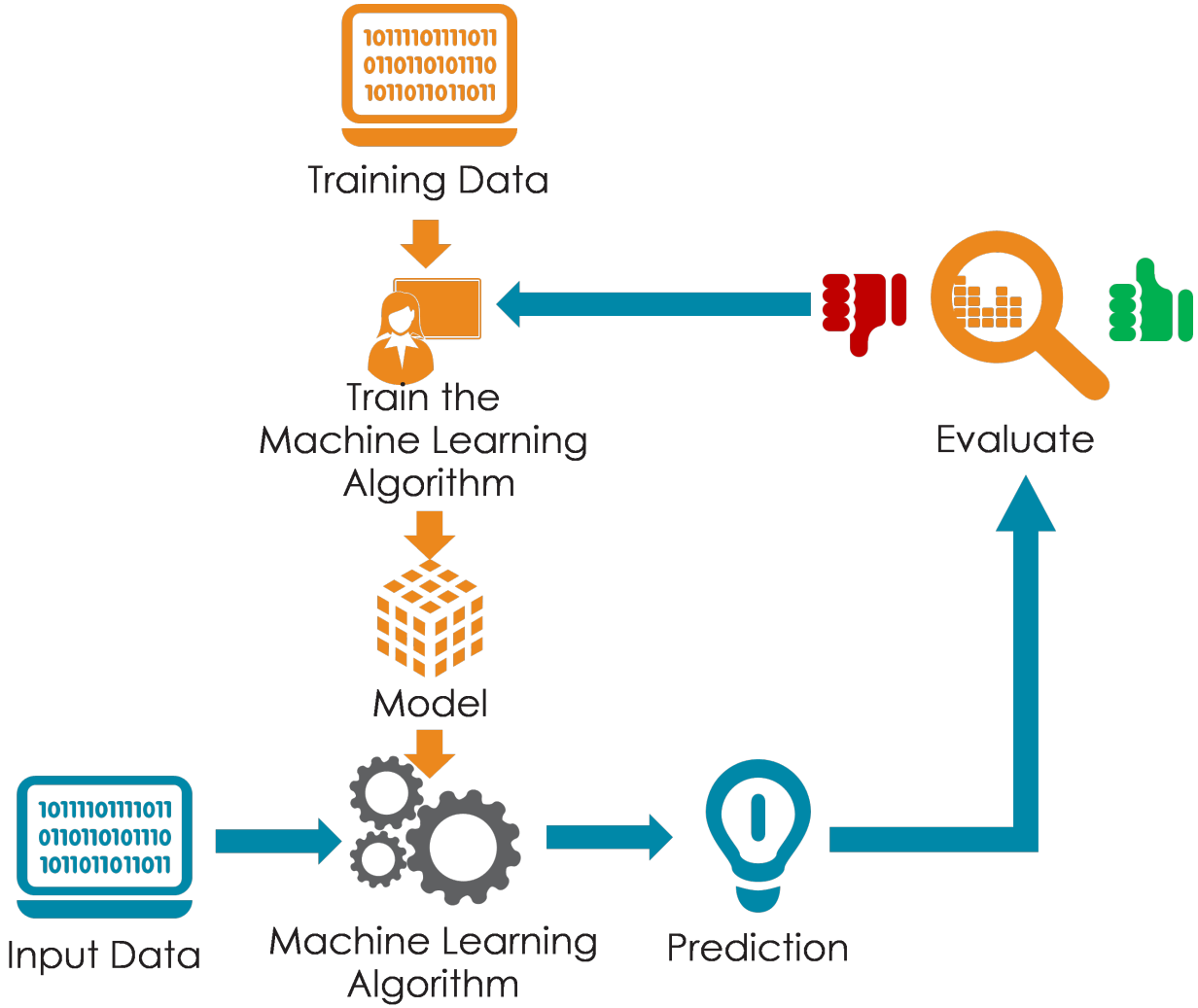
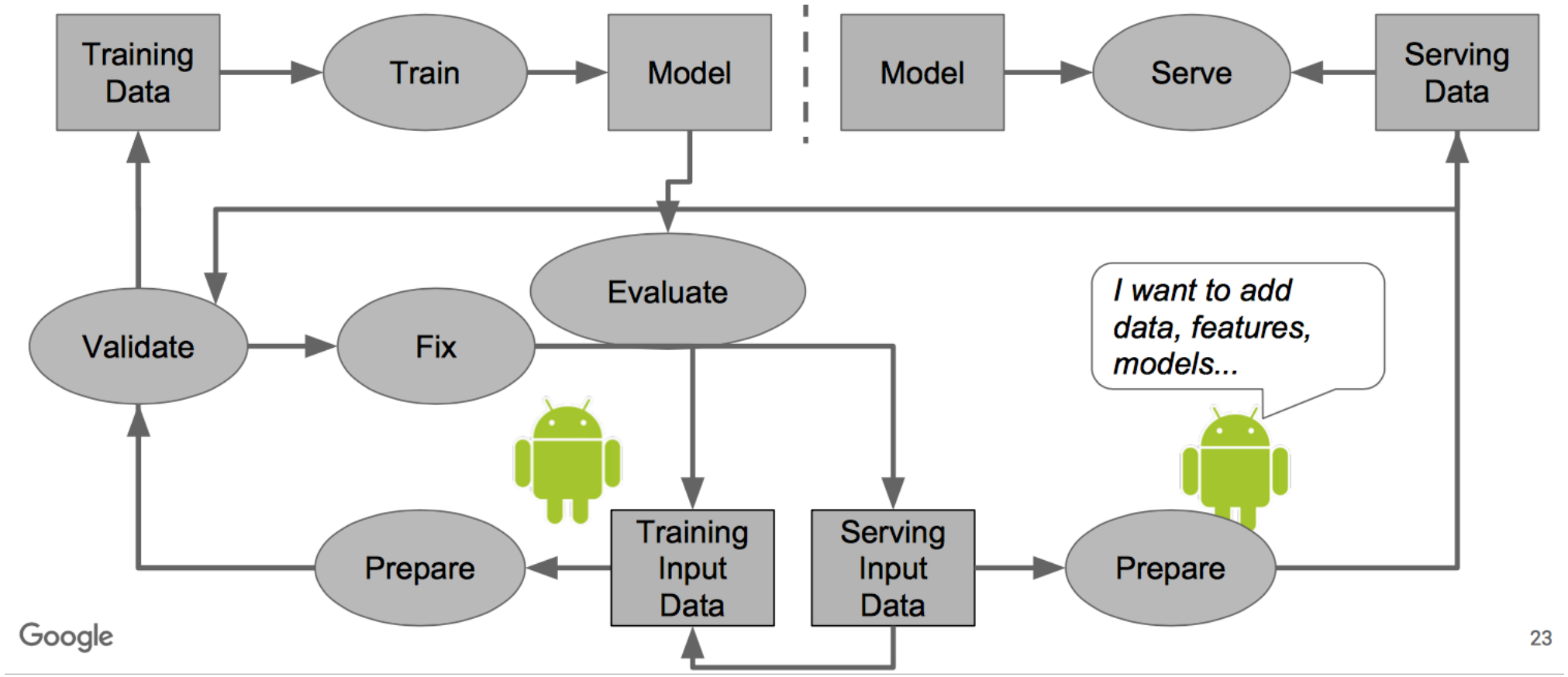
CS839: Modern Data Management and Machine Learning Systems

