

# ECE444: Software Engineering

## Software Engineering for AI/ML 2

Shurui Zhou

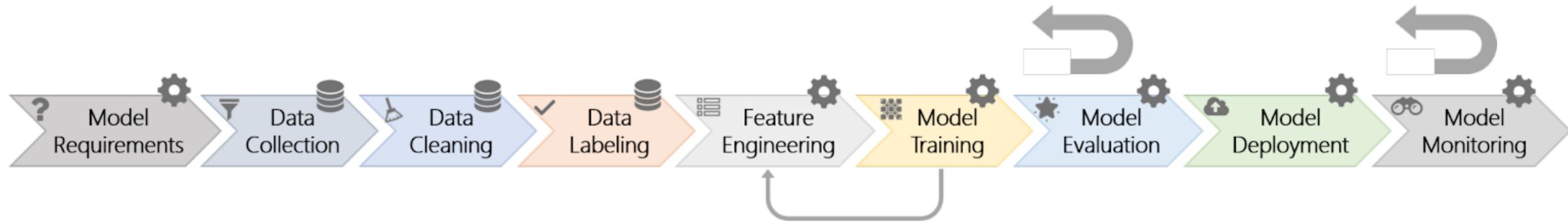


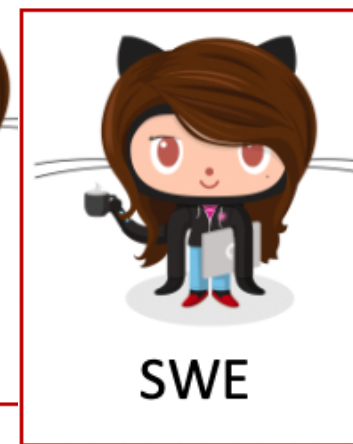
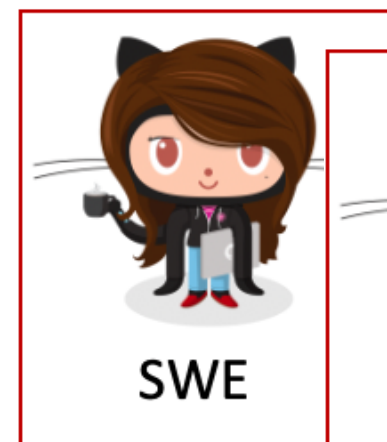
The Edward S. Rogers Sr. Department  
of Electrical & Computer Engineering  
**UNIVERSITY OF TORONTO**

# Learning Goals (last lecture)

- Understand how AI components are parts of larger systems
- Illustrate the challenges in engineering an AI-enabled system beyond accuracy
- Explain the role of specifications and their lack in machine learning and the relationship to deductive and inductive reasoning
- Summarize the respective goals and challenges of software engineers vs data scientists

# Microsoft's view of Software Engineering for ML





# AI 2020 = Software Engineering 1970s

- No unified workflow, tooling, infrastructure
- Inefficient Collaboration

# Learning Goals

- Judge when to apply AI for a problem in a system
- Critique the decision of where an AI model lives (e.g., cloud vs edge vs hybrid), considering the relevant tradeoffs
- Deliberate how and when to update models and how to collect telemetry
- Architecture Design
- QA for ML systems

# WHEN TO USE MACHINE LEARNING?

# When not to use ML/AI?

- If clear specifications are available
- Simple heuristics are *good enough*
- Cost of building and maintaining the system outweighs the benefits (see technical debt paper)
- Correctness is of utmost importance
- Only use ML for the hype, to attract funding

**Examples?**



# When to use ML/AI?

- Big problems: many inputs, massive scale
- Open-ended problems: no single solution, incremental improvements, continue to grow
- Time-changing problems: adapting to constant change, learn with users
- Intrinsically hard problems: unclear rules, heuristics perform poorly

# When to use ML/AI?

- Partial system is viable and interesting: mistakes are acceptable or mitigatable, benefits outweigh costs
- Data for continuous improvement is available: telemetry design
- Cost effective: cheaper than other approaches, meaningful benefits

**Examples?**

# Discussion: Chef Co-Pilot

A new feature with ML/AI?

Big problem? Open ended? Time changing? Hard? Partial system viable? Data continuously available? Influence objectives? Cost effective?

# System Architecture Considerations

# Case Study: Augmented Reality Translation



# Case Study: Augmented Reality Translation



# Qualities of Interest?



# Considerations

- How fast/energy consuming is model execution?
- What latency is needed for the application?
- How big is the model? How often does it need to be updated?
- Cost of operating the model? (distribution + execution)
- Opportunities for telemetry?
- What happens if users are offline?



# Architectural Decision:

- Where should the model live?

