# Operationalizing Machine Learning

## An Interview Study

Shreya Shankar*, Rolando Garcia*, Joseph M. Hellerstein, Aditya G. Parameswaran

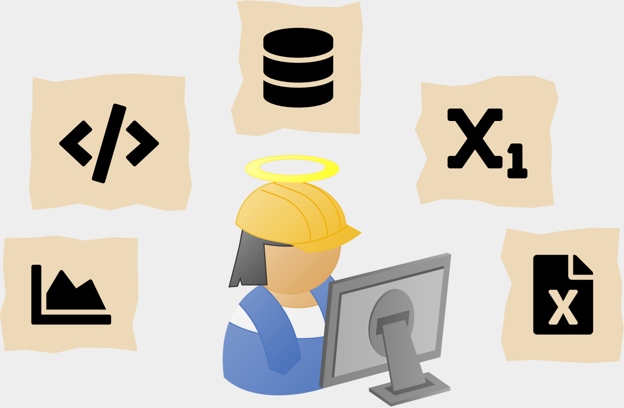
October 2022

University of California, Berkeley
Co-first Authors*

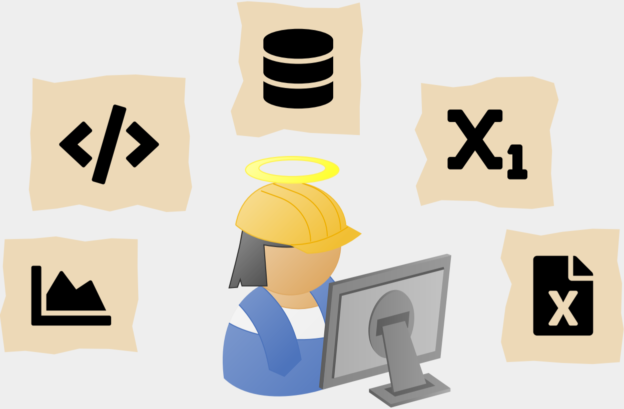
# Outline

# Modern Intelligent Applications

# A Dark Art

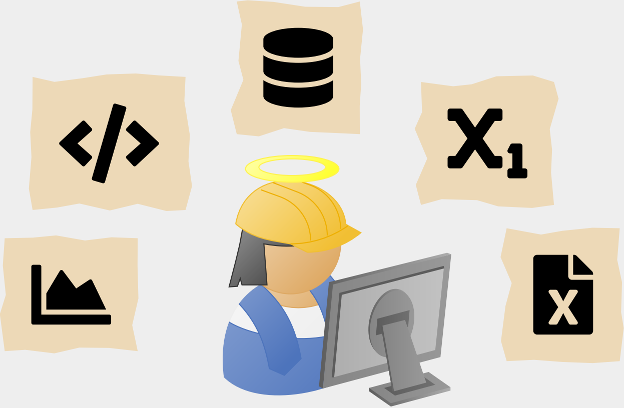


# A Dark Art




How to do this with much less ML and CS training?

# A Dark Art



How to do this with much less ML and CS training?

**Goal: build relevant and useful ML tools**

**Data Collection**

Sourcing, labeling, cleaning, wrangling

# Textbook ML

**Data Collection**

Sourcing, labeling, cleaning, wrangling

↓

**Model Training**

↓

Shankar and Garcia et al.

# Textbook ML

**Data Collection**

Sourcing, labeling, cleaning, wrangling

↓

**Model Training**

↓

**Model Evaluation**

Hold-out dataset, good accuracy, no overfitting

↓

# Textbook ML

**Data Collection**

Sourcing, labeling, cleaning, wrangling

↓

**Model Training**

↓

**Model Evaluation**

Hold-out dataset, good accuracy, no overfitting

↓

**Deploy!**

Shankar and Garcia et al.

# Textbook ML

**Data Collection**

Sourcing, labeling, cleaning, wrangling

↓

**Model Training**

↓

**Model Evaluation**

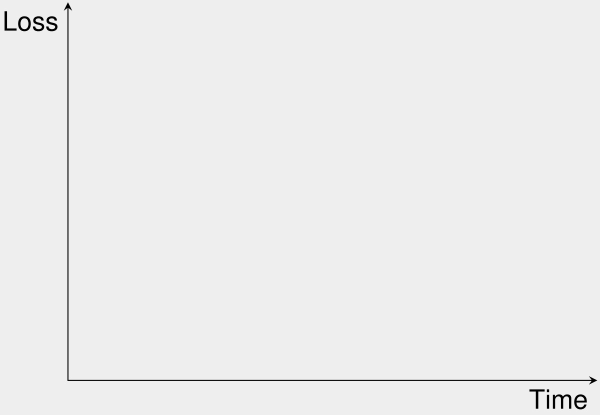
Hold-out dataset, good accuracy, no overfitting

↓

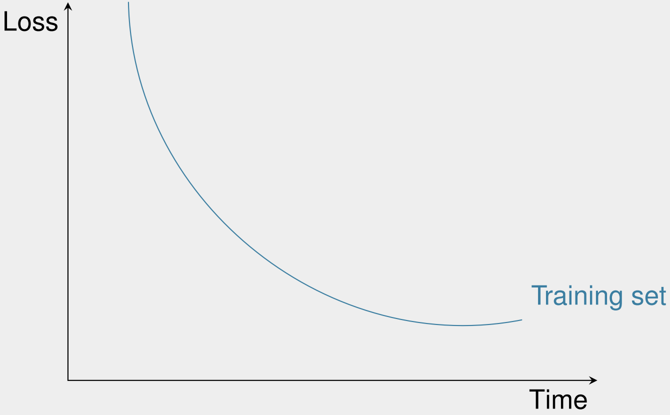
**Deploy!**

Shankar and Garcia et al.