Lecture 1: Course Overview

A naïve ML workload

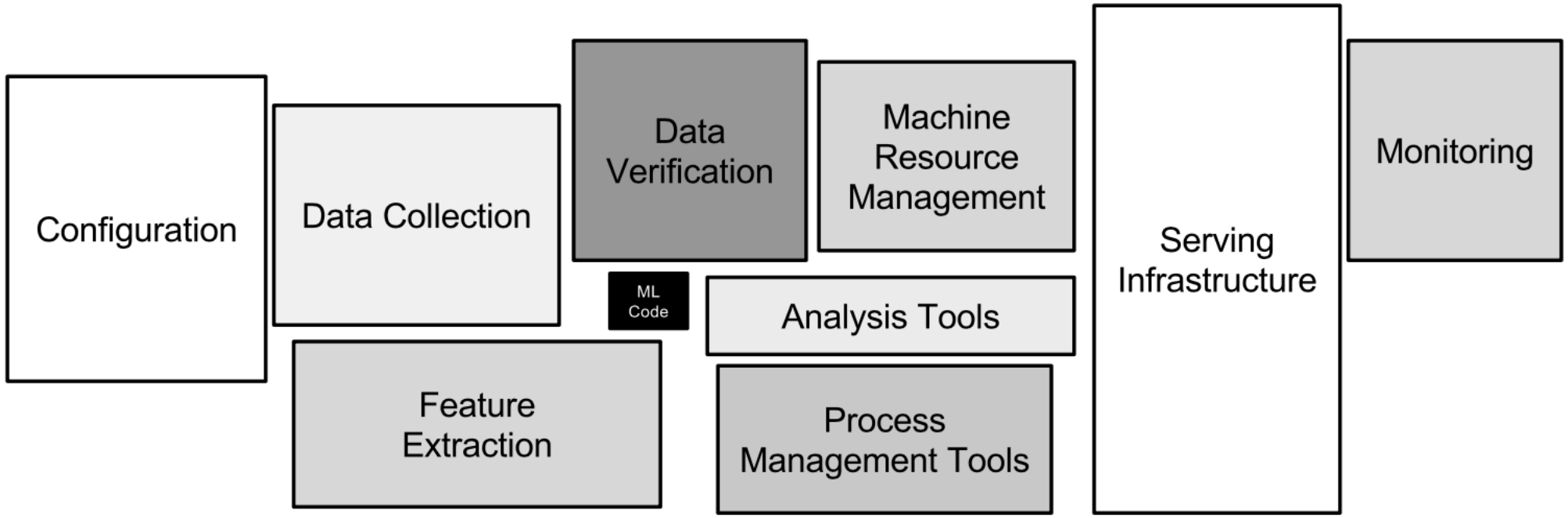


A real-life ML workload



Source: [Polyzotis et al., SIGMOD 2017]

ML-related components in real-life



Today’s Lecture

1. Introduction, Admin & Logistics

2. Course Overview

1. Introduction, Admin & Logistics

Why this class?

- **Intellectual**:
  - Combine ideas from CS, statistics, optimization, etc.
  - The coolest new thing!

Call for Submissions to the Conference on Machine Learning and Systems (MLSys)

Authors are encouraged to submit previously unpublished research at the intersection of computer systems and machine learning. The MLSys Program Committee will select papers based on a combination of novelty, quality, interest, and impact.

Topics of interest include, but are not limited to:

- Efficient model training, inference, and serving
- Distributed and parallel learning algorithms
- Privacy and security for ML applications
- Testing, debugging, and monitoring of ML applications
- Fairness and interpretability for ML applications
- Data preparation, feature selection, and feature extraction
- ML programming models and abstractions
- Programming languages for machine learning
- Visualization of data, models, and predictions
- Customized hardware for machine learning
- Hardware-efficient ML methods
- Machine Learning for Systems

MLSys: The New Frontier of Machine Learning Systems

Alexander Ratner, Dan Alistarh, Gustavo Alonso, David G. Andersen, Peter Bailis, Sarah Bird, Nicholas Carlini, Bryan Catanzaro, Jennifer Chayes, Eric Chung, Bill Dally, Jeff Dean, Inderjit S. Dhillon, Alexandros Dimakis, Pradeep Dubey, Charles Elkan, Grigori Fursin, Gregory R. Ganger, Lise Getoor, Phillip

Why this class?

How should software systems be designed to support the full machine learning lifecycle, from programming interfaces and data preprocessing to output interpretation, debugging and monitoring? Example questions include:

- *How can we enable users to quickly "program" the modern machine learning stack through emerging interfaces such as manipulating or labeling training data, imposing simple priors or constraints, or defining loss functions?*
- *How can we enable developers to define and measure ML models, architectures, and systems in higher-level ways?*
- *How can we support efficient development, monitoring, interpretation, debugging, adaptation, tuning, and overall maintenance of production ML applications- including not just models, but the data, features, labels, and other inputs that define them?*

Why this class?