(ORC [optical character recognition] Translation

- Glasses ?
  - Phone?
  - Cloud?
- 
- What qualities are relevant for the decision ?

# Architectural Decision:

- Static intelligence in the product
- Client-side intelligence
- Server-centric intelligence
- Back-end cached intelligence
- Hybrid models

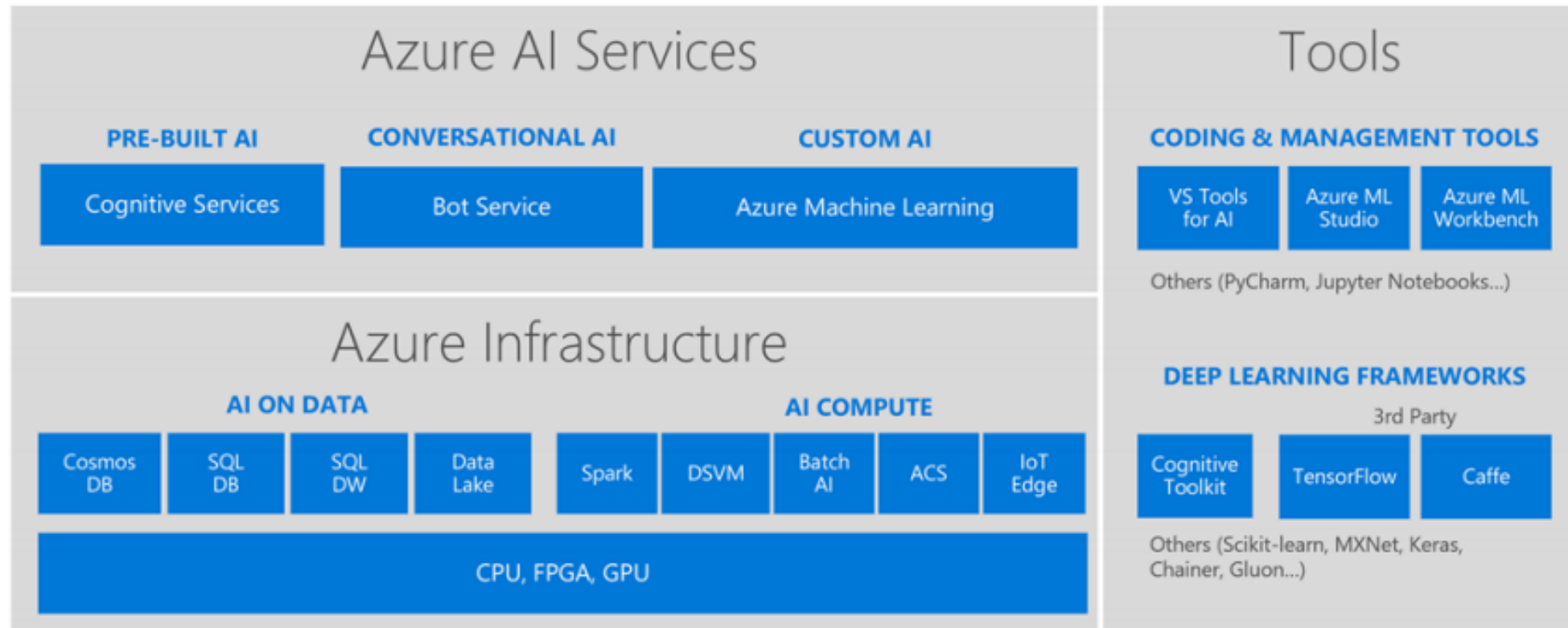
# Architectural Decision:

- Microservices?
- Coupling and Changeability?
- Anticipating and encapsulating change
  - What parts around the model service are likely to change?
  - Rigid vs flexible data formats?
- Versioning of APIs
  - Version numbers vs immutable services?
  - Expecting to run multiple versions in parallel? Implications for learning and evolution?

# AI as a Service

## The Microsoft AI platform

Cloud-powered AI for every developer



# Architecture and Patterns

- The Big Ass Script Architecture
  - Decoupled multi-tiered architecture (data vs data analysis vs reporting; separate business logic from ML)
  - Microservice architecture (multiple learning and inference services)
  - Gateway Routing Architecture
- 
- Pipelines
  - Data lake, lambda architecture
  - Reuse between training and serving pipelines
  - Continuous deployment, ML versioning, pipeline testing

Daniel Smith. "[Exploring Development Patterns in Data Science](#)." TheoryLane Blog Post. 2017.  
Washizaki, Hironori, Hiromu Uchida, Foutse Khomh, and Yann-Gaël Guéhéneuc. "[Machine Learning Architecture and Design Patterns](#)." Draft, 2019

