Machine Learning Data Lifecycle in Production

MLOps Specialization Course 2

In the second course of Machine Learning Engineering for Production Specialization, you will build data pipelines by gathering, cleaning, and validating datasets and assessing data quality; implement feature engineering, transformation, and selection with TensorFlow Extended and get the most predictive power out of your data; and establish the data lifecycle by leveraging data lineage and provenance metadata tools and follow data evolution with enterprise data schemas.

Understanding machine learning and deep learning concepts is essential, but if you're looking to build an effective AI career, you need production engineering capabilities as well. Machine learning engineering for production combines the foundational concepts of machine learning with the functional expertise of modern software development and engineering roles to help you develop production-ready skills.

Week 1: Collecting, Labeling, and Validating data

The only constant in the universe is change

Compiled By	**Peter Boshra**
LinkedIn	**https://www.linkedin.com/in/peterboshra/**
GitHub	**https://github.com/PeterBushra**
Email	**Dev.PeterBoshra@gmail.com**

2.1 Week 1 Collecting, Labeling and Validating Data

- Describe the differences between ML modeling and a production ML system
- Identify responsible data collection for building a fair production ML system
- Discuss data and concept change and how to address it by annotating new training data with direct labeling and/or human labeling
- Address training data issues by generating dataset statistics and creating, comparing and updating data schemas

The importance of data

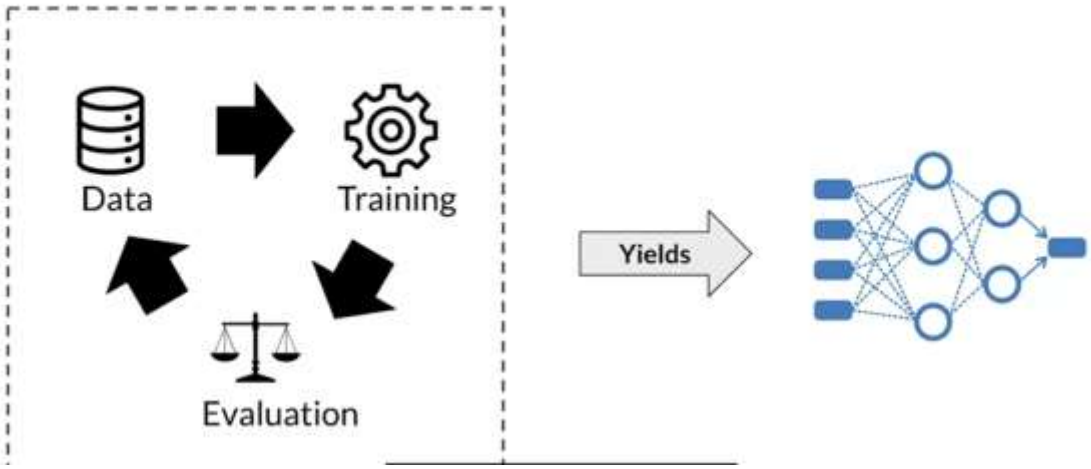
"Data is the hardest part of ML and the most important piece to get right...
Broken data is the most common cause of problems in production ML systems"
- Scaling Machine Learning at Uber with Michelangelo - Uber

"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

2.1.1 Overview

2.1.1.1 Traditional ML Model vs Production ML System

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:

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

Challenges in production grade ML

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

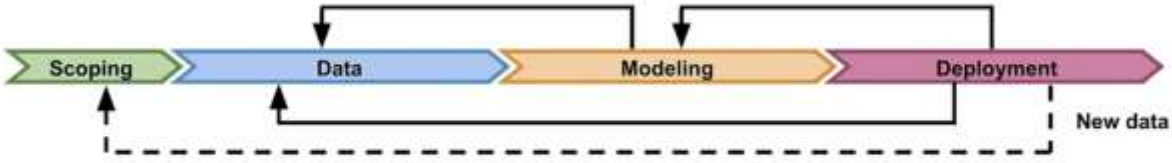
2.1.1.2 ML Pipeline

ML Pipeline are the Heart of any Production ML System

Outline

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

ML pipelines



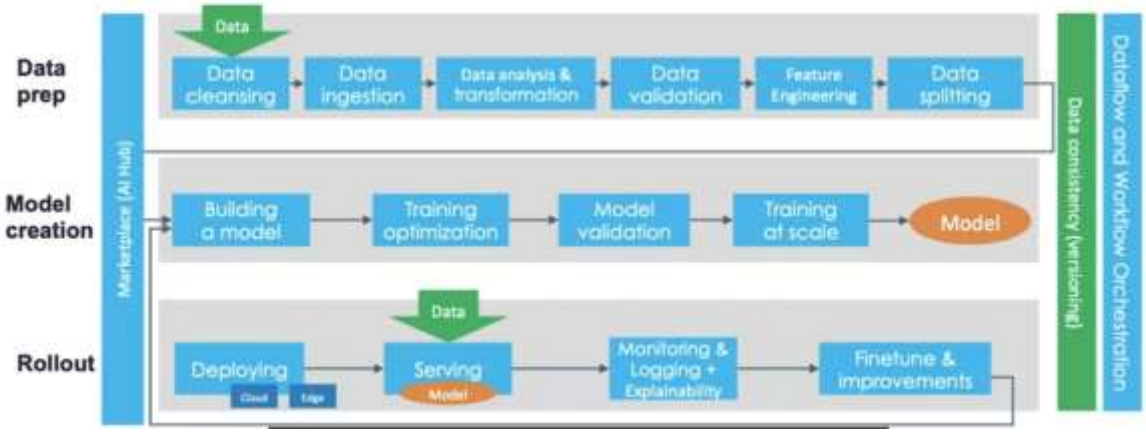
Infrastructure for

automating, monitoring, and maintaining

model training and deployment

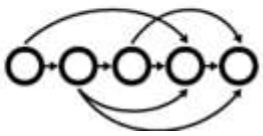
Production ML infrastructure

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.