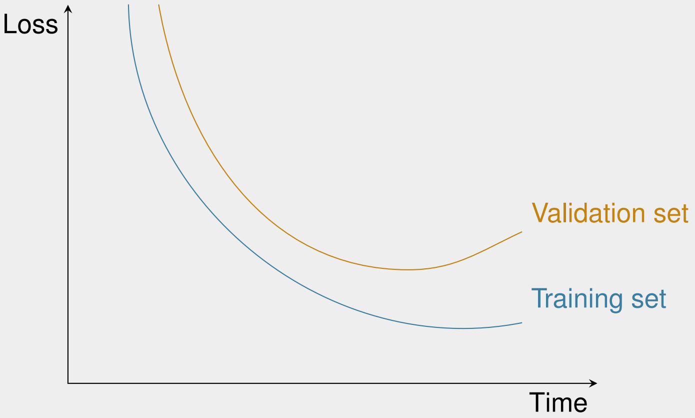
# Textbook Deployment



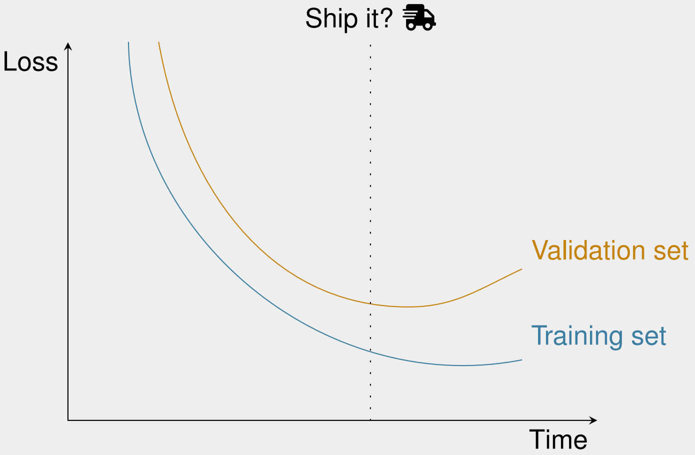
# Textbook Deployment



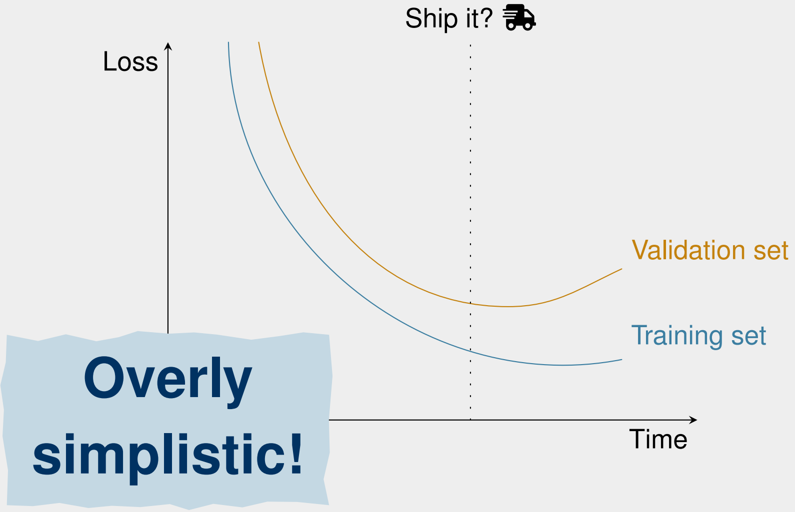
# Textbook Deployment



# Textbook Deployment



# Textbook Deployment

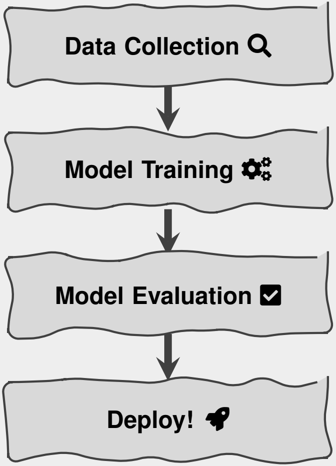



Shankar and Garcia et al.