# Whole System Perspectives

- A model is just one component of a larger system
- Also pipeline to build the model
- Also infrastructure to deploy, update, and serve the model
- Integrating the model with the rest of the system functionality
- User interaction design, dealing with mistakes
- Overall system goals vs model goals

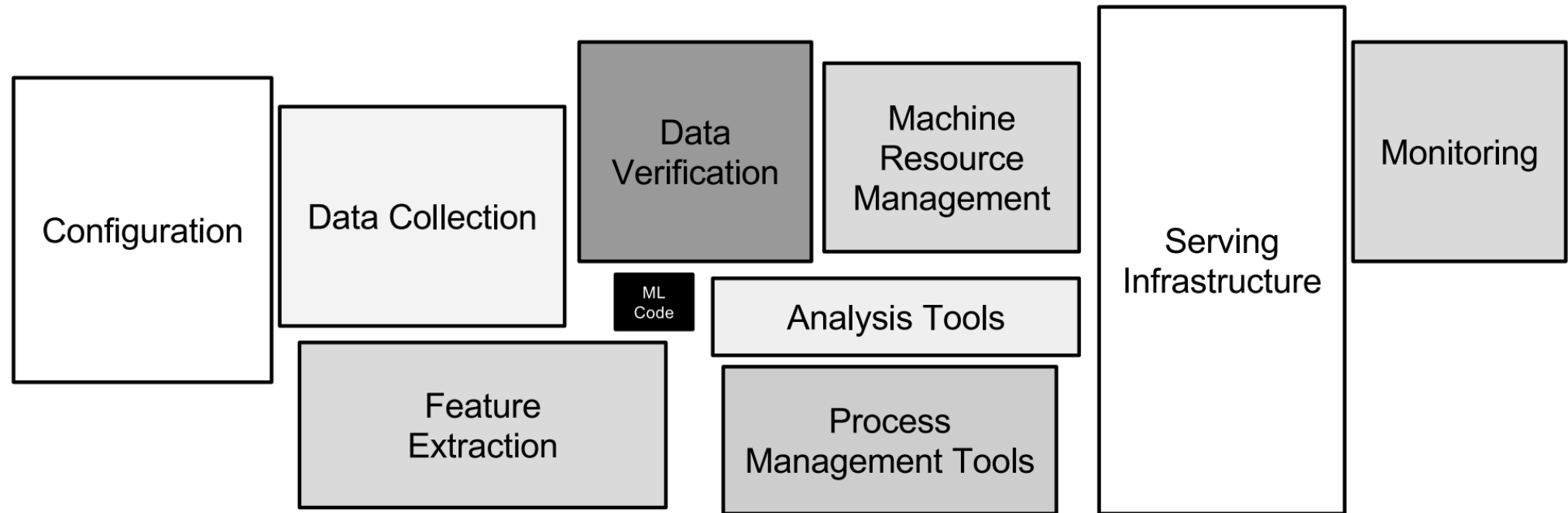


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Hidden Technical Debt in Machine Learning Systems