Pipeline orchestration frameworks

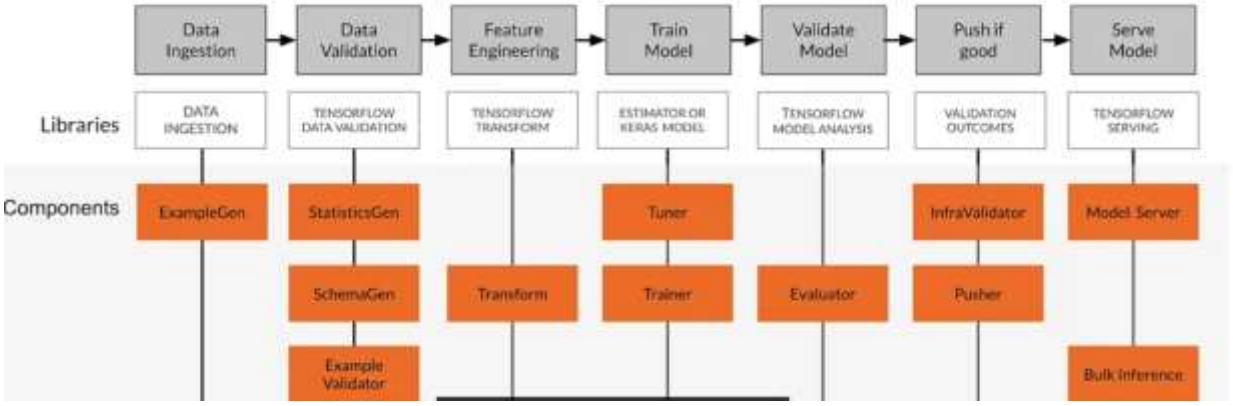
- Responsible for scheduling the various components in an ML pipeline DAG dependencies
- Help with pipeline automation

TensorFlow Extended (TFX)

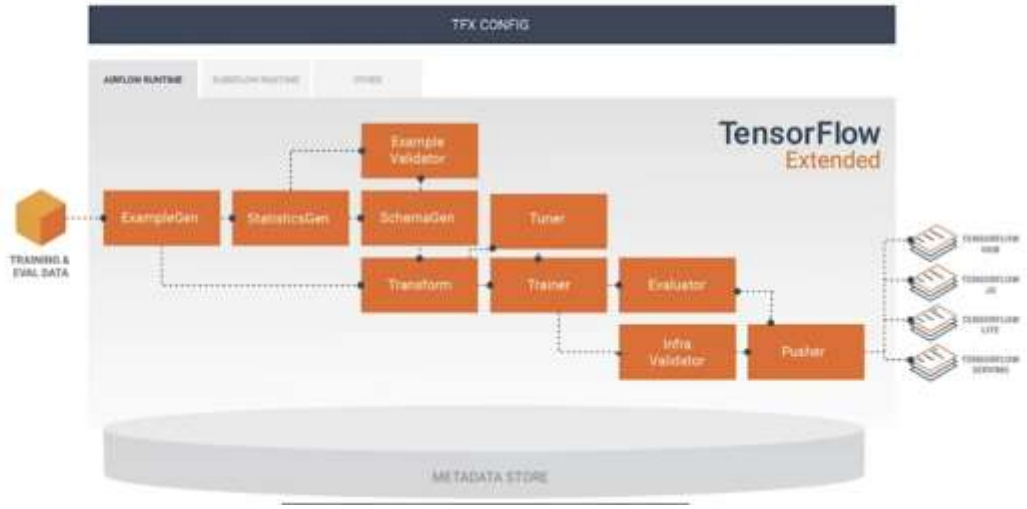
End-to-end platform for deploying production ML pipelines

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

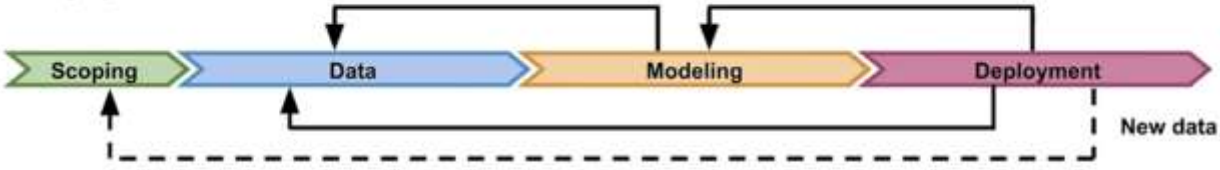
TFX production components



TFX 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

2.1.2 Collecting Data

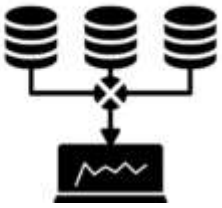
2.1.2.1 Importance of Data

Case Study: predict the amount of time that it would take to get through an airport security checkpoint

We were asked to create a model to predict the amount of time that it would take to get through an airport security checkpoint on different days at different times with different lines of different lengths and so forth. So we need a data, and how are we going to get that data, well we had to measure how long it took people to get through security checkpoints. How are we going to measure that, well, what it came up with is we had to actually go to the airport, we had to get someone cleared through security because anything you do in an airport has to be approved by the security. We had one person standing at the beginning of the line to get into a security checkpoint and they would record the time that somebody entered. And then we had another person standing at the other end, the exit of the checkpoint, and actually they were far enough away from each other. They couldn't even see each other, and they would record the time that each person left. And in that way we would gradually build up a data set of labeled data that gave us the amount of time that people took to get through an airport security checkpoint. Well, as you can imagine, it was incredibly painful, we had to develop applications just to support being able to do that. And it was incredibly expensive, we had to pay people they had to be cleared through security and so forth. So when we talk about collecting data in the real world, hopefully you're not going to deal with a situation like that. But it will be a real world situation where you need to think about how you're going to get the data you need unless, you're very lucky and somebody already has the data for you, which is great.