How should hardware systems be designed for machine learning? Example questions include:

- *How can we develop specialized, heterogeneous hardware for training and deploying machine learning models, fit to their new operation sets and data access patterns?*
- *How can we take advantage of the stochastic nature of ML workloads to discover new trade-offs with respect to precision, stability, fidelity, and more?*
- *How should distributed systems be designed to support ML training and serving?*

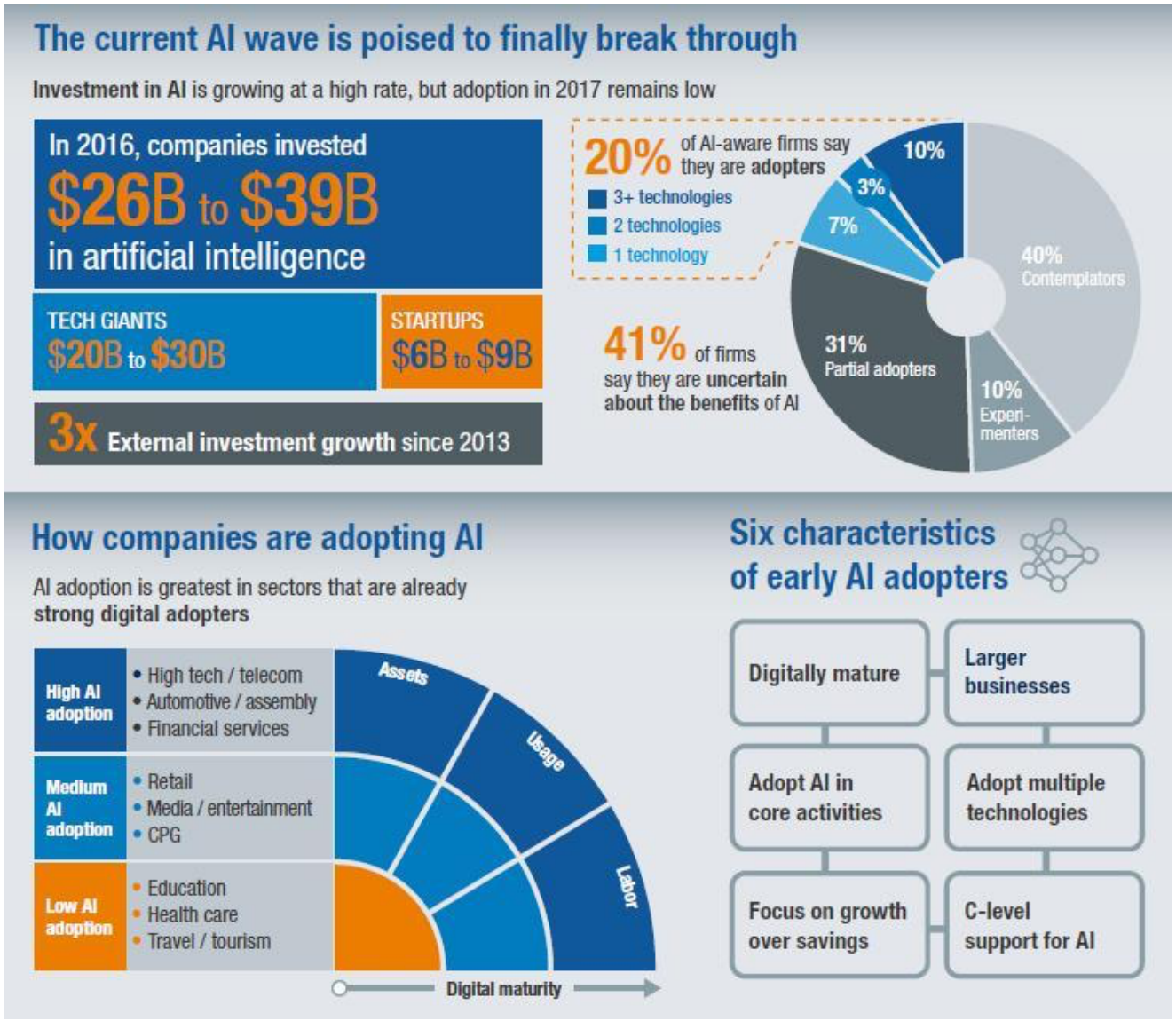
Why this class?

How should machine learning systems be designed to satisfy metrics beyond predictive accuracy, such as power and memory efficiency, accessibility, cost, latency, privacy, security, fairness, and interpretability? Example questions include:

- *How can machine learning algorithms and systems be designed for device constraints such as power, latency, and memory limits?*
- *How can ML systems be designed to support full-stack privacy and security guarantees, including, e.g., federated learning and other similar settings?*
- *How can we increase the accessibility of ML, to empower an increasingly broad range of users who may be neither ML nor systems experts?*

Why this class?

- **Mercenary-make more $$$:**



What this course is (and is not)

- Discuss **data management and systems challenges** at different stages of ML pipelines
  - How to collect data to be consumed by ML pipelines.
  - How to manage noisy data the data
  - How to serve ML models fast
  - Not how to be a Tensorflow ninja; not how to go into a Deep (learning) frenzy.
- We'll cover **algorithms and systems**
- And explore **how to build the next generation of data mangeent and ML systems**

Who am I…

Instructor (me) Theo Rekatsinas

- Faculty in the Computer Sciences and part of the UW-Database Group
- **Research**: data integration and cleaning, statistical analytics, and machine learning.
- thodrek@cs.wisc.edu
- Office hours: By appointment @CS 4361