# A New Wave of Software Engineering

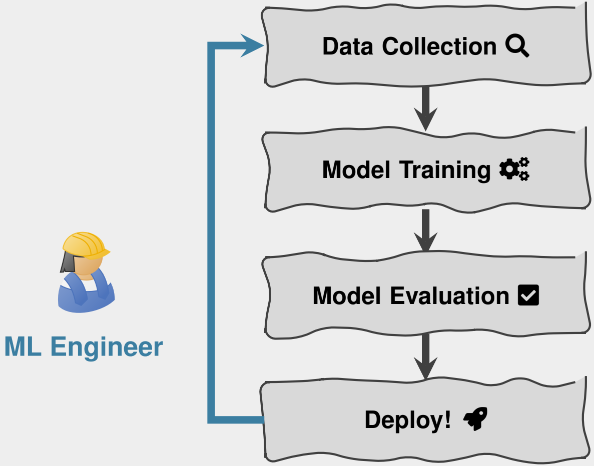
## ML is Hard to Operationalize

# A Typical MLOps Scenario



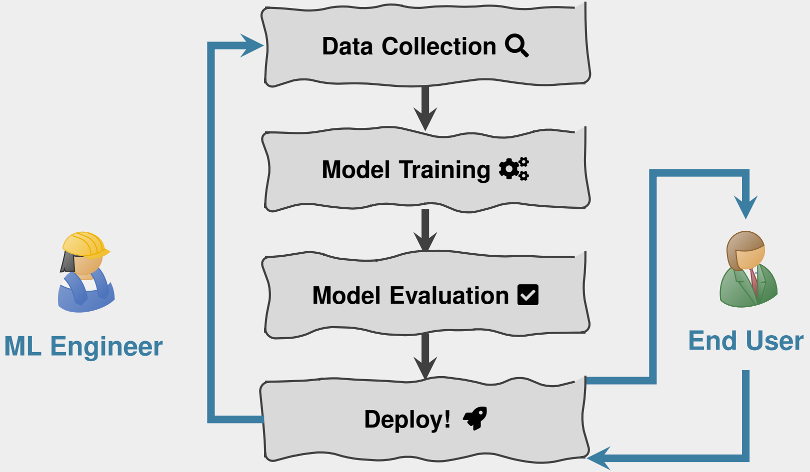

Shankar and Garcia et al.

# A Typical MLOps Scenario



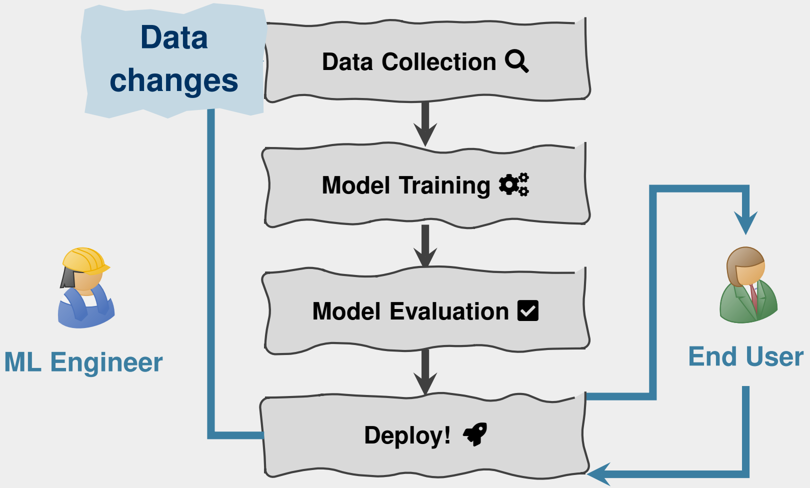

Shankar and Garcia et al.

# A Typical MLOps Scenario



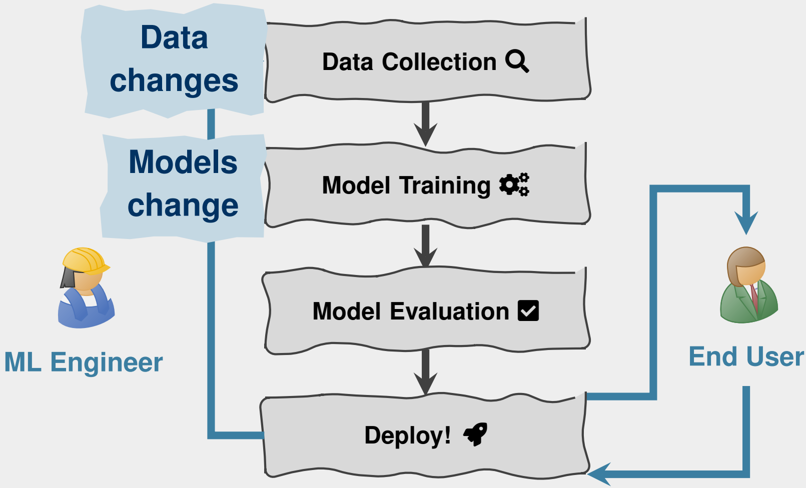

Shankar and Garcia et al.

# A Typical MLOps Scenario



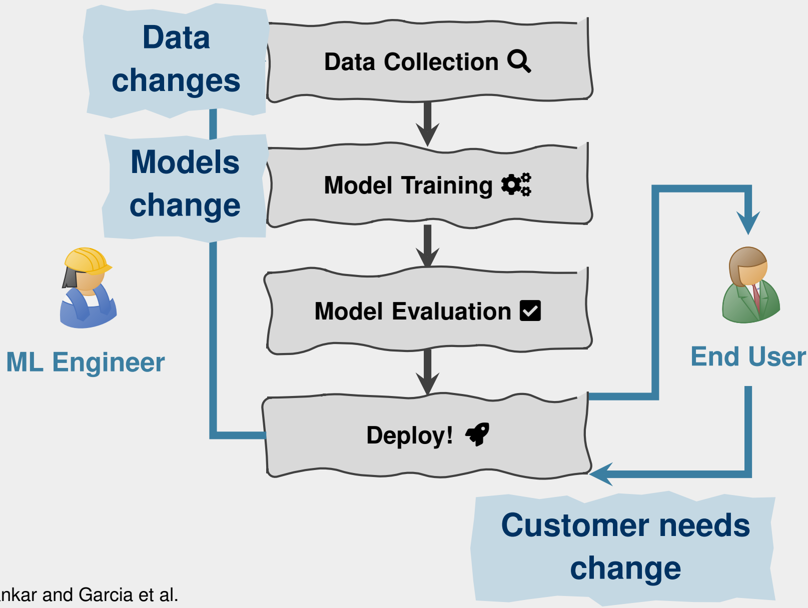

Shankar and Garcia et al.