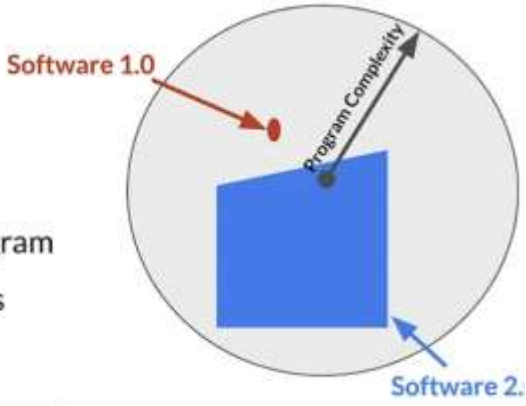
Outline

- Importance of data quality
- Data pipeline: data collection, ingestion and preparation
- Data collection and monitoring



ML: Data is a first class citizen

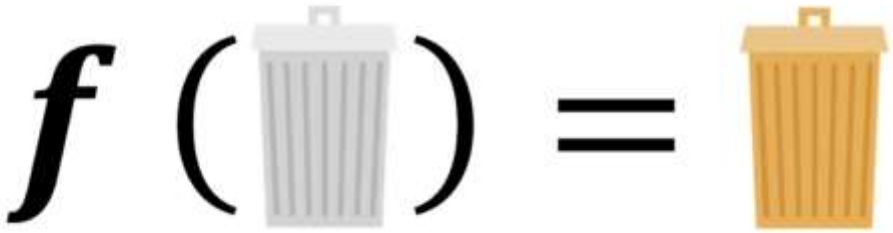
- Software 1.0
  - Explicit instructions to the computer
- Software 2.0
  - Specify some goal on the behavior of a program
  - Find solution using optimization techniques
  - Good data is key for success
  - Code in Software = Data in ML



Everything starts with data

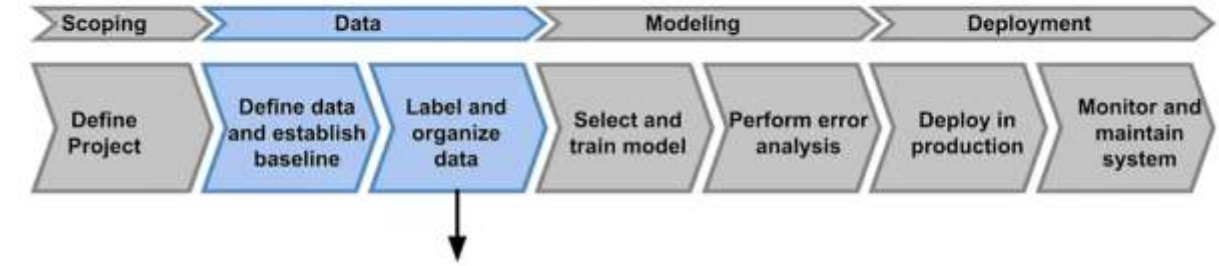
- Models aren't magic
- Meaningful data:
  - maximize predictive content
  - remove non-informative data
  - feature space coverage

Garbage in, garbage out



So if your data is garbage, if your data quality is low your model and your application will, be low quality. The good news actually for ML is that we can measure that in a lot of software applications. You, it might not be as easy to measure how bad your solution or good your solution is doing, but in the end well it's it's pretty easy to measure that. So data collection is an important and critical first step to building ml systems and data. You want to avoid problems with downtime, you need to make sure that your training a model that you can scale, that you can serve predictions and you need to think about.

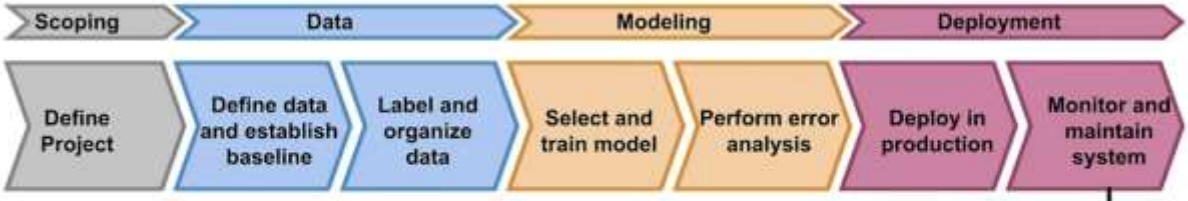
Data pipeline



- Data collection
- Data ingestion
- Data formatting
- Feature engineering
- Feature e

critical first step to building ml systems and data.

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

2.1.2.2 Example Application: Suggesting Runs

Now let's talk about collecting data and look at an example application. For this example, we're going to be looking at an application that suggests runs to runners. There's different runners with different level of fitness. The first step is really to try to understand the users as we've discussed. This system is going to suggest runs based on the user's behavior and by leveraging observed patterns and preferences. The goal is to improve the consistency of running and for runners to complete those runs and to really be happy about that.

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	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

2.1.2.3 Collecting Data [Security, Privacy Fairness]

Outline

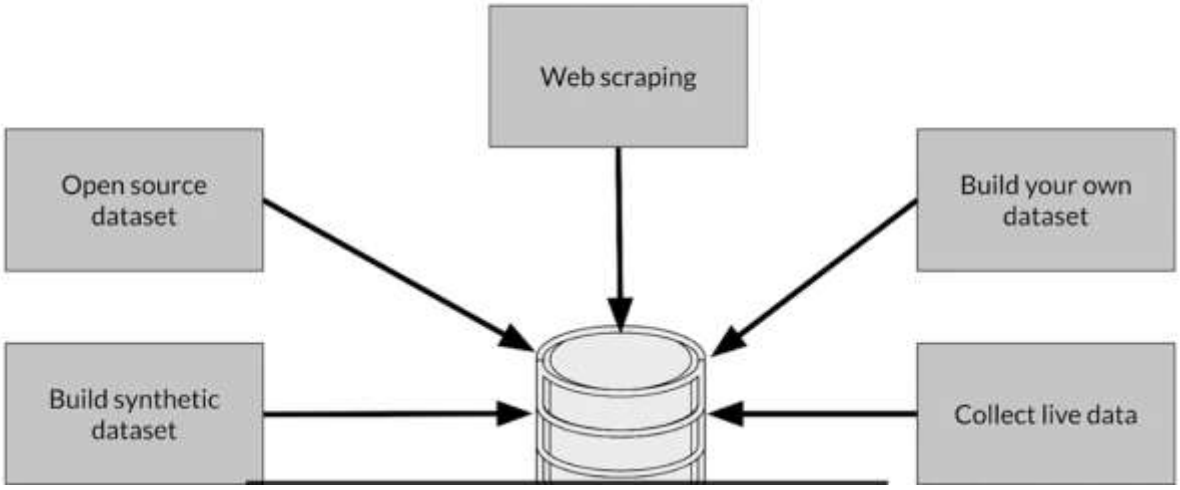
- Data Sourcing
- Data Security and User Privacy
- Bias and Fairness

Avoiding problematic biases in datasets

Example: classifier trained on the Open Images dataset

The Most Right One is Mis Labeled or Biased, it's a Ceremony in Africa.