Communication

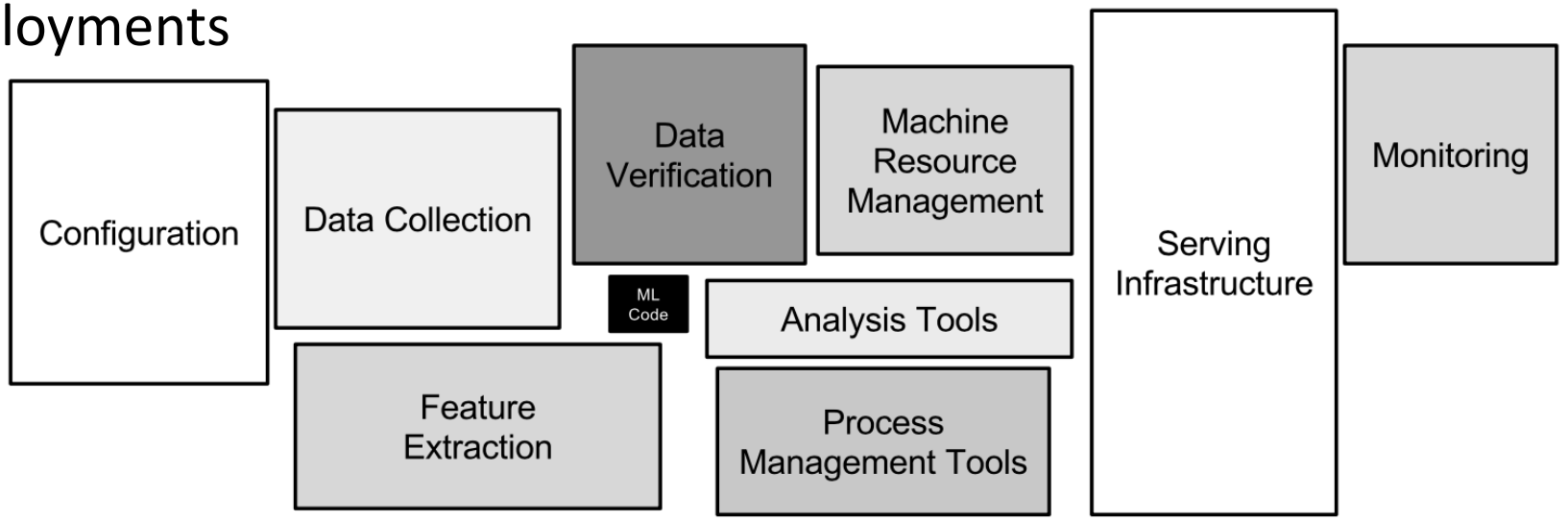
- Via email: thodrek@cs.wisc.edu
- *By appointment!*

Course Website:

https://thodrek.github.io/cs839_sp20/

Lectures and Readings

- Lectures split into weekly themes:
  1. Two-classes per theme
  2. First class: lecture by instructor (background and core-concepts)
  3. Second class: paper review, presentation, and discussion
- Each part covers one core topic in real-life ML deployments



Lectures and Readings

- **One** mandatory paper to read every week
  - 10 papers in total
- More papers posted for every topic; These will help you for your projects

- How does this work:
  - We will ask for 10 volunteers to present the paper (discussion leaders)
    - The volunteers will give a 20 minute whiteboard talk on the paper
    - If we do not have enough volunteers we will select people at random
  - We will ask for 10 volunteers as scribes
    - The scribe will need to keep minutes and summarize the discussion around the paper
  - The presentation will be followed by a detailed paper discussion
  - All people but the presenter and the scribe are required to submit a conference-style review
    - The volunteer will not have to submit a review

Conference-Style Reviews

- Summary
- 3 strong points - 3 week points
- Detailed comments
- **You need to read and understand the papers!**

Graded Elements

- Reviews and Presentations (20%)
- Project Presentation (20%)
- Final Project Report (60%)
- **No exams or other homework**

Research Project

- **What can it be?**
  - An open problem relevant to your research and the course
  - An open problem relevant to the course
  - Will post some suggestions later in the semester

- **Milestones**
  - Part 1: Proposal (due 2/25); feedback the following week
  - Part 2: Final presentations on 4/28 and 4/30;
    - Random assignment
  - Part 3: Final report (due 5/8)

- **What is a good project?**

What is expected from you

- **Attend lectures**
  - If you don't, it's at your own peril
- **Be active and think critically**
  - Ask questions, post comments
- **Do an amazing project**
  - Start early and push hard

Lectures Overview

1. Machine Learning Life-Cycle
2. Training Data Collection
3. Data Validation
4. Adaptive Data Management
5. Serving Feature Selection Workloads
6. Automated ML
7. Distributed Model Training
8. Efficient Prediction Serving
9. Model Compression
10. Relational Bias in Neural Networks