# A Typical MLOps Scenario



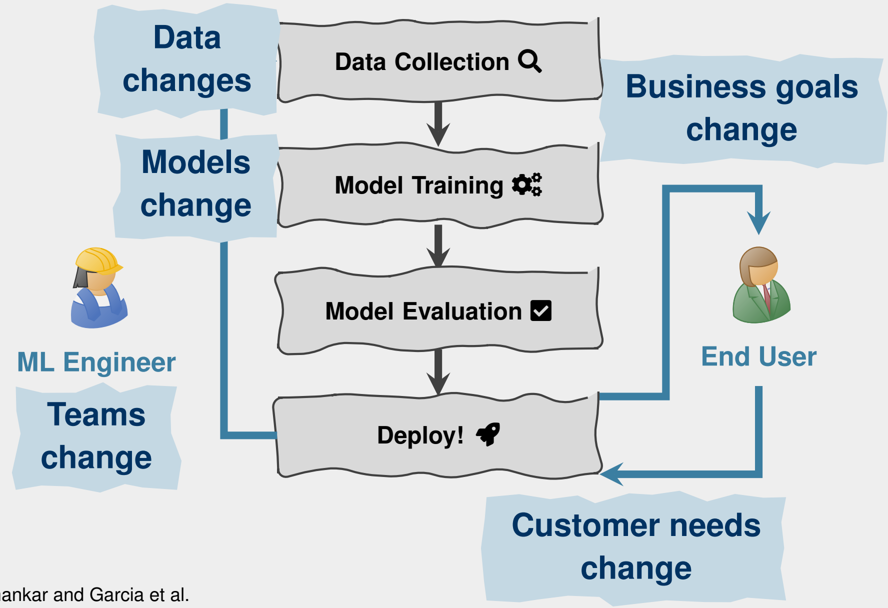

Shankar and Garcia et al.

# A Typical MLOps Scenario



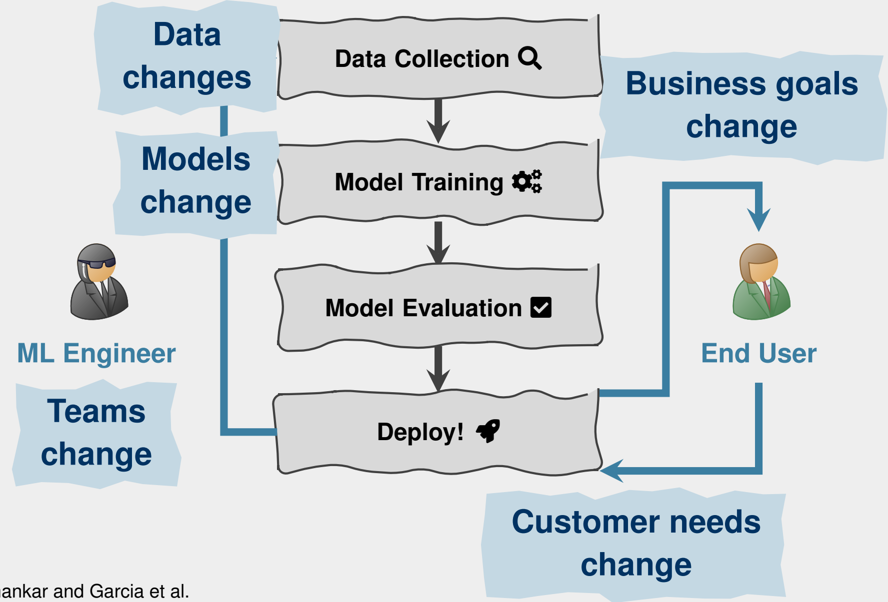

Shankar and Garcia et al.

# A Typical MLOps Scenario



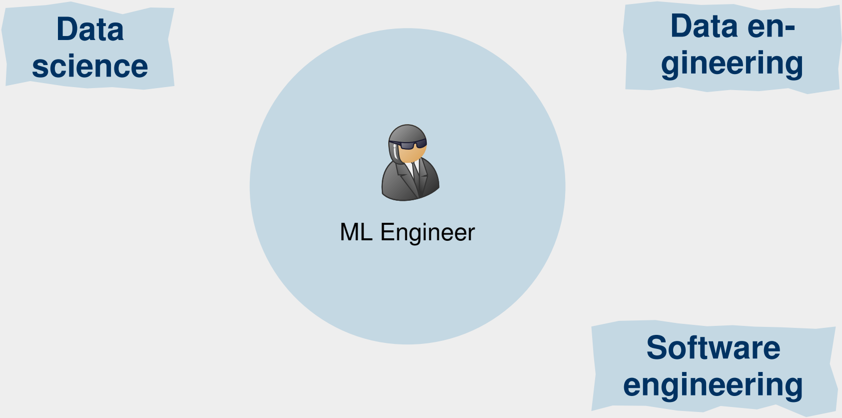

Shankar and Garcia et al.