Source Data Responsibly



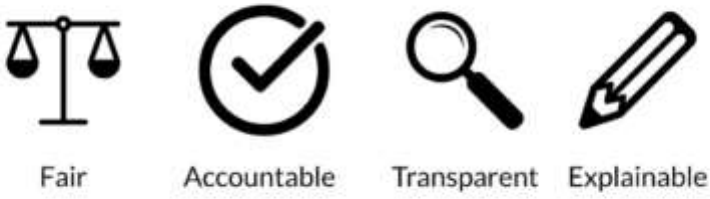
Data security and privacy

- 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



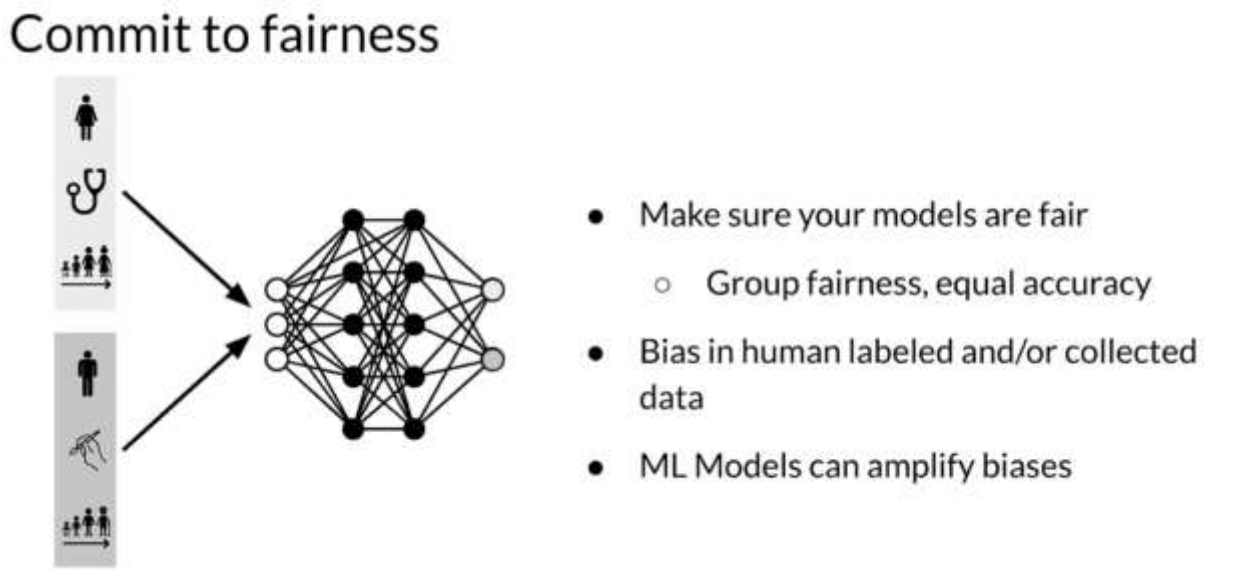
- Representational harm
- Opportunity denial
- Disproportionate product failure
- Harm by disadvantage

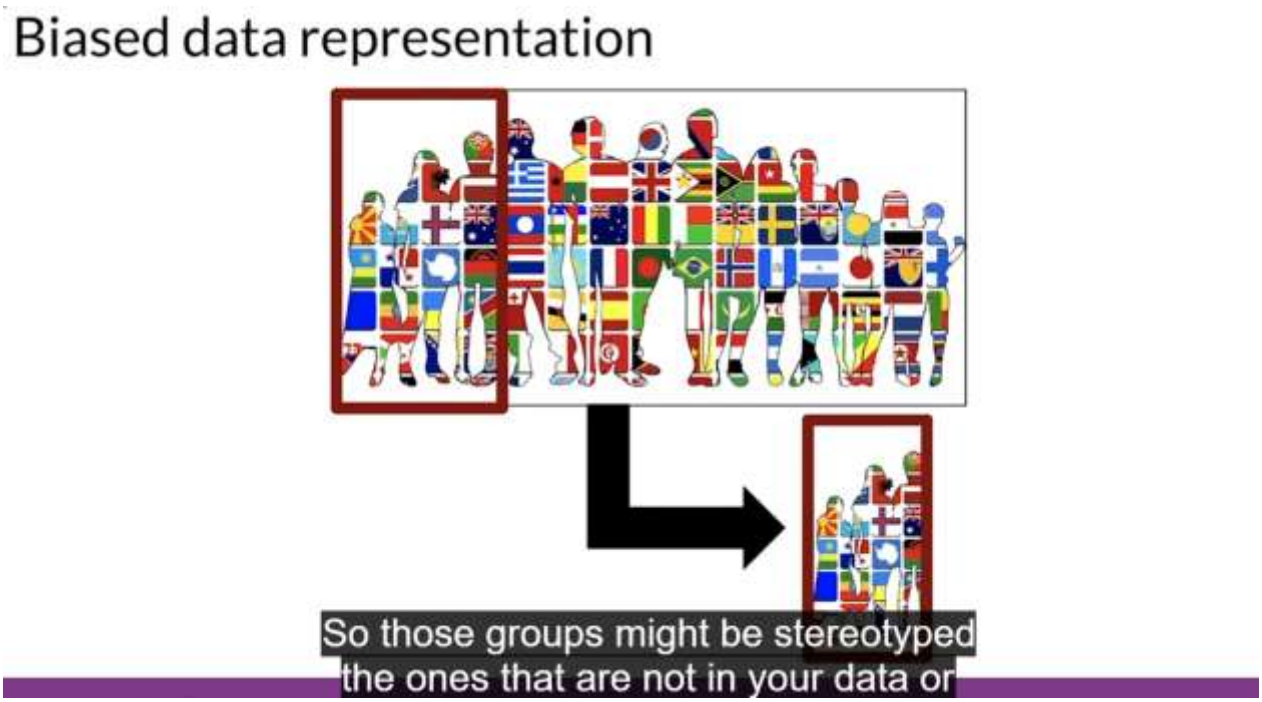
representational harm is where a system will amplify or reflect a negative stereotype about particular groups.