Machine Learning Life-Cycle

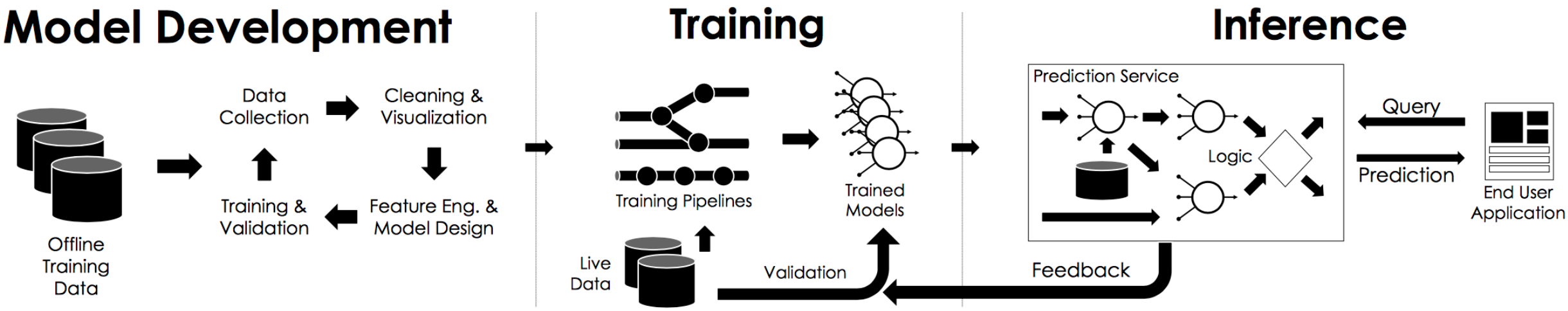
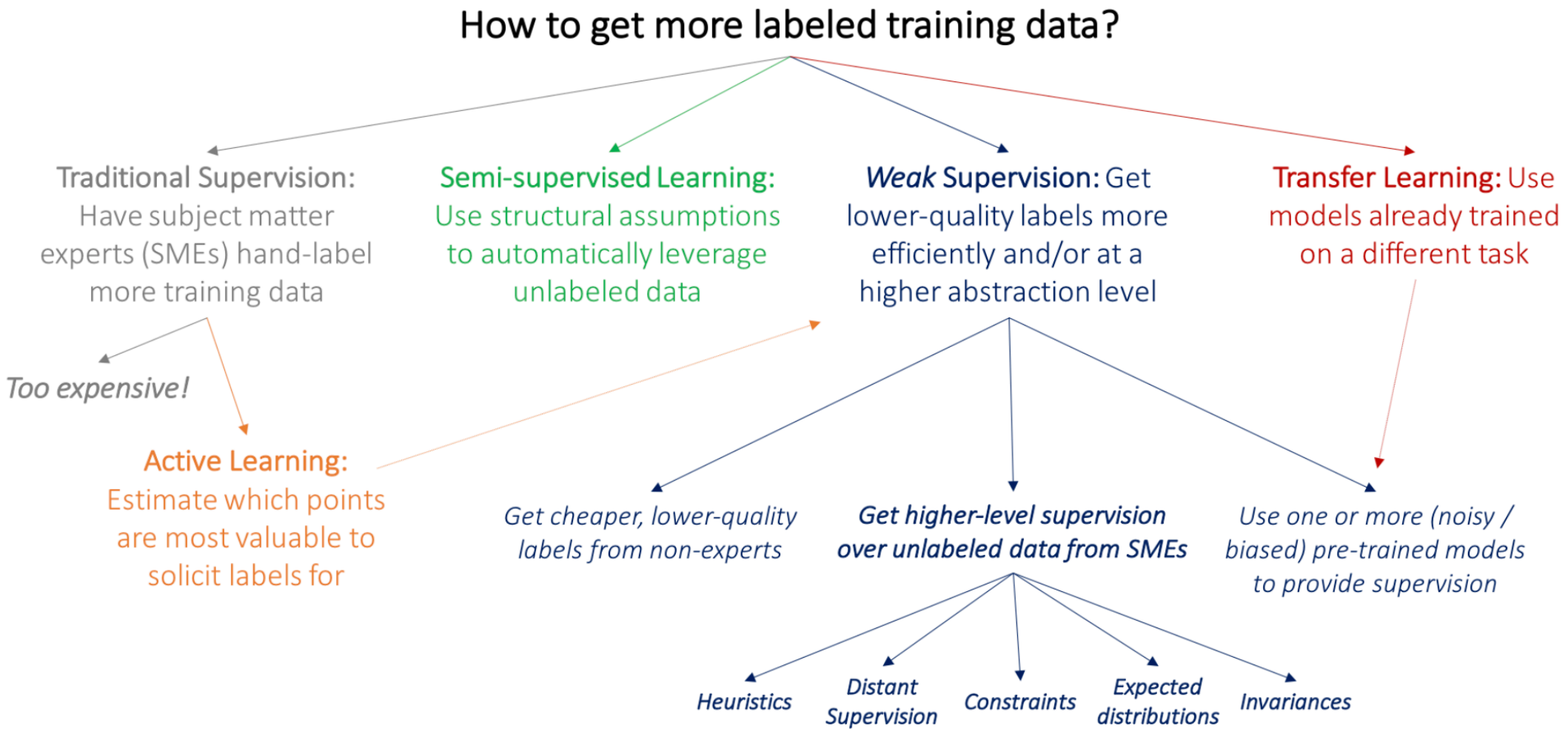


Figure 1: **Machine Learning Life-cycle.** A simplified depiction of the key stages of a machine learning application.

Source: [Gonzalez, IEEE Data Engineering Bulletin 2018]

Training Data Collection



Data Validation

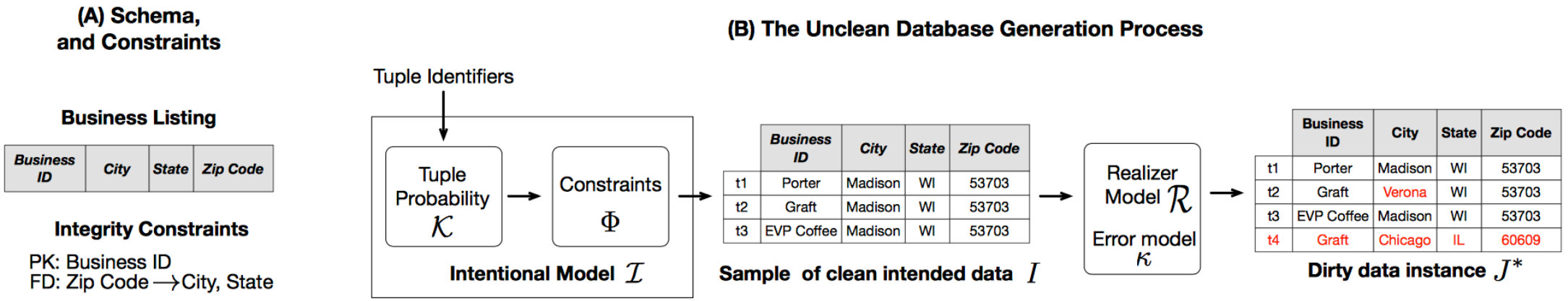
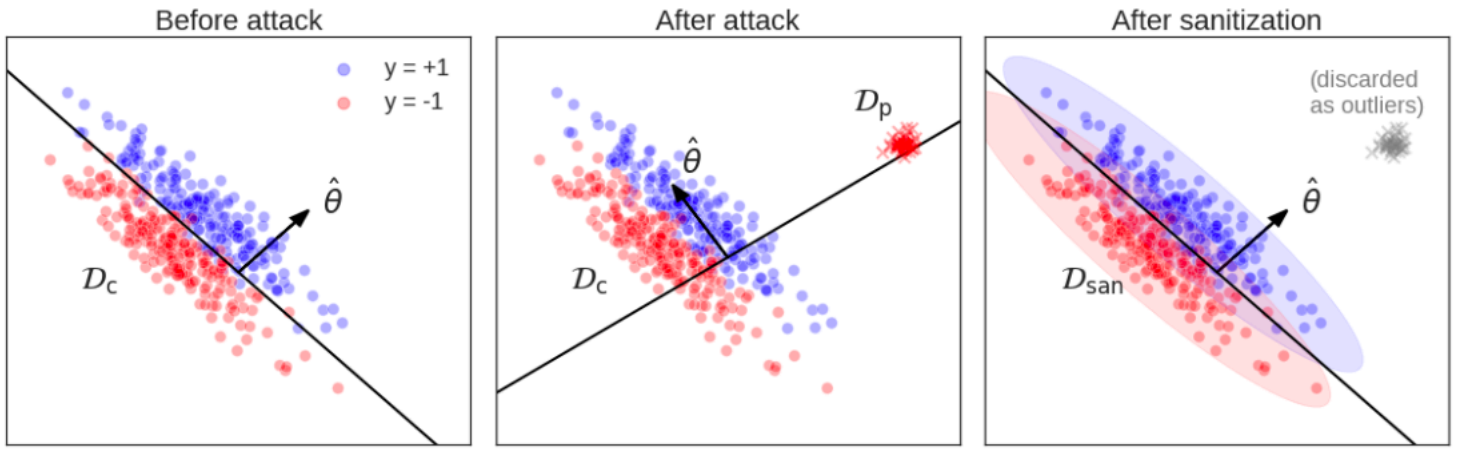


Figure 1 Overview of the PUD framework.