# QA for ML



# Data Quality

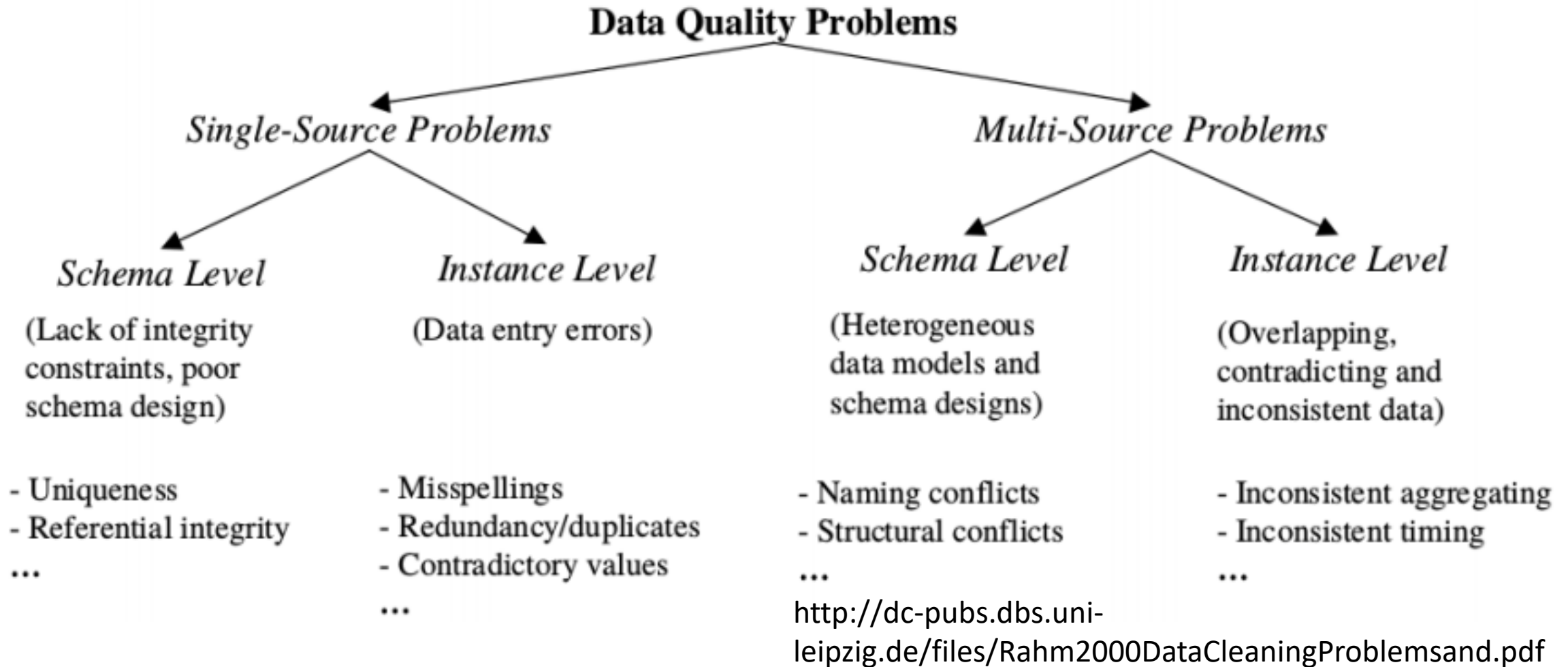
# Data is noisy

- Unreliable sensors or data entry
- Wrong results and computations, crashes
- Duplicate data, near-duplicate data
- Out of order data
- Data format invalid

# What makes good quality data?

- Accuracy
  - The data was recorded correctly.
- Completeness
  - All relevant data was recorded.
- Uniqueness
  - The entries are recorded once.
- Consistency
  - The data agrees with itself.
- Timeliness
  - The data is kept up to date.

# Data Cleaning



# Single-source problem examples:

- Schema level:
  - Illegal attribute values: `bdate=30.13.70`
  - Violated attribute dependencies: `age=22, bdate=12.02.70`
  - Uniqueness violation: `(name="John Smith", SSN="123456"), (name="Peter Miller", SSN="123456")`
  - Referential integrity violation: `emp=(name="John Smith", deptno=127)` if department 127 not defined
- Instance level:
  - Missing values: `phone=9999-999999`
  - Misspellings: `city=Pittsburg`
  - Misfielded values: `city=USA`
  - Duplicate records: `name=John Smith, name=J. Smith`
  - Wrong reference: `emp=(name="John Smith", deptno=127)` if department 127 defined but wrong

# Dirty Data

	DBAName	AKAName	Address	City	State	Zip	
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	<b>Chicago</b>	IL	<b>60608</b>	Conflicts
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60609	
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60609	
t4	<b>Johnnyo's</b>	Johnnyo's	3465 S Morgan ST	<b>Cicago</b>	IL	60608	

Does not obey data distribution

Conflict

# Data Quality

- More data -> better models (up to a point, diminishing effects)
- Noisy data (imprecise) -> less confident models, more data needed
  - some ML techniques are more or less robust to noise (more on robustness in a later lecture)
- Inaccurate data -> misleading models, biased models
- Need the "right" data
- Invest in data quality, not just quantity

<https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15>

# Data Debugging

- Validate Input Data Using a Data Schema
  - For your feature data, understand the range and distribution. For categorical features, understand the set of possible values.
  - Encode your understanding into rules defined in the schema.
  - Test your data against the data schema.
- Test Engineered Data: For example:
  - All numeric features are scaled, for example, between 0 and 1.
  - One-hot encoded vectors only contain a single 1 and N-1 zeroes.
  - Missing data is replaced by mean or default values.
  - Data distributions after transformation conform to expectations.
  - Outliers are handled, such as by scaling or clipping.