Opportunity denial is when a system makes predictions that have negative real-life consequences that could result in lasting impacts.

Disproportionate product failure is where the effectiveness of your model is really skewed so that the outputs happen more frequently for particular groups of users, you get skewed outputs more frequently essentially can think of as errors more frequently.

Harm by disadvantage is where a system will infer disadvantageous associations between different demographic characteristics and the user behaviors around that. So fairness is important and you should really commit to it from the beginning.



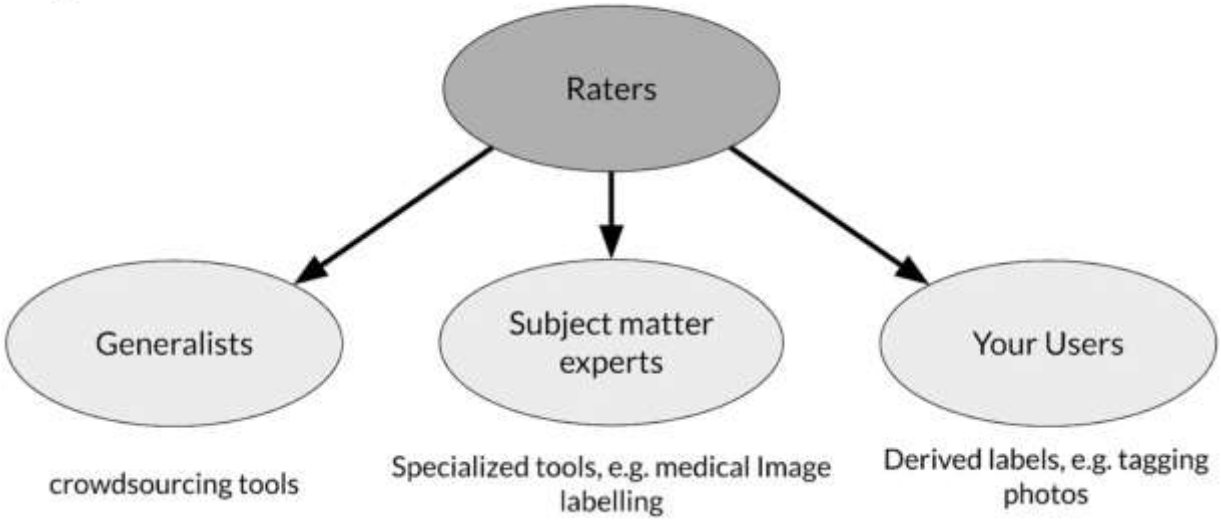


Reducing bias: Design fair labeling systems

- 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

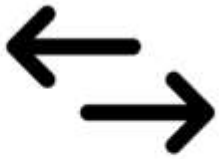
2.1.3 Labelling Data

2.1.3.1 Case Study: Degraded Model Performance

"The only constant in the universe is change"

Outline

- Detecting problems with deployed models
  - Data and concept change
- Changing ground truth
  - Easy problems
  - Harder problems
  - Really hard problems