# A Typical MLOps Scenario



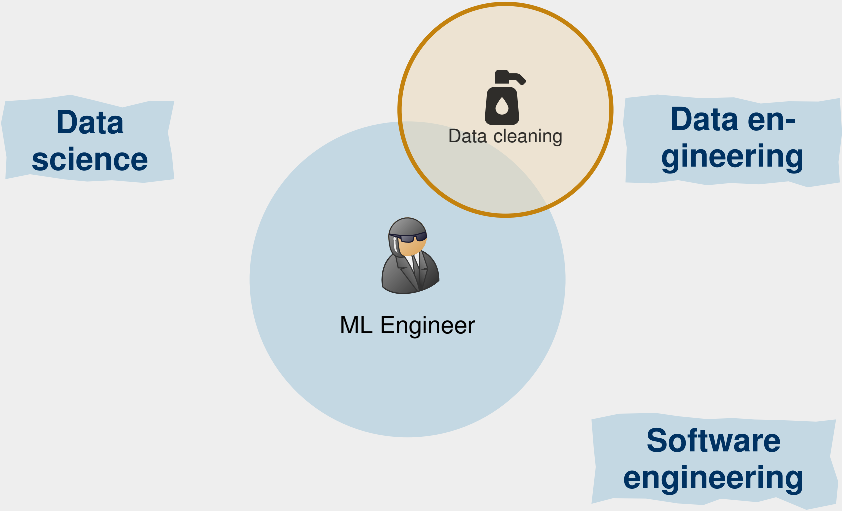

Shankar and Garcia et al.

# ML Engineers: Doing it All 🏃




Shankar and Garcia et al.

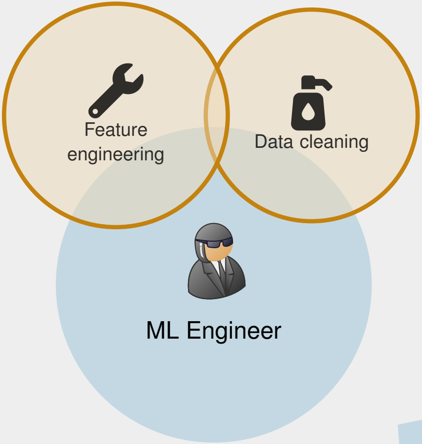
# ML Engineers: Doing it All 🏃




Shankar and Garcia et al.

# ML Engineers: Doing it All

**Data science**

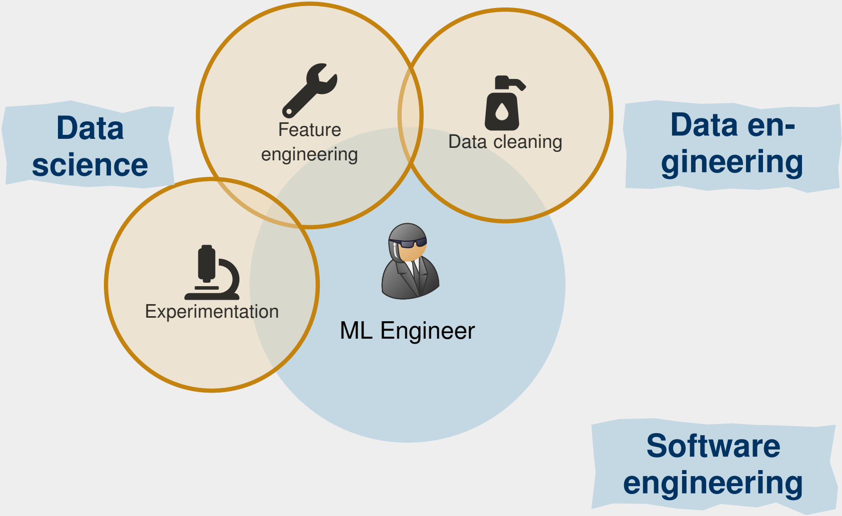



**Data engineering**

**Software engineering**

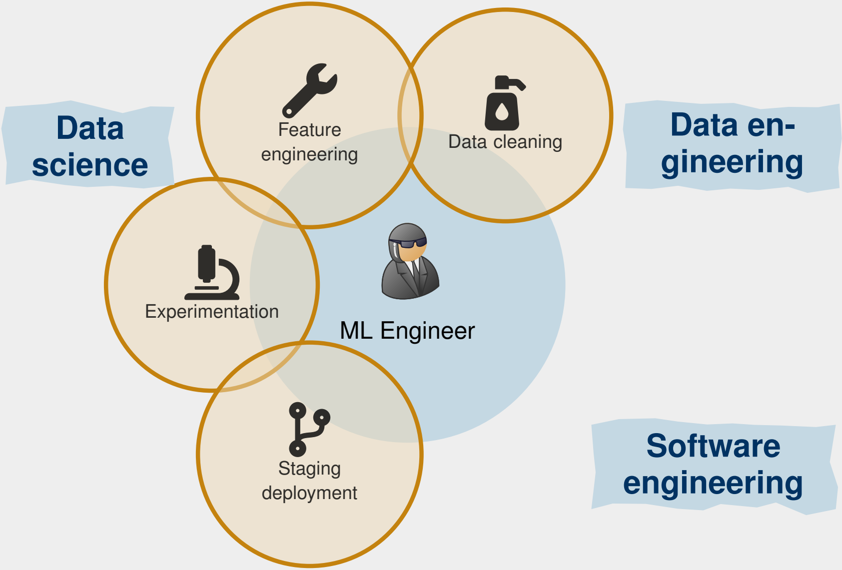
Shankar and Garcia et al.

