data and assign labels manually

Raw data

Unlabeled and ambiguous data is sent to raters for annotation

A training data set is ready for use

Human labeling - Methodology

- Unlabeled data is collected
- Human "raters" are recruited Instructions to guide raters are created Data is divided and assigned to raters
- Labels are collected and conflicts resolved

Human labeling - pros and cons

- More labels
- Pure supervised learning

Human labeling - pros and cons

- More labels
- Pure supervised learning

- Quality consistency: Difficult datasets
- Slow
- Expensive
- Small dataset curation

Why is human labeling a problem?

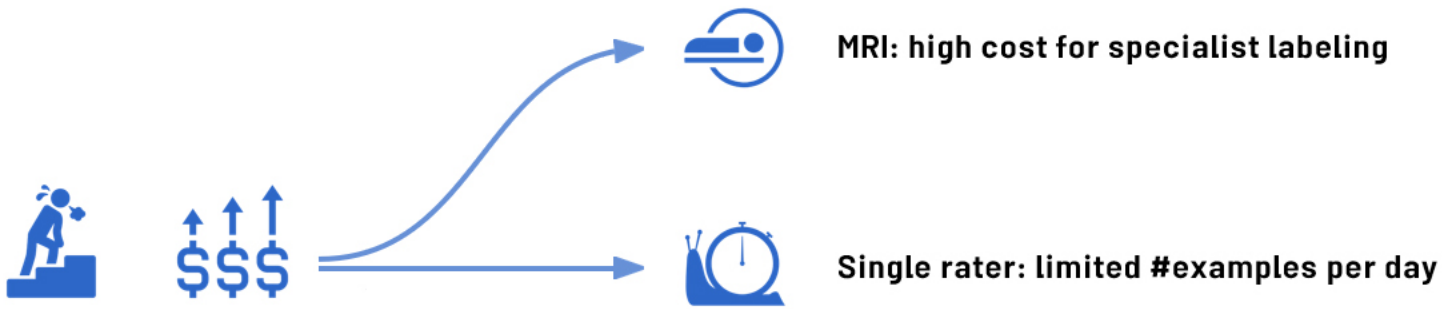
Slow, difficult and expensive

Why is human labeling a problem?