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

Data labeling

Variety of Methods

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

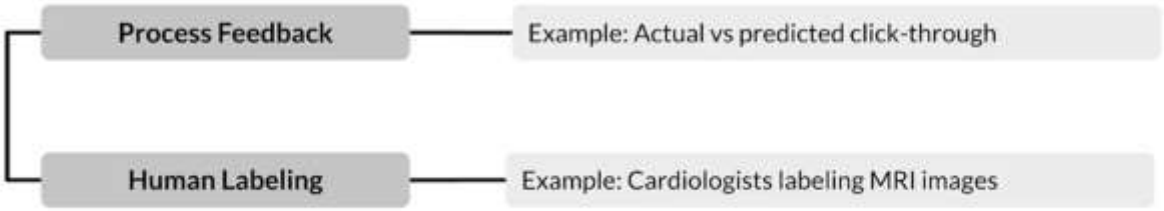
Data labeling

Variety of Methods

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

➡ Practice later as advanced labeling methods

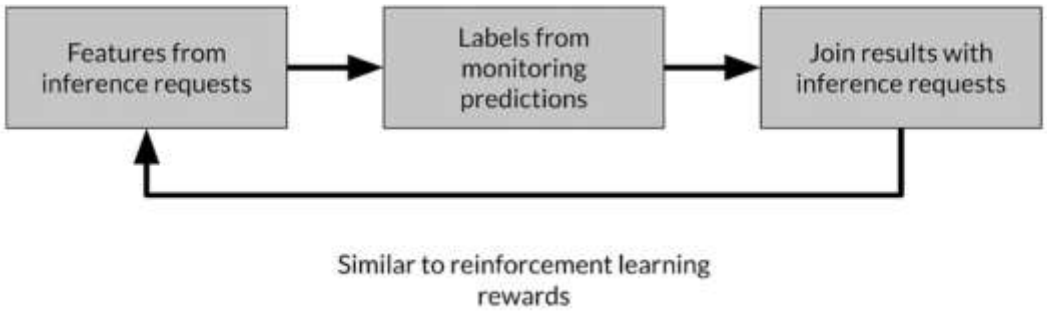
Data labeling



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

Direct labeling: continuous creation of training dataset



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

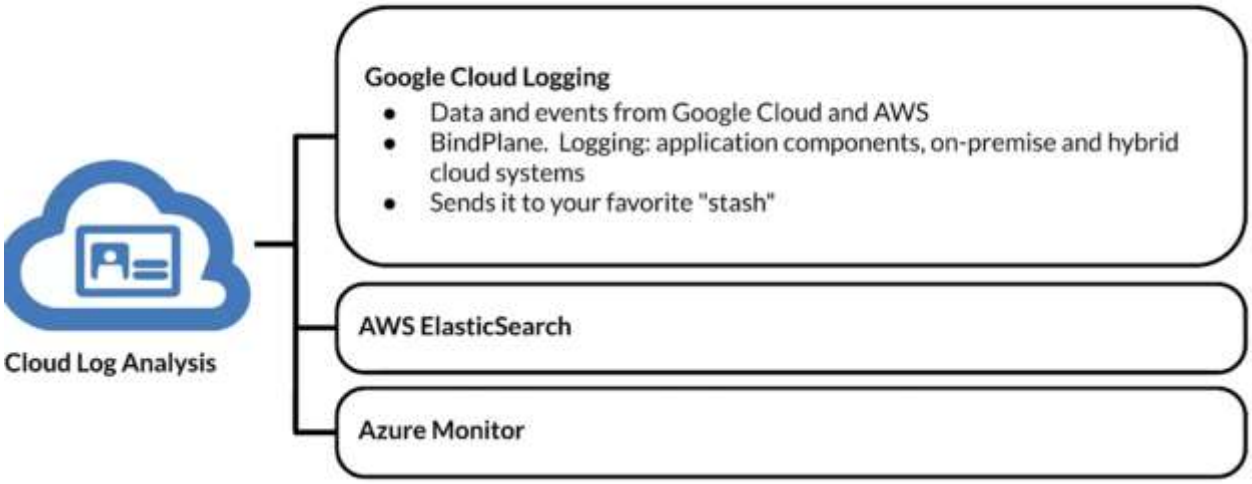
Free and open source data processing pipeline

- Ingests data from a multitude of sources
- Transforms it
- Sends it to your favorite "stash."

Fluentd

Open source data collector
Unify the data collection and consumption

Process feedback - Cloud log analytics



Human labeling

People ("raters") to examine data and assign labels manually

Human labeling - Methodology

Unlabeled data is collected

Human "raters" are recruited

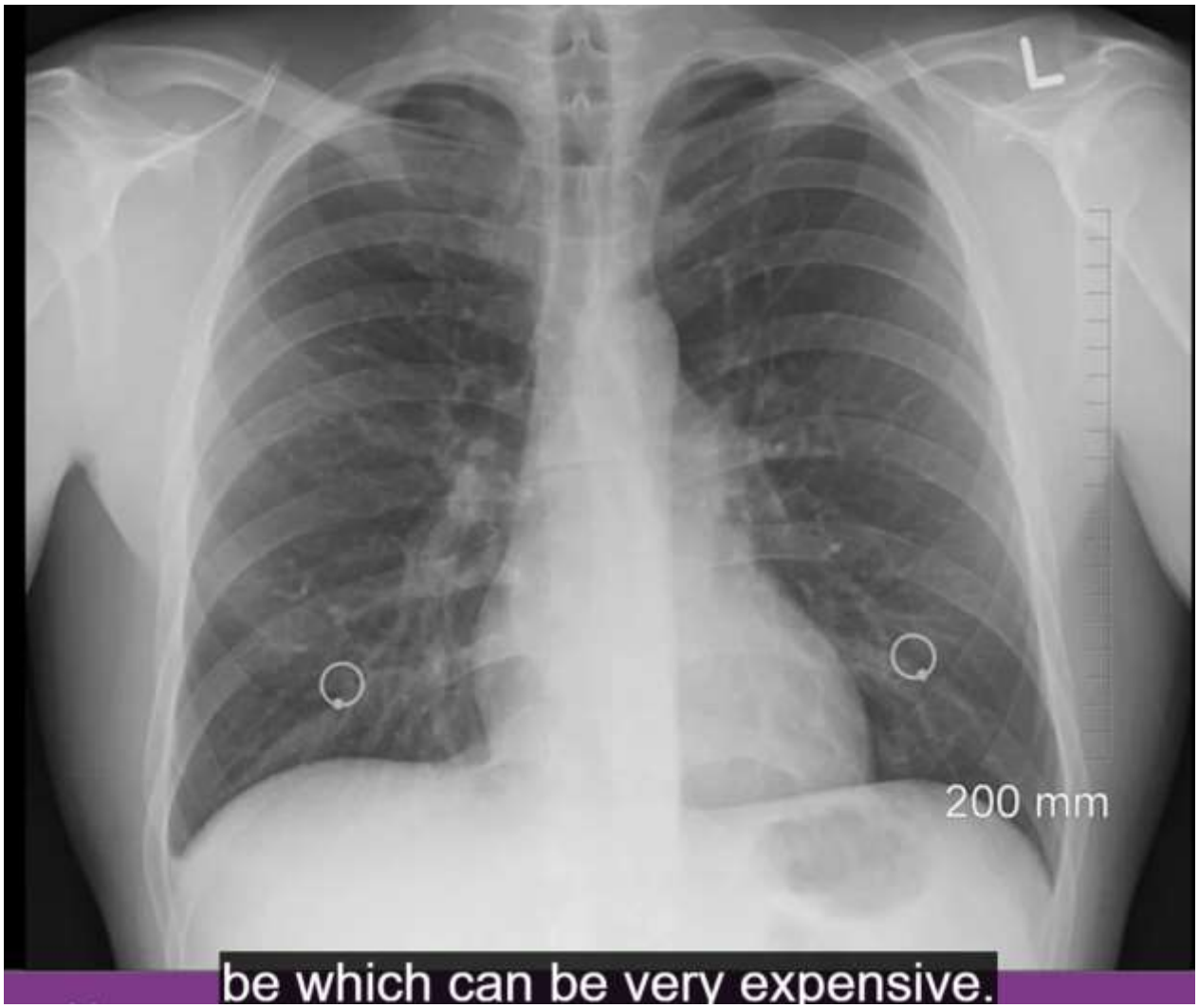
Instructions to guide raters are created

Data is divided and assigned to raters

Labels are collected and conflicts resolved

Human labeling - advantages

- More labels
- Pure supervised learning


L
200 mm
be which can be very expensive.