# ML Engineers: Doing it All



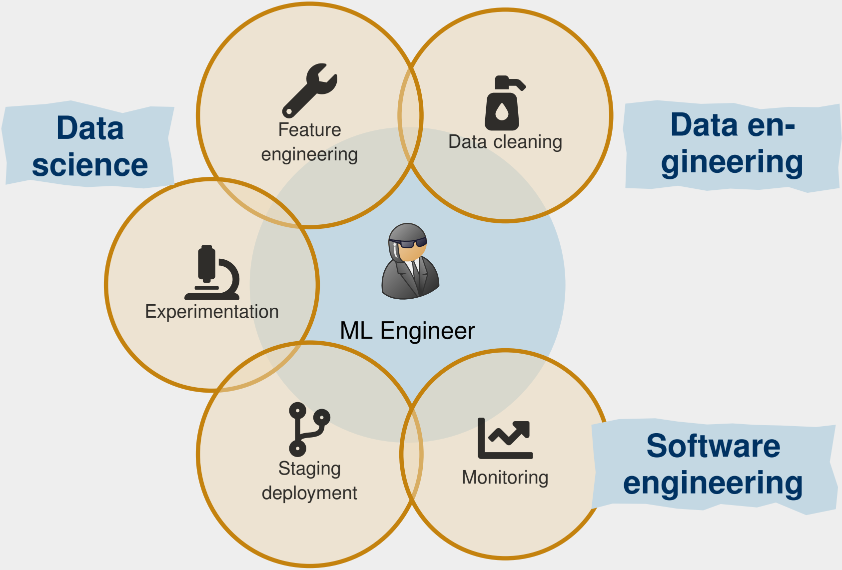

Shankar and Garcia et al.

# ML Engineers: Doing it All



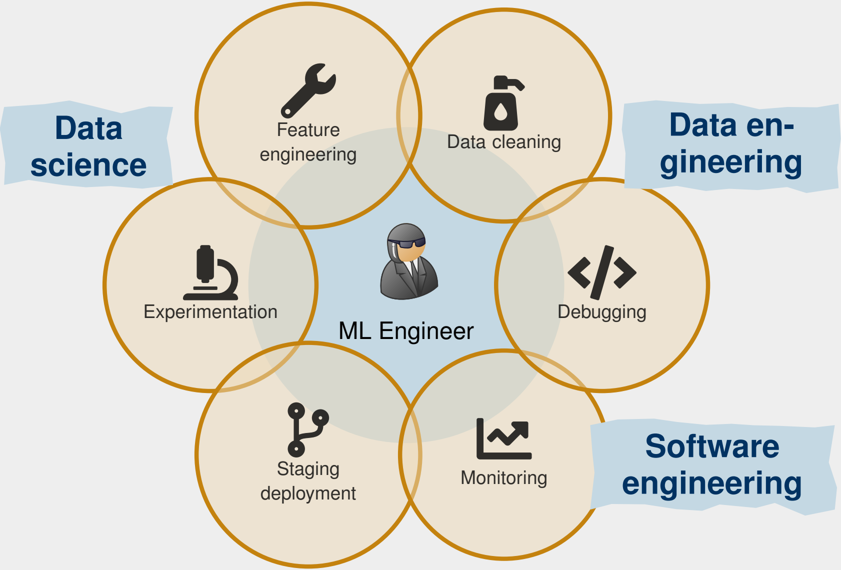

Shankar and Garcia et al.

# ML Engineers: Doing it All




Shankar and Garcia et al.

# ML Engineers: Doing it All




Shankar and Garcia et al.

# How do people put and keep ML models in production?

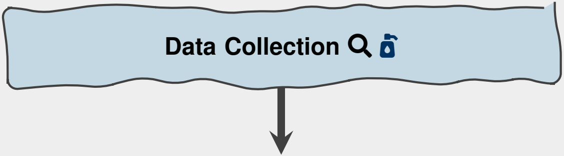
# Interview Setup

| | Role | Company Size | Sector |
|---|---|---|---|
| p1 | MLE Manager | Large | Autonomous vehicles |
| p2 | MLE | Medium | Autonomous vehicles |
| p3 | MLE | Small | Computer hardware |
| p4 | MLE | Medium | Retail |
| p5 | MLE Manager | Large | Ads |
| p6 | MLE | Large | Cloud computing |
| p7 | MLE | Small | Finance |
| p8 | MLE | Small | Bioinformatics |
| p10 | MLE | Small | Banking |
| p11 | MLE Manager | Medium | Banking |
| p12 | MLE | Large | Cloud computing |
| p13 | MLE | Small | Bioinformatics |
| p14 | MLE | Medium | Cybersecurity |
| p15 | MLE | Medium | Fintech |
| p16 | MLE | Small | Marketing and analytics |
| p17 | MLE | Medium | Website builder |
| p18 | MLE | Large | Recommender systems |
| p19 | MLE Manager | Large | Ads |

Small: $<$ 100, medium-sized: 100-1000, and large: $>$ 1000 employees.