Slow, difficult and expensive

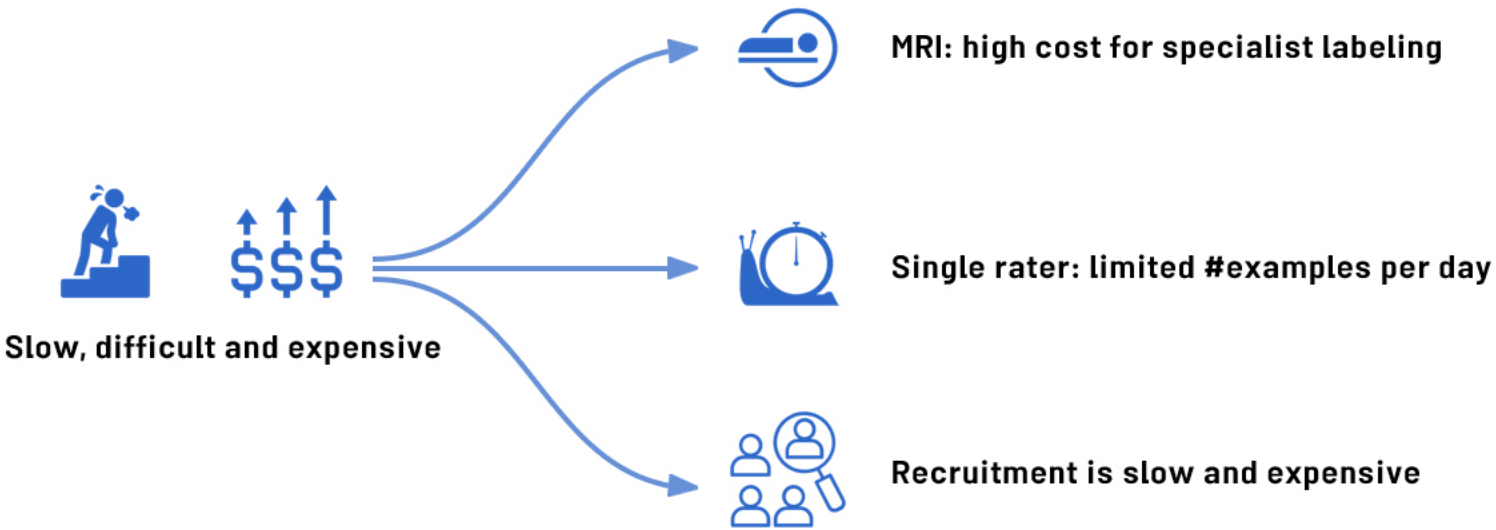
MRI: high cost for specialist labeling

Why is human labeling a problem?

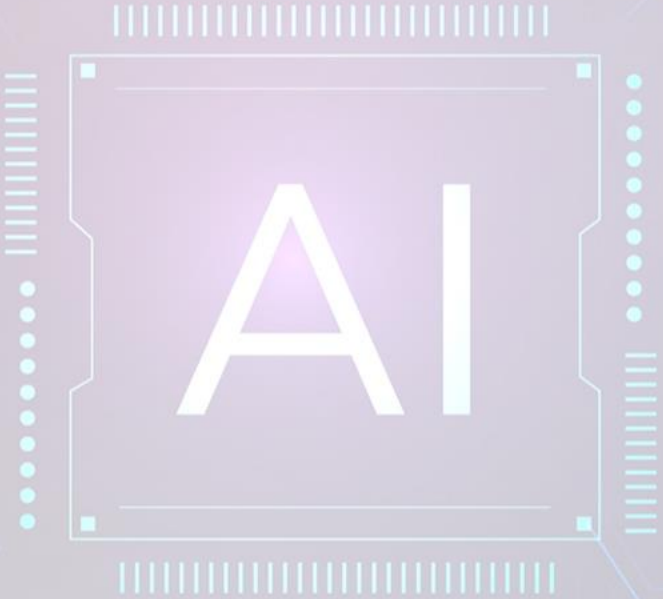


Slow, difficult and expensive

Why is human labeling a problem?

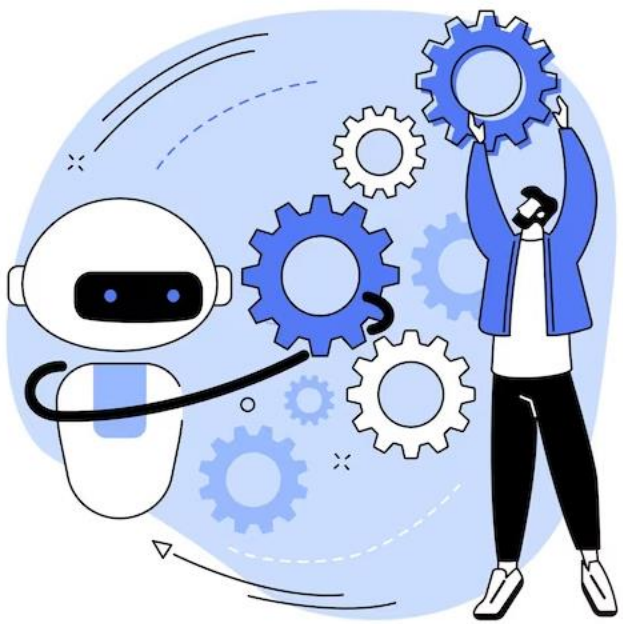


Data Issues



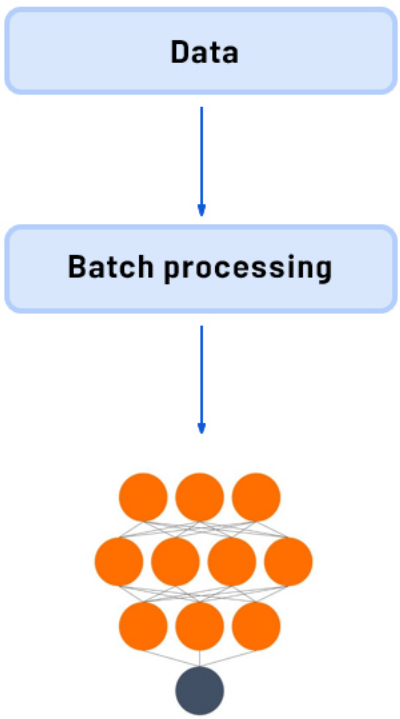
Drift and skew

- **Drift**
  - Changes in data over time, such as data collected once a day
- **Skew**
  - Difference between two static versions, or different sources, such as training set and serving set