# HoloClean: Data Quality Management with Theodoros Rekatsinas



By **SE Daily**

Podcast | Tuesday, June 2 2020



00:00



00:00



<https://softwareengineeringdaily.com/2020/06/02/holoclean-data-quality-management-with-theodoros-rekatsinas/>

## Input

Dataset to be cleaned

	DBAName	Address	City	State	Zip
t1	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60609
t3	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60609
t4	Johnnyo's	3465 S Morgan ST	Cicago	IL	60608

### Denial Constraints

c1: DBAName  $\rightarrow$  Zip  
 c2: Zip  $\rightarrow$  City, State  
 c3: City, State, Address  $\rightarrow$  Zip

### Matching Dependencies

m1: Zip = Ext\_Zip  $\rightarrow$  City = Ext\_City  
 m2: Zip = Ext\_Zip  $\rightarrow$  State = Ext\_State  
 m3: City = Ext\_City  $\wedge$  State = Ext\_State  $\wedge$  Address = Ext\_Address  $\rightarrow$  Zip = Ext\_Zip

### External Information

Ext_Address	Ext_City	Ext_State	Ext_Zip
3465 S Morgan ST	Chicago	IL	60608
1208 N Wells ST	Chicago	IL	60610
259 E Erie ST	Chicago	IL	60611
2806 W Cermak Rd	Chicago	IL	60623

## The HoloClean Framework

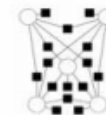
### 1. Error detection module



### 2. Automatic compilation to a probabilistic graphical model



### 3. Repair via statistical learning and inference



## Output

Proposed Cleaned Dataset

	DBAName	Address	City	State	Zip
t1	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	<b>60608</b>
t3	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	<b>60608</b>
t4	<b>John Veliotis Sr.</b>	3465 S Morgan ST	<b>Chicago</b>	IL	60608

Marginal Distribution of Cell Assignments

Cell	Possible Values	Probability
t2.Zip	60608	0.84
	60609	0.16
t4.City	Chicago	0.95
	Cicago	0.05
t4.DBAName	John Veliotis Sr.	0.99
	Johnnyo's	0.01

# Data linter

- Miscoding
  - Number, date, time as string
  - Enum as real
  - Tokenizable string (long strings, all unique)
  - Zip code as number
- Outliers and scaling
  - Unnormalized feature (varies widely)
  - Tailed distributions
  - Uncommon sign
- Packaging
  - Duplicate rows
  - Empty/missing data

<https://github.com/brain-research/data-linter>

# Model Quality


# Model Debugging

- Check that the data can predict the labels.
  - Use 10 examples from your dataset that the model can easily learn from. Alternatively, use synthetic data.
- Establish a baseline
  - Use a linear model trained solely on most predictive feature
  - In classification, always predict the most common label
  - In regression, always predict the mean value


# Beyond data and model

# Telemetry

- A telemetry system is responsible for collecting observations about how users are interacting with your Intelligent System and sending some or all of these observations back to you.
- Main tasks:
  1. Making sure the system is working the way it is supposed to.
  2. Making sure users are getting the intended outcomes.
  3. Gathering data to create new and better intelligence.

 Skype for Business ✕

How was the call quality?



Good

Audio Issues

☐ Distorted speech

☒ Electronic feedback

☒ Background noise

☐ Muffled speech

☐ Echo

Video Issues

☐ Frozen video

☐ Pixelated video

☐ Blurry image

☐ Poor color


☒ Dark video

blog post demo

[Privacy Statement](#)


Submit


Close




Matt Millman  
Because I'm happy 😊


⋮

 Chats

 Calls


 Contacts


RECENT CHATS ▾

 Besties  
10/10/2018

EN

Elena Nilsson, Anna Davie...  
It was great talking to all of ...  
7/27/2018

 Anna Davies  
coffee awaits!  
6/26/2018

 Maarten Smenk  
📞 Missed call  
5/25/2018

MS

Maarten Smenk, Anna Dav...  
Hi, happy Monday!  
5/21/2018


Settings

Help and feedback

Report a problem

Sign out



00:00  Offset 00:00 01:31:27

Play



Back 5s

1x

Speed



Volume

## NOTES

Write your notes here

## Speaker 5 ▶ 07:44

Yeah. So there's a slight story behind that. So back when I was in, uh, Undergrad, I wrote a program for myself to measure a, the amount of time I did data entry from my father's business and I was on windows at the time and there wasn't a function called time dot [inaudible] time, uh, which I needed to parse dates to get back to time, top of representation, uh, I figured out a way to do it and I gave it to what's called the python cookbook because it just seemed like something other people could use. So it was just trying to be helpful. Uh, subsequently I had to figure out how to make it work because I didn't really have to. Basically, it bothered me that you had to input all the locale information and I figured out how to do it over the subsequent months. And actually as a graduation gift from my Undergrad, the week following, I solved it and wrote it all out.

## Speaker 5 ▶ 08:38

And I asked, uh, Alex Martelli, the editor of the Python Cookbook, which had published my original recipe, a, how do I get this into python? I think it might help

How did we do on your transcript?