# Outline

# Revisiting the ML Lifecycle



# Revisiting the ML Lifecycle

**Data Collection**

↓

~~**Model Training**~~ **Experimentation**

↓

# Revisiting the ML Lifecycle

**Data Collection**

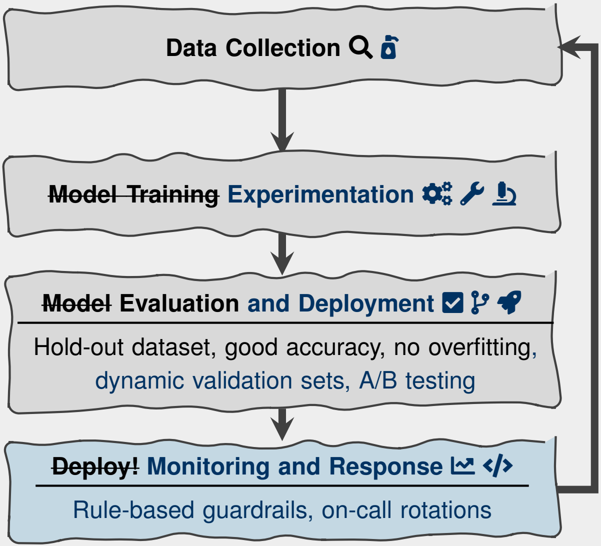
**~~Model Training~~ Experimentation**

**~~Model~~ Evaluation and Deployment**

Hold-out dataset, good accuracy, no overfitting, dynamic validation sets, A/B testing

Shankar and Garcia et al.

# Revisiting the ML Lifecycle




Shankar and Garcia et al.

# Outline

# MLOps Practices

## Operationalizing evaluation requires active efforts

# Dynamic Validation Datasets

Every [failed prediction] gets into the same queue, and 3 of us sit down once a week and go through the queue...then our [analysts] collect more [similar] data. (P8)

# Dynamic Validation Datasets

**Every [failed prediction]** gets into the same queue, and 3 of us sit down once a week and go through the queue...then our [analysts] **collect more [similar] data**. (P8)

# Multi-Staged Evaluation

We spent a long time very slowly, ramping up the model to very small percentages of traffic and watching what happened. [When there was a failure mode,] a product person would ping us and say: hey, this was kind of weird, should we create a rule around this [suggested text] to filter this out? (P15)

# Multi-Staged Evaluation

We spent a long time very slowly, **ramping up the model to very small percentages of traffic** and watching what happened. [When there was a failure mode,] a product person would ping us and say: hey, this was kind of weird, **should we create a rule** around this [suggested text] to filter this out? (P15)

# Balancing ML and Product Metrics ⚖

The first task is to figure out, what are customers actually interested in, or what's the metric that they care about. (P16)

# Balancing ML and Product Metrics ⚖

The first task is to figure out, what are **customers** actually interested in, or what's the **metric** that they care about. (P16)

# Balancing ML and Product Metrics ⚖

The first task is to figure out, what are **customers** actually interested in, or what's the **metric** that they care about. (P16)

If we can get a statistically significant greater percentage [of] people to subscribe to [the product], then [we can fully deploy]. (P17)

# Balancing ML and Product Metrics ⚖

The first task is to figure out, what are **customers** actually interested in, or what's the **metric** that they care about. (P16)

If we can get a statistically significant **greater percentage [of] people to subscribe** to [the product], then [we can fully deploy]. (P17)

# Keeping up with Change ⇄

**Myths**

- ✗ Static hold-out set
- ✗ Global metric suffices
- ✗ ML metrics only

# Keeping up with Change ⇄

**Myths**

- ✗ Static hold-out set
- ✗ Global metric suffices
- ✗ ML metrics only

**Tips**

- ✓ Dynamic datasets
- ✓ Check underperforming populations
- ✓ ML *and* product metrics

Shankar and Garcia et al.

# Revisiting the ML Lifecycle

**Data Collection**

**~~Model Training~~ Experimentation**

**~~Model~~ Evaluation and Deployment**

Hold-out dataset, good accuracy, no overfitting, dynamic validation sets, A/B testing

**~~Deploy!~~ Monitoring and Response**

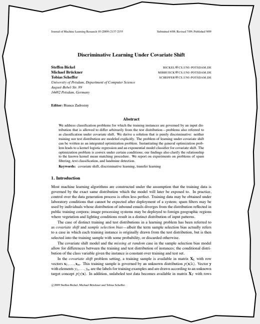
Rule-based guardrails, on-call rotations

Shankar and Garcia et al.

# MLOps Practices

## Non-ML rules and human-in-the-loop practices keep models reliable in production

# Data Distribution Shift


Journal of Machine Learning Research 10 (2009) 2137-2155 Submitted 4/08; Revised 7/09; Published 9/09

**Discriminative Learning Under Covariate Shift**

**Steffen Bickel** BICKEL@CS.UNI-POTSDAM.DE
**Michael Brückner** MIBRUECK@CS.UNI-POTSDAM.DE
**Tobias Scheffer** SCHEFFER@CS.UNI-POTSDAM.DE
*University of Potsdam, Department of Computer Science*
*August-Bebel-Str. 89*
*14482 Potsdam, Germany*

**Editor:** Bianca Zadrozny

**Abstract**

We address classification problems for which the training instances are governed by an input distribution that is allowed to differ arbitrarily from the test distribution—problems also referred to as classification under covariate shift. We derive a solution that is purely discriminative: neither training nor test distribution are modeled explicitly. The problem of learning under covariate shift can be written as an integrated optimization problem. Instantiating the general optimization problem leads to a kernel logistic regression and an exponential model classifier for covariate shift. The optimization problem is convex under certain conditions; our findings also clarify the relationship to the known kernel mean matching procedure. We report on experiments on problems of spam filtering, text classification, and landmine detection.