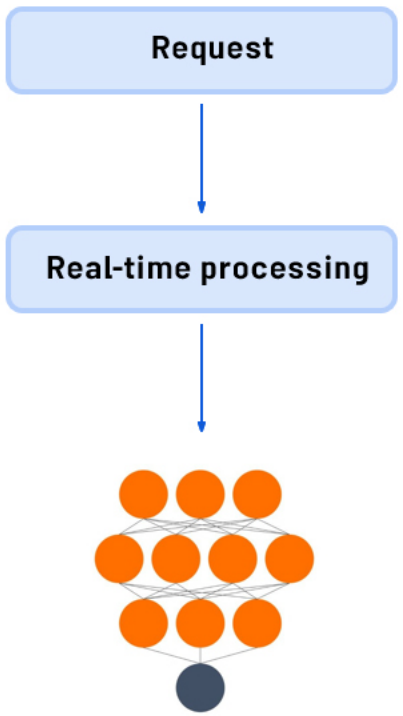


Typical ML pipeline

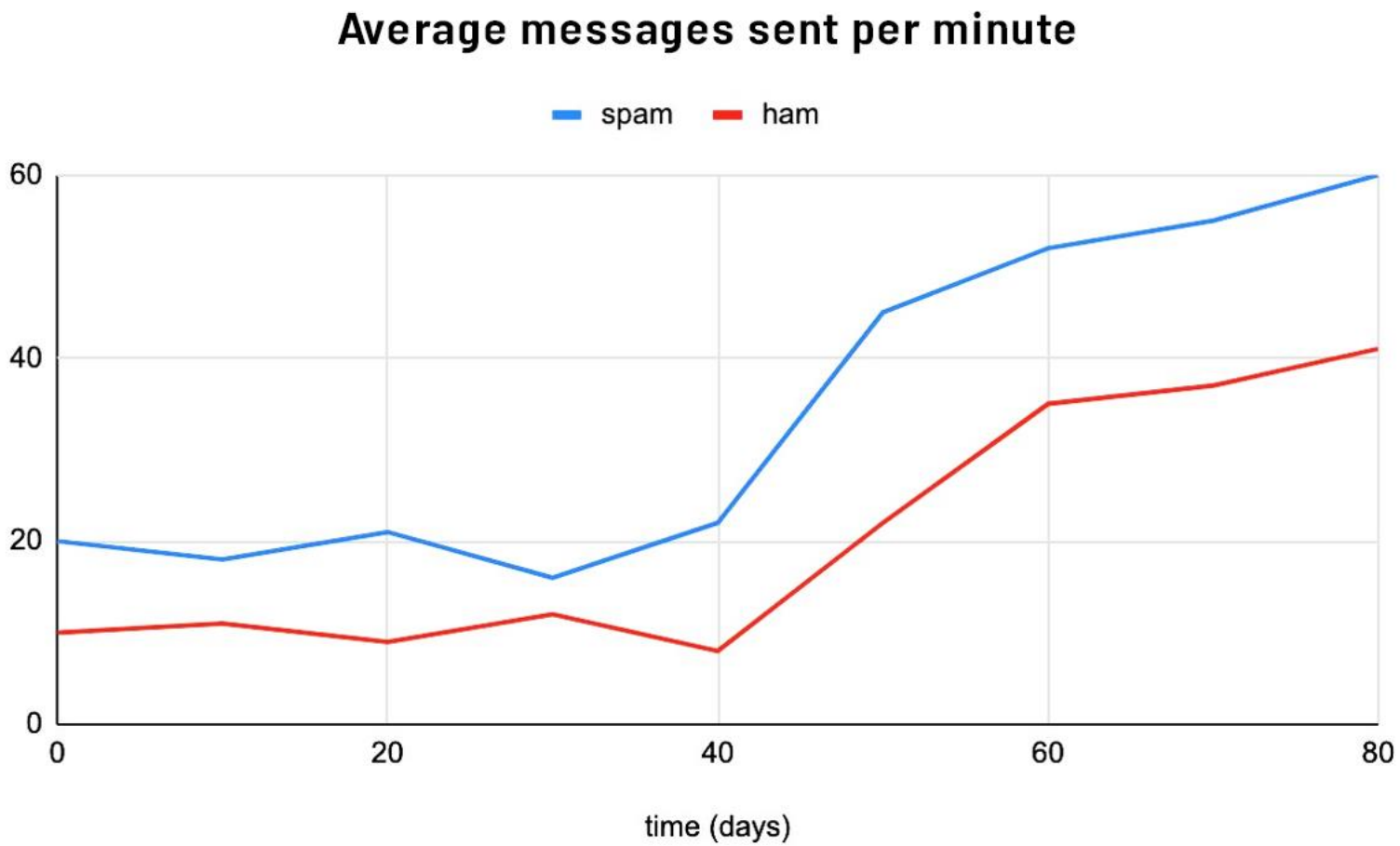
During training



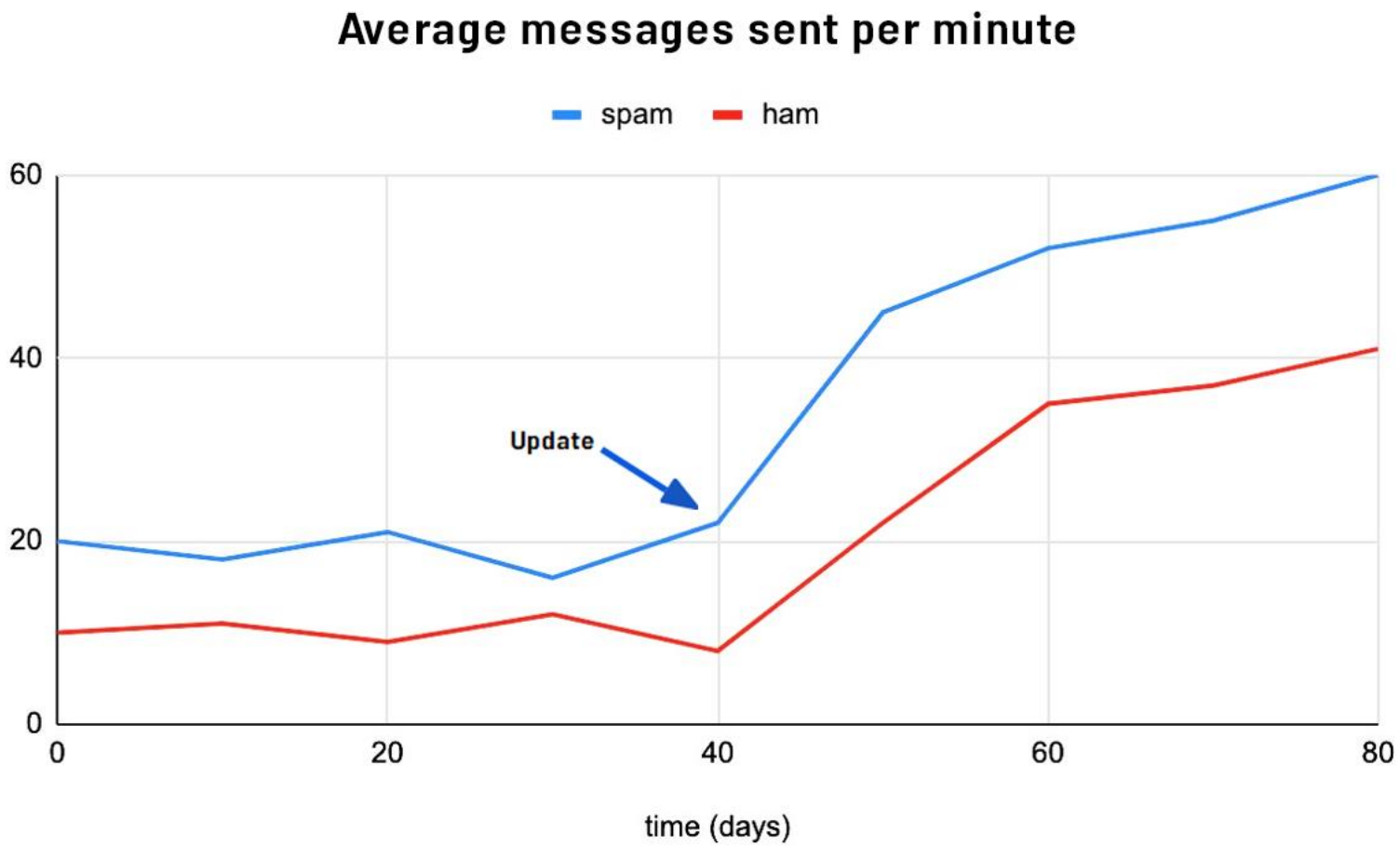
During serving



Model Decay: Data drift