# Identifying Data Drift with telemetry

- Model degradations observed with telemetry
- Telemetry indicates different outputs over time for similar inputs
- Relabeling training data changes labels
- Interpretable ML models indicate rules that no longer fit

*(many papers on this topic, typically on statistical detection)*

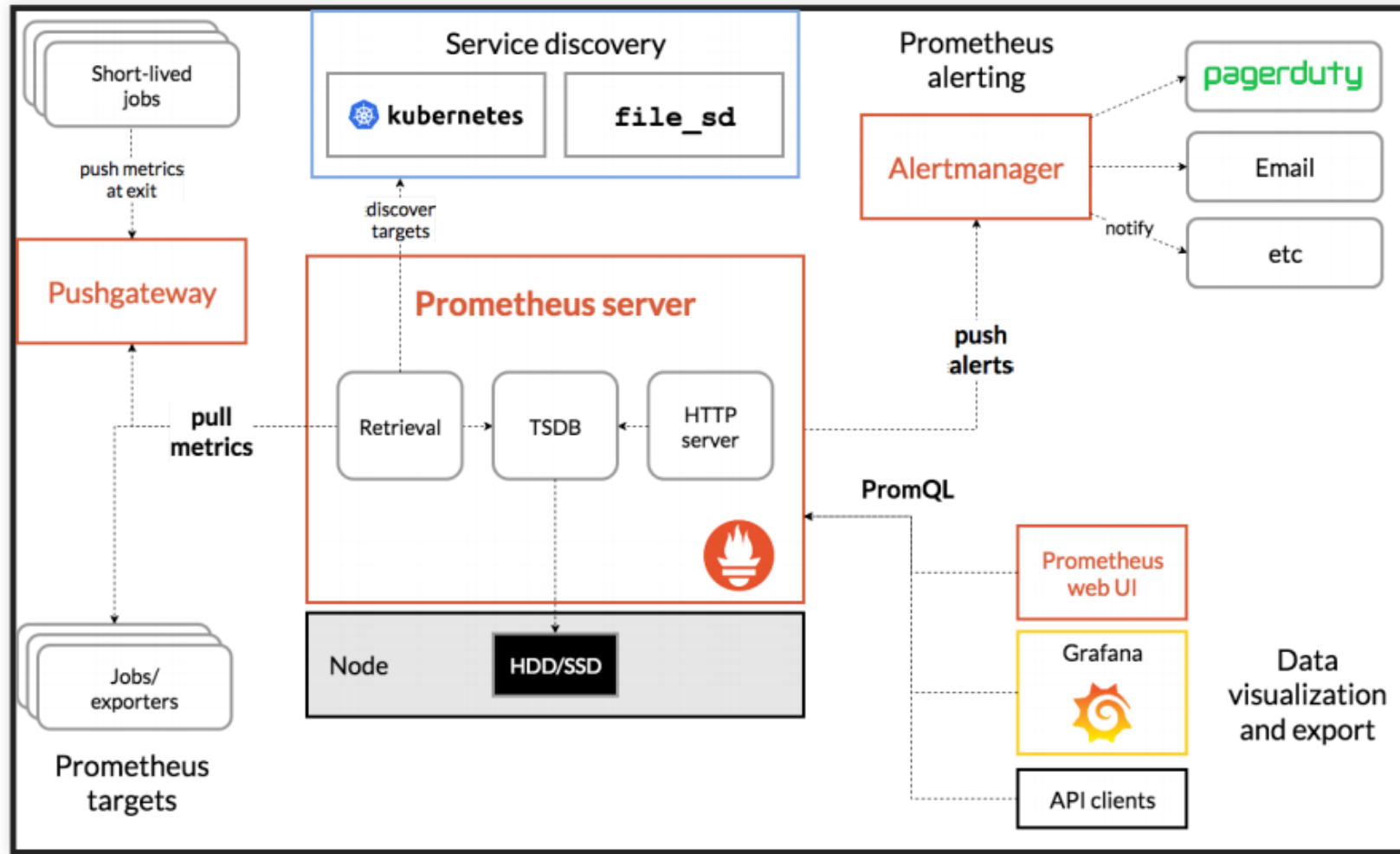
# Measuring Model Quality with telemetry

- Three steps:
  - Metric: Identify quality of concern
  - Telemetry: Describe data collection procedure
  - Operationalization: Measure quality metric in terms of data
- Telemetry can provide insights for correctness
  - sometimes very accurate labels for real unseen data
  - sometimes only mistakes
  - sometimes delayed
  - often just samples
  - often just weak proxies for correctness
- Often sufficient to approximate precision/recall or other measures
- Mismatch to (static) evaluation set may indicate stale or unrepresentative data
- Trend analysis can provide insights even for inaccurate proxy measures