Adaptive Data Management

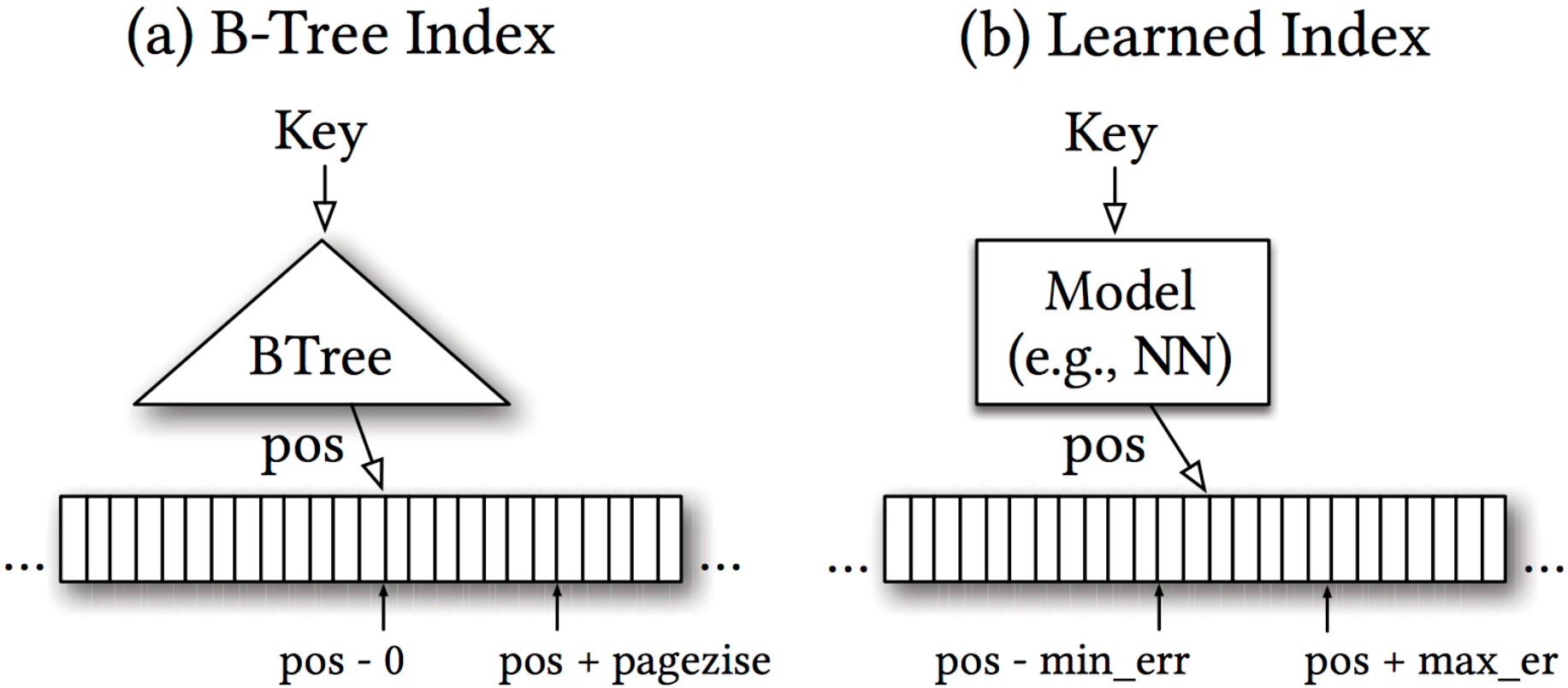
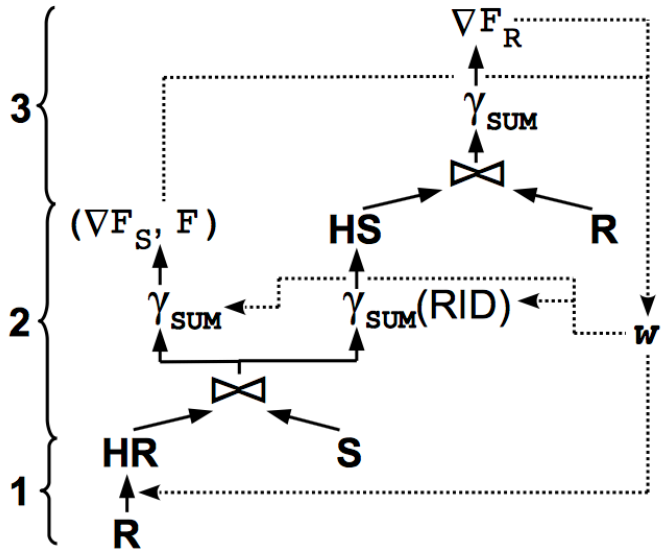