Human labeling - Disadvantages

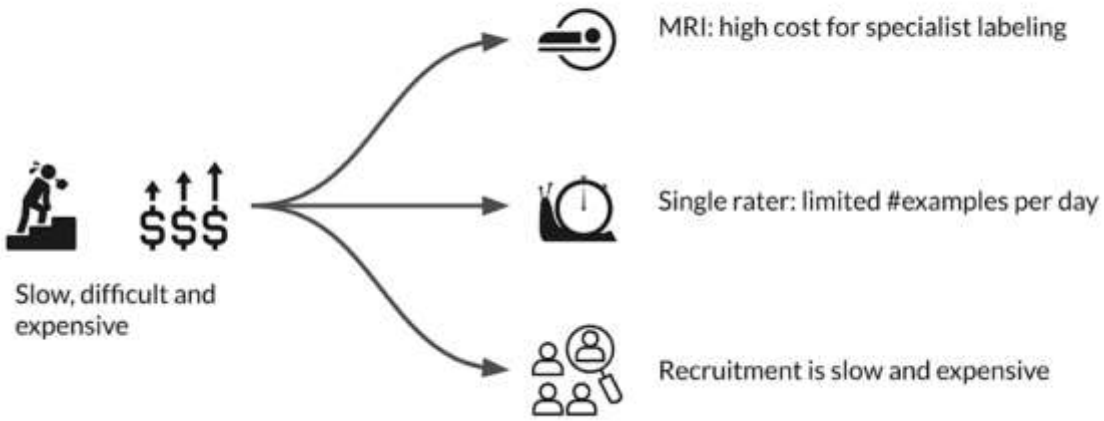
Quality consistency: Many datasets difficult for human labeling

Slow

Expensive

Small dataset curation

Why is human labeling a problem?



Key points

- Various methods of data labeling
  - Process feedback
  - Human labeling

- Advantages and disadvantages of both

2.1.4 Validating Data

Outline

- Data issues
  - Drift and skew
    - Data and concept Drift
    - Schema Skew
    - Distribution Skew
- Detecting data issues

Drift and skew

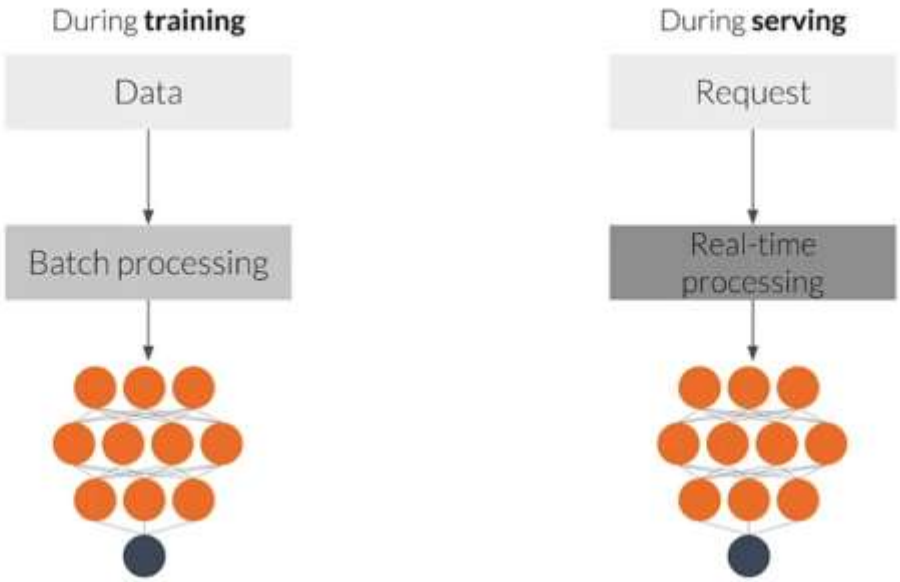
Drift

Changes in data over time, such as data collected once a day

Skew

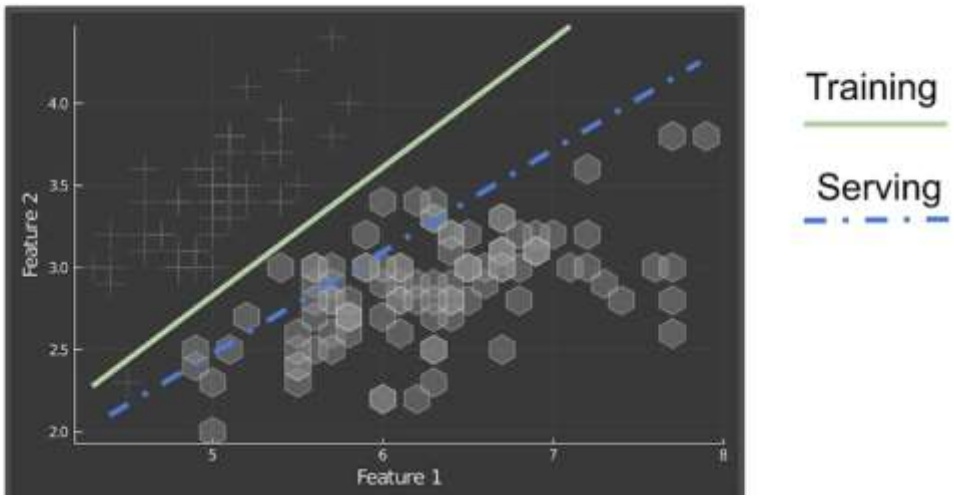
Difference between two static versions, or different sources, such as training set and serving set