# Measuring Model Quality in production

- Monitor model quality together with other quality attributes (e.g., uptime, response time, load)
- Set up automatic alerts when model quality drops
- Watch for jumps after releases
  - roll back after negative jump
- Watch for slow degradation
  - Stale models, data drift, feedback loops, adversaries
- Debug common or important problems
  - Monitor characteristics of requests
  - Mistakes uniform across populations?
  - Challenging problems -> refine training, add regression tests

# Prometheus



# Grafana





# Telemetry: Tradeoffs?

## Purpose:

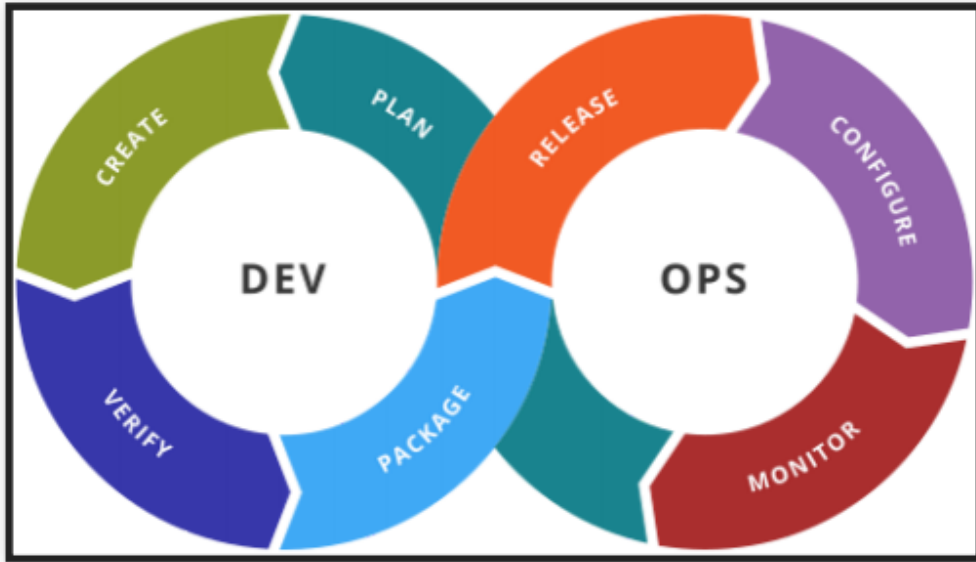
- Monitor operation
- Monitor mistakes (e.g., accuracy)
- Improve models over time (e.g., detect new features)

## Challenges:

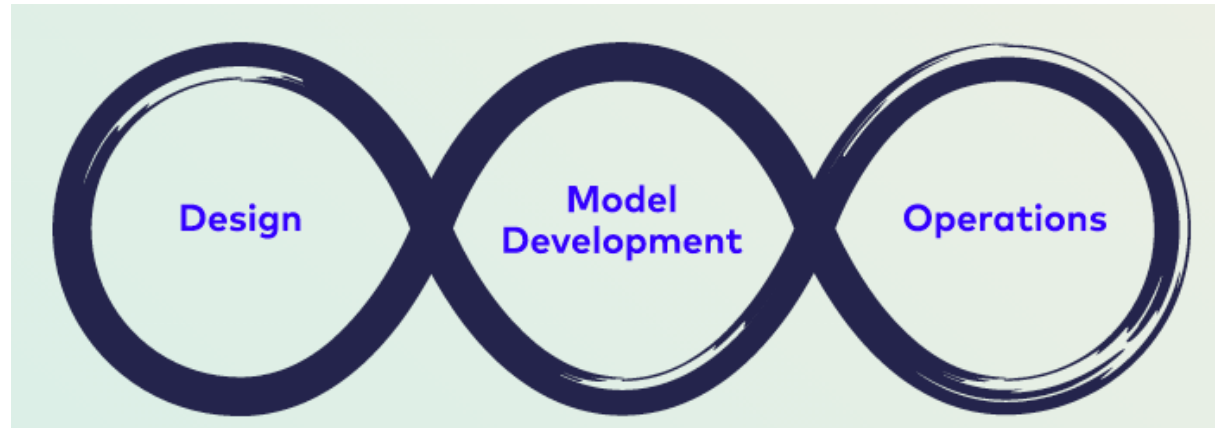
- too much data
- no/not enough data
- hard to measure, poor proxy measures
- rare events
- cost
- privacy