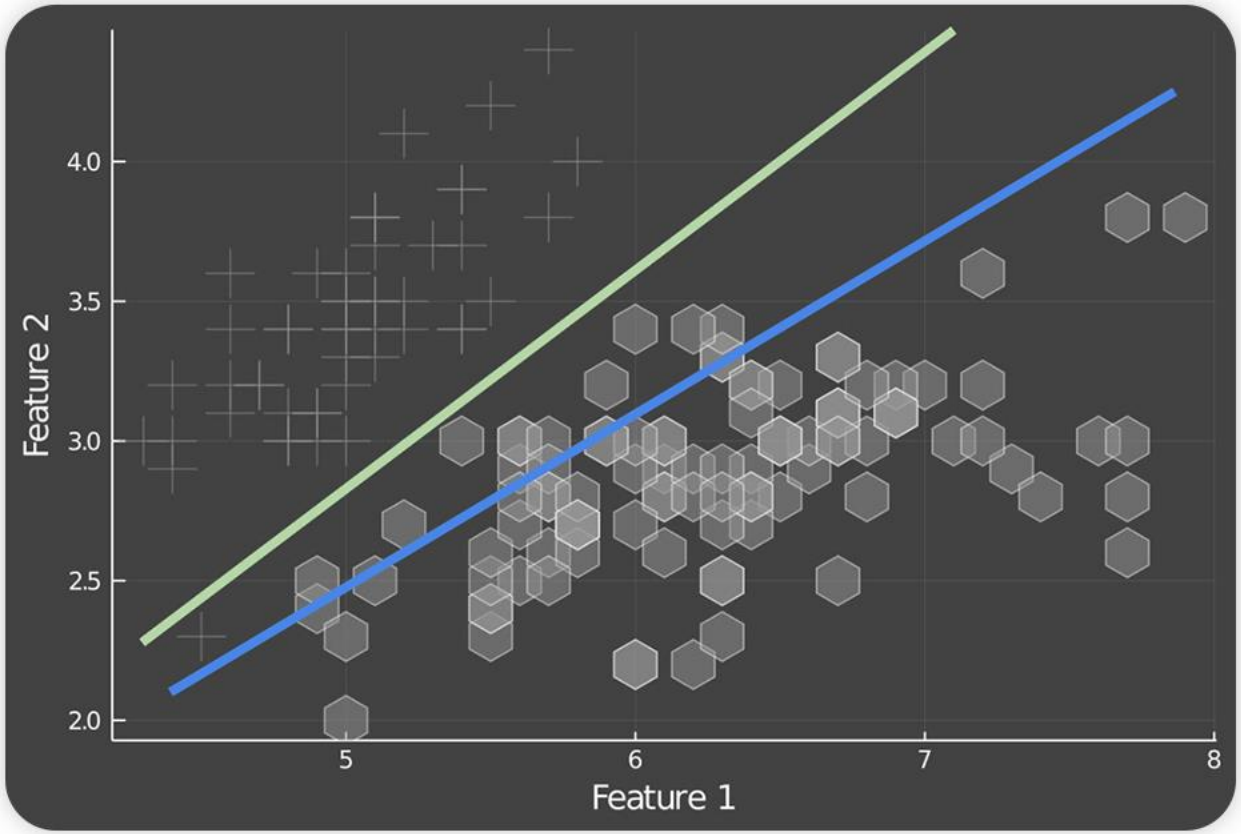
Model Decay: Data drift



Performance decay: Concept drift

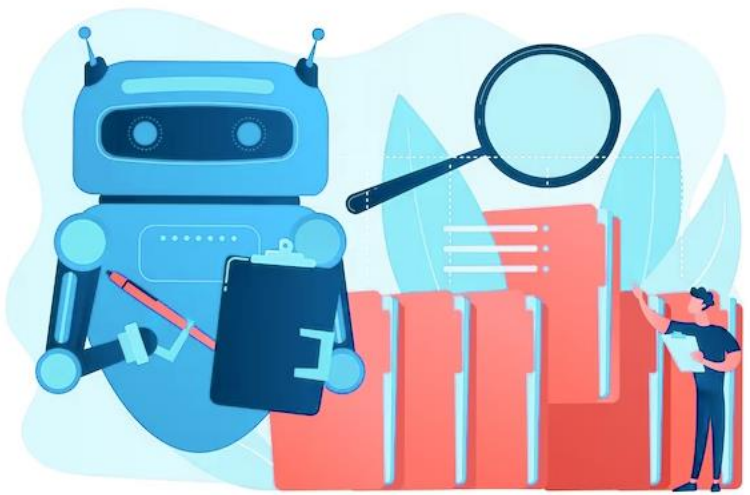
Training

Serving



Detecting data issues

- **Detecting schema skew**
  - Training and serving data do not conform to the same schema
- **Detecting distribution skew**
  - Dataset shift → covariate or concept shift
- **Requires continuous evaluation**