Figure 1: Why B-Trees are models

Serving Feature Selection Workloads



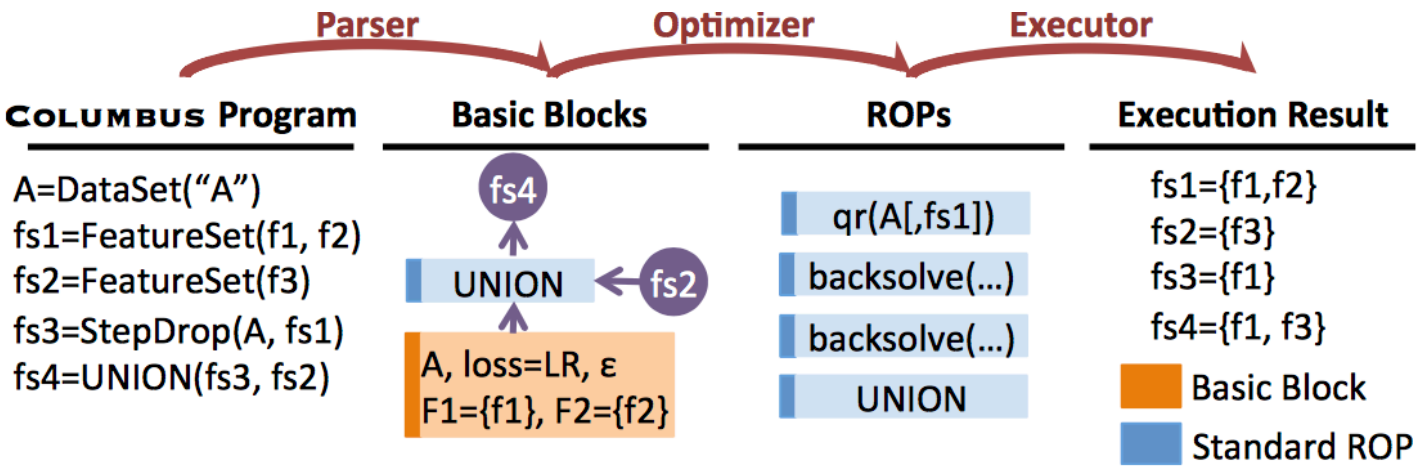
Logical Schemas:

R(RID, $\mathbf{X}_R$)

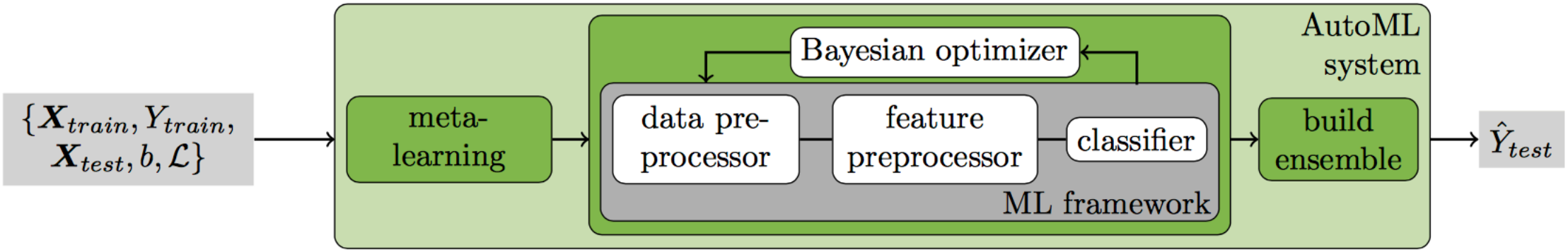
S(SID, Y, $\mathbf{X}_S$, FK)

HR(RID, PartialIP)

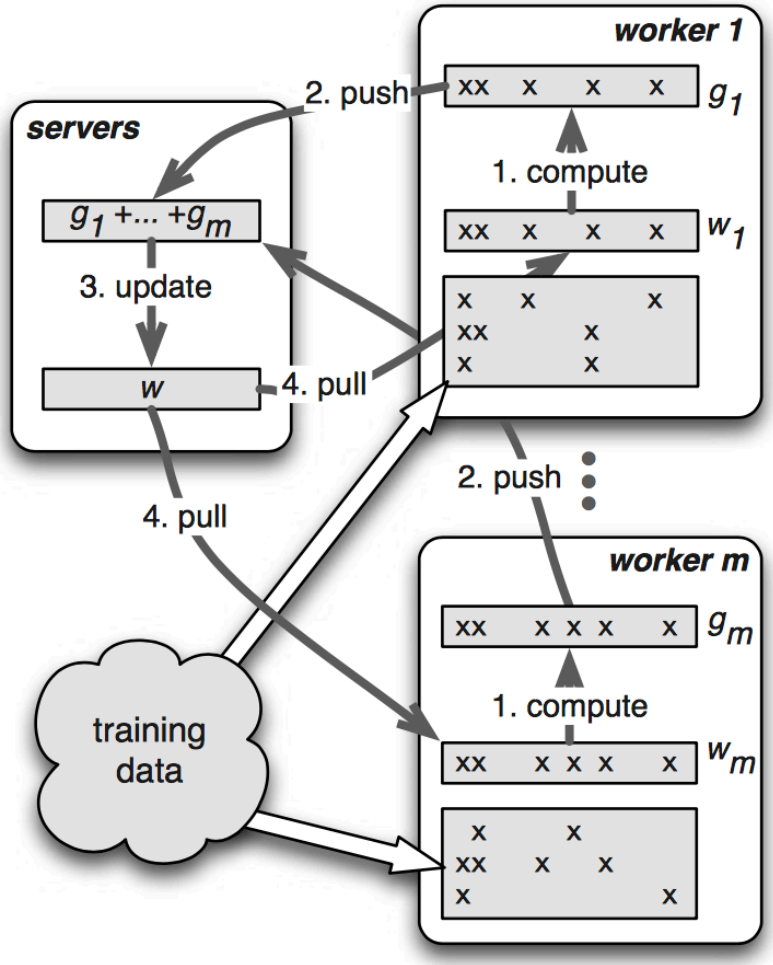
HS(RID, SumScaledIP)