# DevOps



# MLOps



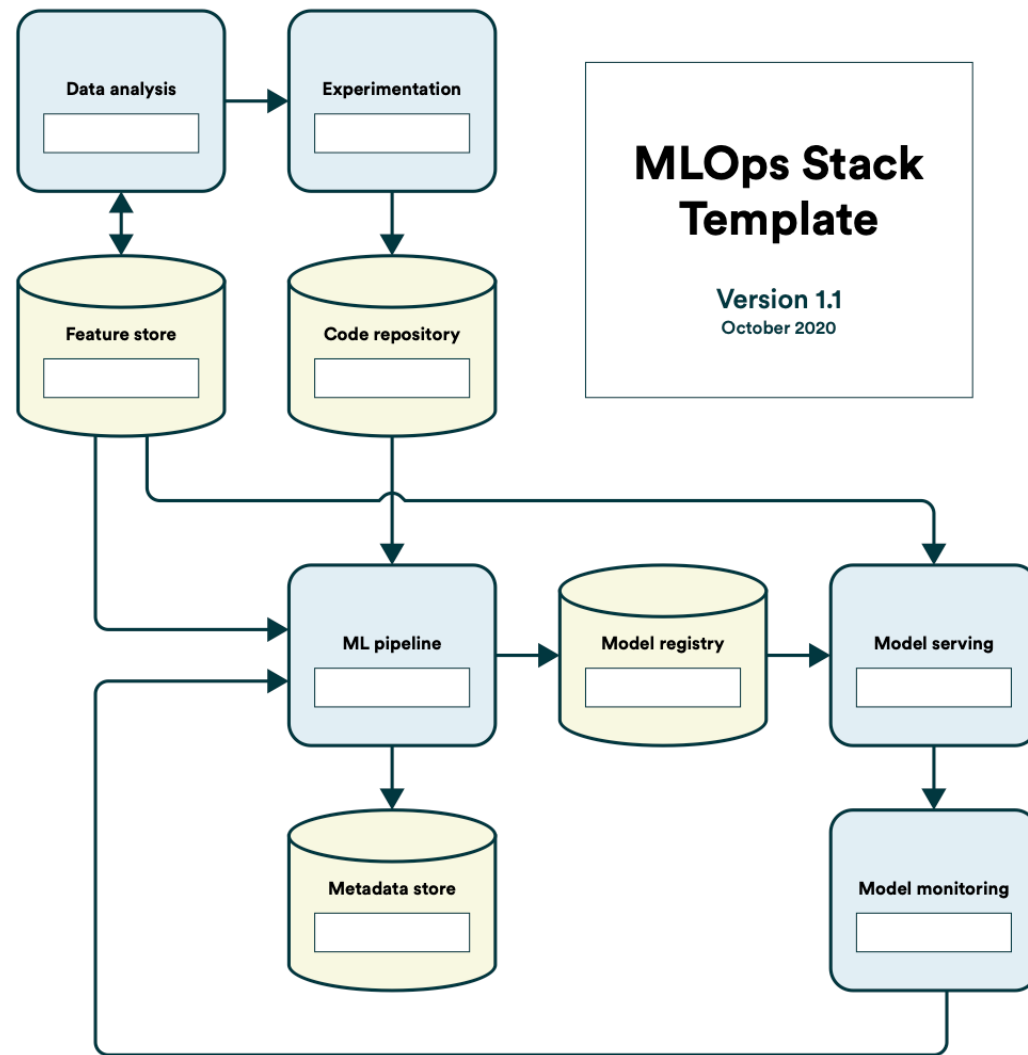
<https://ml-ops.org/>

# On Terminology

- Many vague buzzwords, often not clearly defined
- *MLOps*: Collaboration and communication between data scientists and operators, e.g.,
  - Automate model deployment
  - Model training and versioning infrastructure
  - Model deployment and monitoring
- *AI Ops*: Using AI/ML to make operations decision, e.g. in a data center
- *DataOps*: Data analytics, often business setting and reporting
  - Infrastructure to collect data (ETL) and support reporting
  - Monitor data analytics pipelines
  - Combines agile, DevOps, Lean Manufacturing ideas

# MLOps overview

- Integrate ML artifacts into software release process, unify process
- Automated data and model validation (continuous deployment)
- Data engineering, data programming
- Continuous deployment for ML models
  - From experimenting in notebooks to quick feedback in production
- Versioning of models and datasets
- Monitoring in production



<https://ml-ops.org/content/state-of-mlops>

MLOps Setup Components	Tools
Data Analysis	Python, Pandas
Source Control	Git
Test & Build Services	PyTest & Make
Deployment Services	Git, DVC
Model & Dataset Registry	DVC[aws s3]
Feature Store	Project code library
ML Metadata Store	DVC
ML Pipeline Orchestrator	DVC & Make



# Summary

- Machine learning in production systems is challenging
- Many tradeoffs in selecting ML components and in integrating them in larger system
- Plan for updates
- Manage mistakes, plan for telemetry