Detecting distribution skew

- **Dataset shift**

$$P_{\text{train}}(y, x) \neq P_{\text{serve}}(y, x)$$

- **Covariate shift**

$$P_{\text{train}}(y|x) = P_{\text{serve}}(y|x)$$

$$P_{\text{train}}(x) \neq P_{\text{serve}}(x)$$

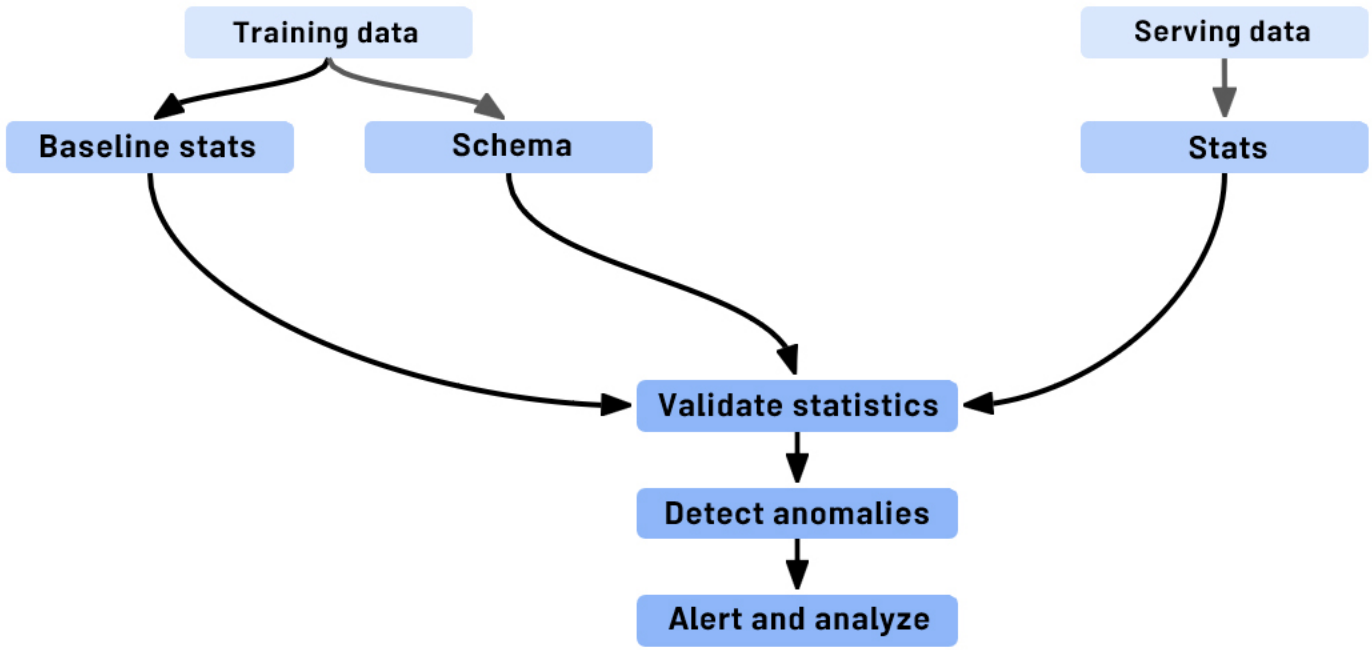
- **Concept shift**

$$P_{\text{train}}(y|x) \neq P_{\text{serve}}(y|x)$$

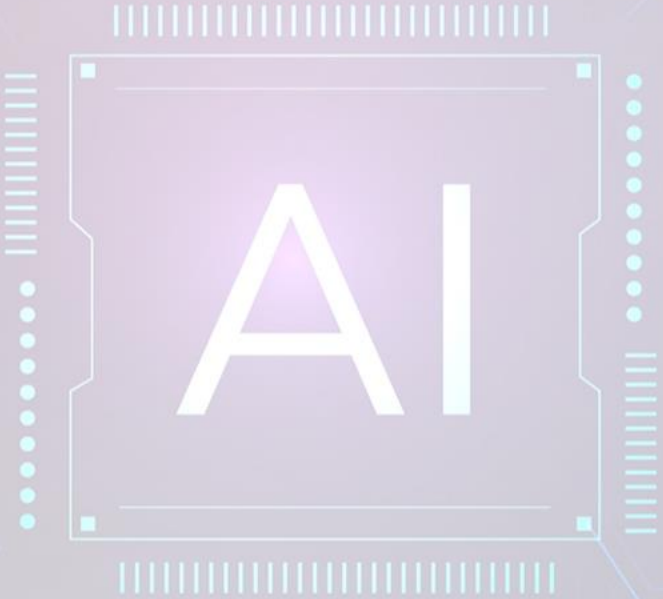
$$P_{\text{train}}(x) = P_{\text{serve}}(x)$$

	Training	Serving
Joint	$P_{\text{train}}(y, x)$	$P_{\text{serve}}(y, x)$
Conditional	$P_{\text{train}}(y\|x)$	$P_{\text{serve}}(y\|x)$
Marginal	$P_{\text{train}}(x)$	$P_{\text{serve}}(x)$

Skew detection workflow

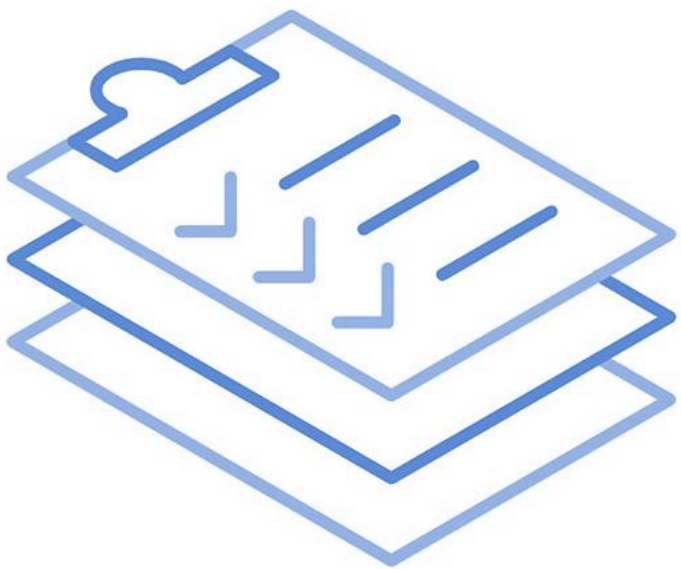


Tensorflow Data Validation



Tensorflow Data Validation (TFDV)