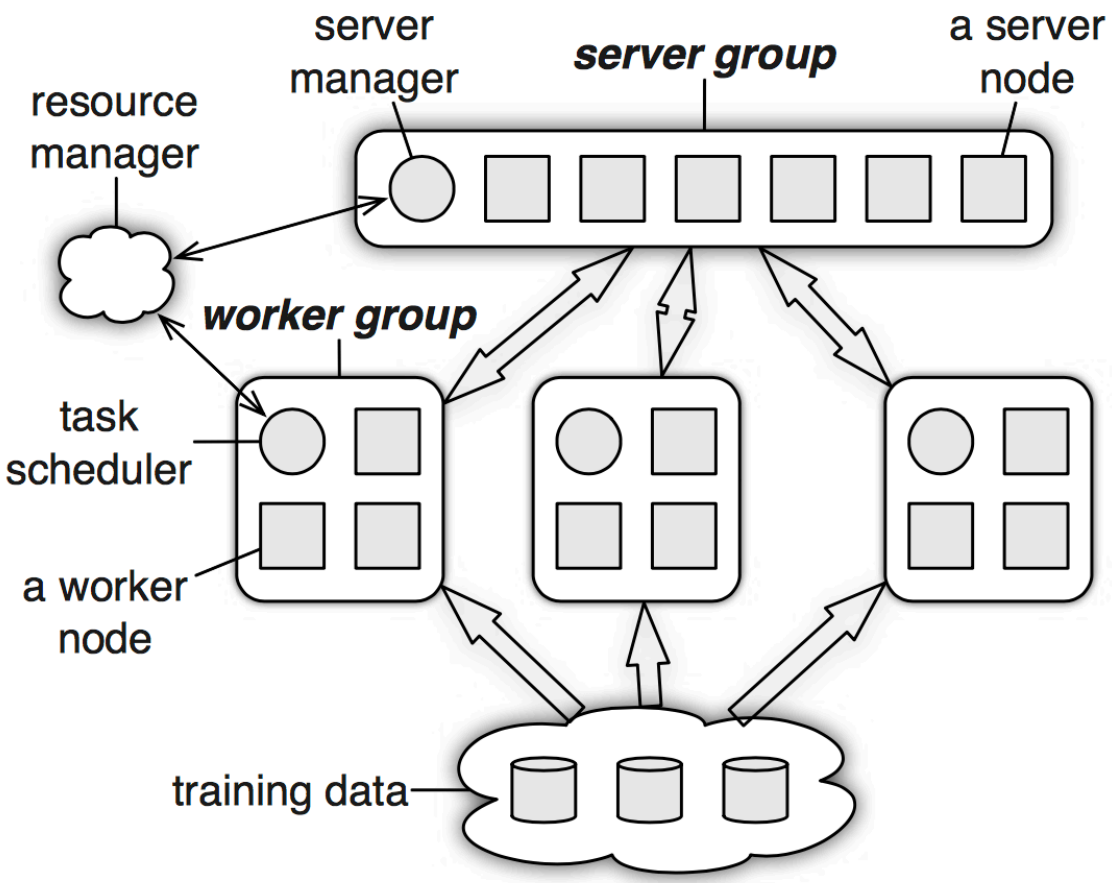


AutoML

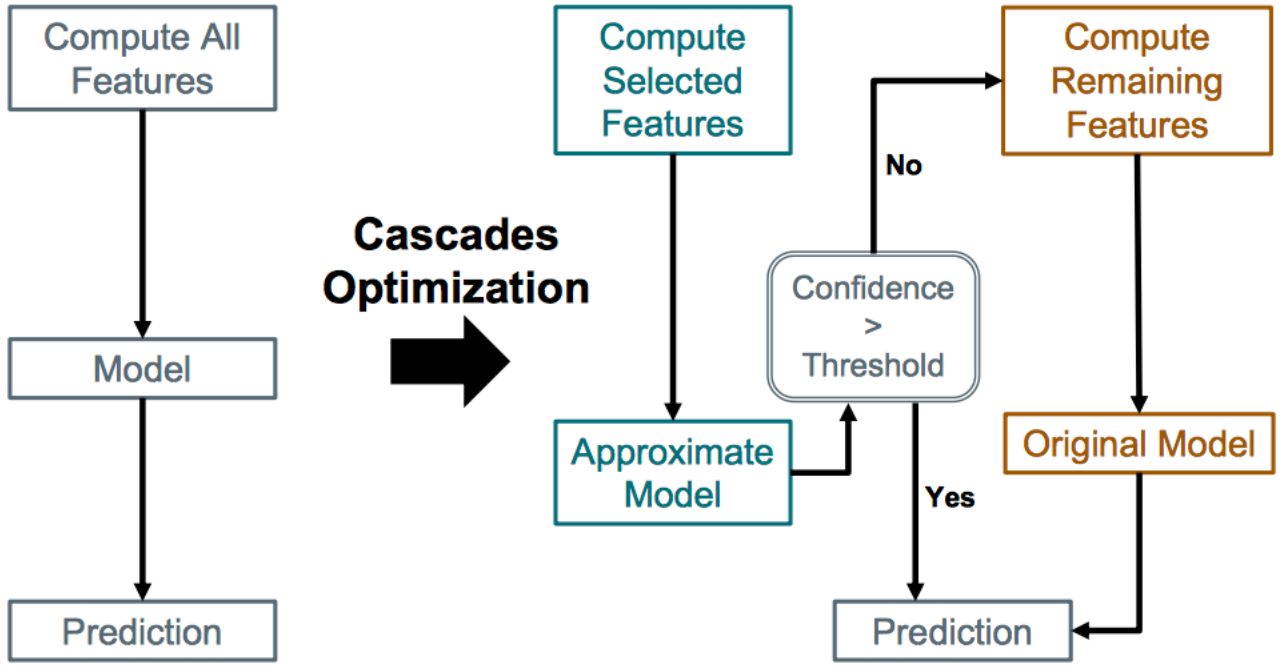

$\{\boldsymbol{X}_{train}, Y_{train}, \boldsymbol{X}_{test}, b, \mathcal{L}\}$
meta-learning
Bayesian optimizer
data pre-processor
feature preprocessor
classifier
ML framework
AutoML system
build ensemble
$\hat{Y}_{test}$

Distributed Model Training





Efficient Prediction Serving



Model Compression

SUMMARIZATION OF DIFFERENT APPROACHES FOR MODEL COMPRESSION AND ACCELERATION.

Theme Name	Description	Applications	More details
Parameter pruning and sharing	Reducing redundant parameters which are not sensitive to the performance	Convolutional layer and fully connected layer	Robust to various settings, can achieve good performance, can support both train from scratch and pre-trained model
Low-rank factorization	Using matrix/tensor decomposition to estimate the informative parameters	Convolutional layer and fully connected layer	Standardized pipeline, easily to be implemented, can support both train from scratch and pre-trained model
Transferred/compact convolutional filters	Designing special structural convolutional filters to save parameters	Convolutional layer only	Algorithms are dependent on applications, usually achieve good performance, only support train from scratch
Knowledge distillation	Training a compact neural network with distilled knowledge of a large model	Convolutional layer and fully connected layer	Model performances are sensitive to applications and network structure only support train from scratch

Relational Inductive Bias in Neural Networks