Typical ML pipeline



Model Decay : Data drift

Performance decay : Concept drift



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

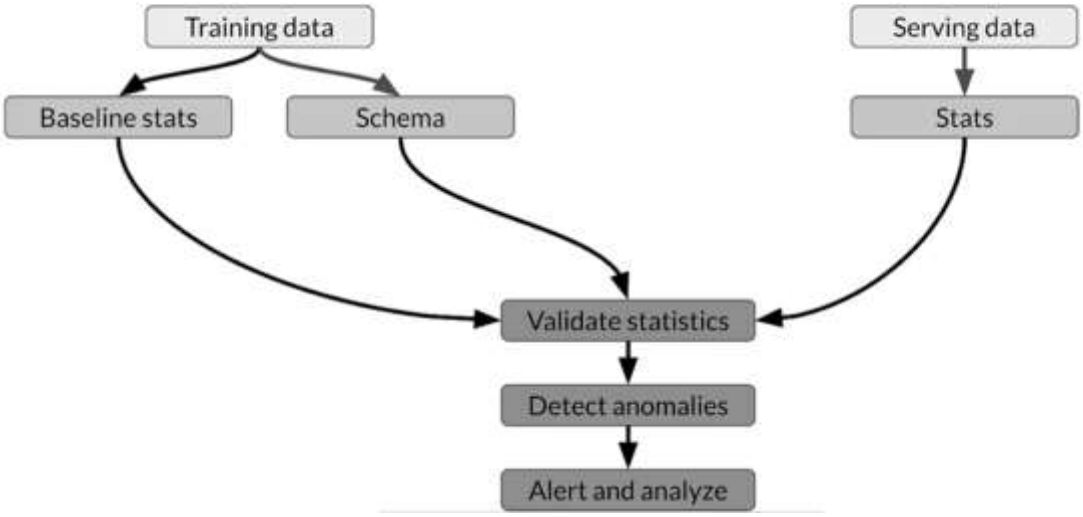
	Training	Serving
Joint	$P_{\text{train}}(y, x)$	$P_{\text{serve}}(y, x)$
Conditional	$P_{\text{train}}(y\|x)$	$P_{\text{serve}}(y\|x)$
Marginal	$P_{\text{train}}(x)$	$P_{\text{serve}}(x)$

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

Covariate shift $P_{\text{train}}(y|x) = P_{\text{serve}}(y|x)$ and $P_{\text{train}}(x) \neq P_{\text{serve}}(x)$

Concept shift $P_{\text{train}}(y|x) \neq P_{\text{serve}}(y|x)$ and $P_{\text{train}}(x) = P_{\text{serve}}(x)$

Skew detection workflow

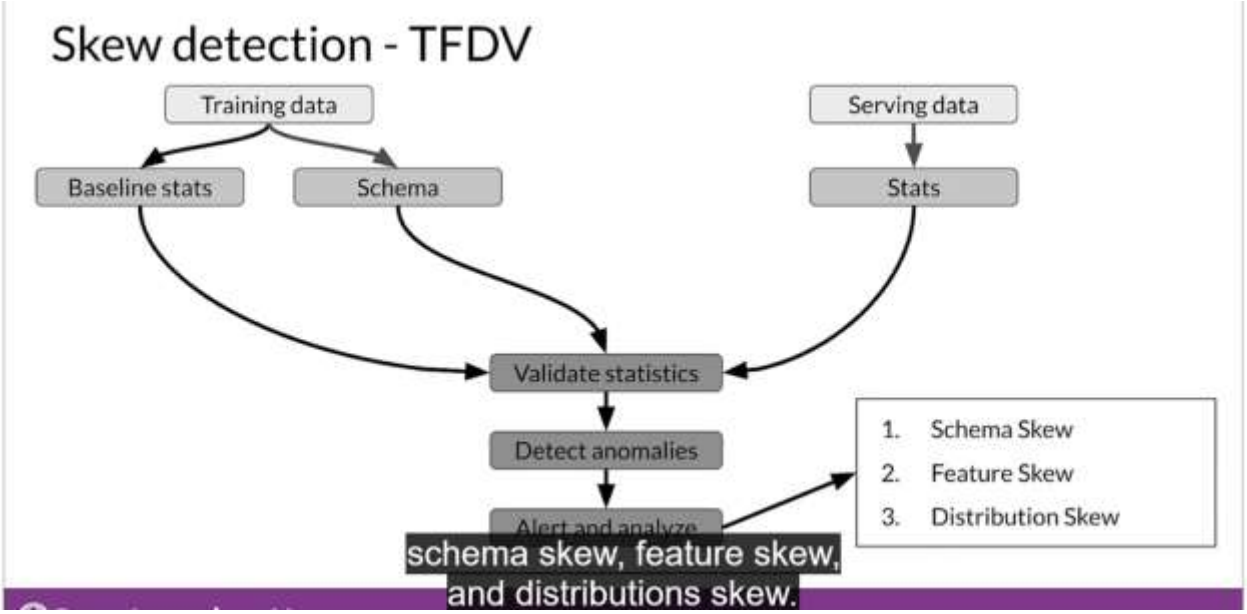


TensorFlow Data Validation (TFDV)

- 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

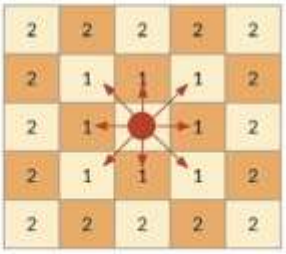
- Generates data statistics and browser visualizations
- Infers the data schema
- Performs validity checks against schema
- Detects training/serving skew



Skew - TFDV

- Supported for categorical features
- Expressed in terms of L-infinity distance (Chebyshev Distance):

$$D_{\text{Chebyshev}}(x, y) = \max_i(|x_i - y_i|)$$



- Set a threshold to receive warnings

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

Key points

- TFDV: Descriptive statistics at scale with the embedded facets visualizations
- It provides insight into:
  - What are the underlying statistics of your data
  - How does your training, evaluation, and serving dataset statistics compare
  - How can you detect and fix data anomalies

Wrap up

- Differences between ML modeling and a production ML system
- Responsible data collection for building a fair production ML system
- Process feedback and human labeling
- Detecting data issues

Practice data validation with TFDV in this week's exercise notebook

Test your skills with the programming assignment

References

- https://github.com/cdfoundation/sig-mlops/blob/master/roadmap/2020/MLOpsRoadmap2020.md
- https://karpathy.medium.com/software-2-0-a64152b37c35 [Data 1st Citizen]

Summary

of the Assignment and Exercise

- Removing Irrelevant Features
- Generate and Visualize Statistics
- Infer a Data Schema
- Compare Training and Evaluation Data
- Detect Anomalies
- Fix Evaluation Anomalies
- Check anomalies in the serving set
- Modifying the Domain
- Detect Anomalies with Specific Environment
- Check Data Drift and Skew and Setting the Alarm
- Display Stats for Data Slices
  slice the dataset and calculate the statistics for each unique value of a feature
- Freeze the Schema and Saving it