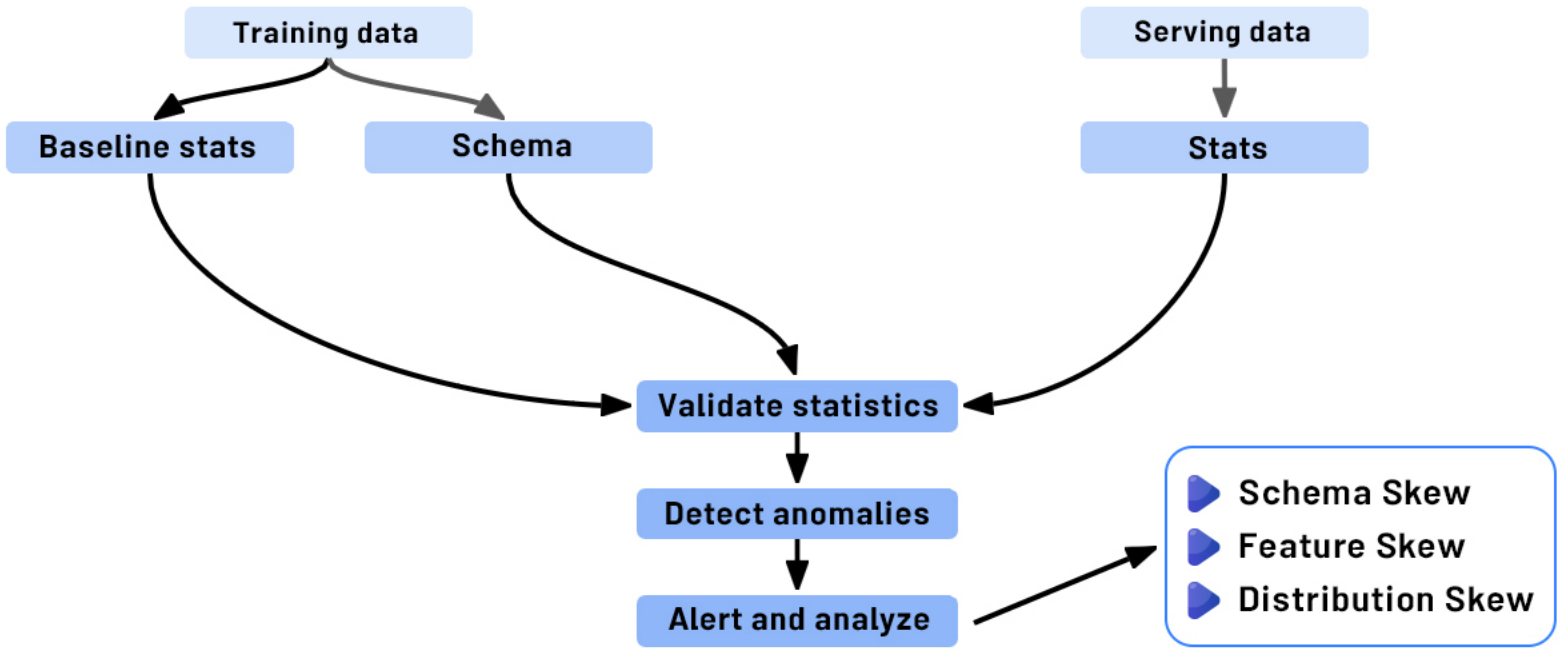
- Understand, validate, and monitor ML data at scale
- Used to analyze and validate petabytes of data at Google every day
- Proven track record in helping TFX users maintain the health of their ML pipelines



TFDV capabilities

- Generates data statistics and browser visualizations
- Infers the data schema
- Performs validity checks against schema
- Detects training/serving skew

Skew detection - TFDV



Schema skew

- **Serving and training data don't conform to same schema:**
  - For example, int != float

Feature skew

- **Training feature values are different than the serving feature values:**
  - Feature values are modified between training and serving time
  - Transformation applied only in one of the two instances

Distribution skew

- **Distribution** of serving and training dataset is significantly different:
  - Faulty sampling method during training
  - Different data sources for training and serving data
  - Trend, seasonality, changes in data over time