**Keywords:** covariate shift, discriminative learning, transfer learning

**1. Introduction**

Most machine learning algorithms are constructed under the assumption that the training data is governed by the exact same distribution which the model will later be exposed to. In practice, control over the data generation process is often less perfect. Training data may be obtained under laboratory conditions that cannot be expected after deployment of a system; spam filters may be used by individuals whose distribution of inbound emails diverges from the distribution reflected in public training corpora; image processing systems may be deployed to foreign geographic regions where vegetation and lighting conditions result in a distinct distribution of input patterns.

The case of distinct training and test distributions in a learning problem has been referred to as *covariate shift* and *sample selection bias*—albeit the term sample selection bias actually refers to a case in which each training instance is originally drawn from the test distribution, but is then selected into the training sample with some probability, or discarded otherwise.

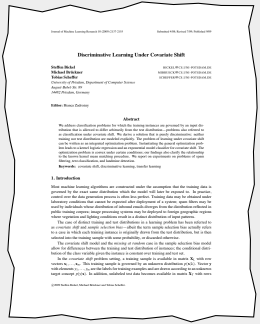
The covariate shift model and the *missing at random* case in the sample selection bias model allow for differences between the training and test distribution of instances; the conditional distribution of the class variable given the instance is constant over training and test set.

In the *covariate shift* problem setting, a training sample is available in matrix $\mathbf{X}_L$ with row vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$. This training sample is governed by an unknown distribution $p(\mathbf{x}|\lambda)$. Vector $\mathbf{y}$ with elements $y_1, \ldots, y_m$ are the labels for training examples and are drawn according to an unknown target concept $p(y|\mathbf{x})$. In addition, unlabeled test data becomes available in matrix $\mathbf{X}_T$ with rows

©2009 Steffen Bickel, Michael Brückner and Tobias Scheffer.

Shankar and Garcia et al.

# Data Distribution Shift



Journal of Machine Learning Research 10 (2009) 2137-2155 Submitted 4/08; Revised 7/09; Published 9/09

**Discriminative Learning Under Covariate Shift**

**Steffen Bickel** BICKEL@CS.UNI-POTSDAM.DE
**Michael Brückner** MIBRUECK@CS.UNI-POTSDAM.DE
**Tobias Scheffer** SCHEFFER@CS.UNI-POTSDAM.DE
*University of Potsdam, Department of Computer Science*
*August-Bebel-Str. 89*
*14482 Potsdam, Germany*

**Editor:** Bianca Zadrozny

**Abstract**

We address classification problems for which the training instances are governed by an input distribution that is allowed to differ arbitrarily from the test distribution—problems also referred to as classification under covariate shift. We derive a solution that is purely discriminative: neither training nor test distribution are modeled explicitly. The problem of learning under covariate shift can be written as an integrated optimization problem. Instantiating the general optimization problem leads to a kernel logistic regression and an exponential model classifier for covariate shift. The optimization problem is convex under certain conditions; our findings also clarify the relationship to the known kernel mean matching procedure. We report on experiments on problems of spam filtering, text classification, and landmine detection.

**Keywords:** covariate shift, discriminative learning, transfer learning

**1. Introduction**

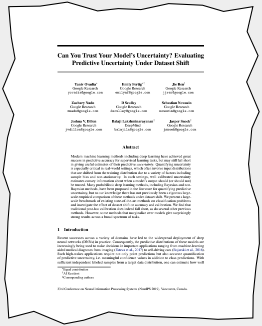
Most machine learning algorithms are constructed under the assumption that the training data is governed by the exact same distribution which the model will later be exposed to. In practice, control over the data generation process is often less perfect. Training data may be obtained under laboratory conditions that cannot be expected after deployment of a system; spam filters may be used by individuals whose distribution of inbound emails diverges from the distribution reflected in public training corpora; image processing systems may be deployed to foreign geographic regions where vegetation and lighting conditions result in a distinct distribution of input patterns.

The case of distinct training and test distributions in a learning problem has been referred to as *covariate shift* and *sample selection bias*—albeit the term sample selection bias actually refers to a case in which each training instance is originally drawn from the test distribution, but is then selected into the training sample with some probability, or discarded otherwise.

The covariate shift model and the *missing at random* case in the sample selection bias model allow for differences between the training and test distribution of instances; the conditional distribution of the class variable given the instance is constant over training and test set.

In the *covariate shift* problem setting, a training sample is available in matrix $\mathbf{X}_L$ with row vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$. This training sample is governed by an unknown distribution $p(\mathbf{x}|\lambda)$. Vector $\mathbf{y}$ with elements $y_1, \ldots, y_m$ are the labels for training examples and are drawn according to an unknown target concept $p(y|\mathbf{x})$. In addition, unlabeled test data becomes available in matrix $\mathbf{X}_T$ with rows





**Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift**

**Yaniv Ovadia**[*] Google Research yovadia@google.com
**Emily Fertig**[*†] Google Research emilyaf@google.com
**Jie Ren**[†] Google Research jjren@google.com
**Zachary Nado** Google Research znado@google.com
**D Sculley** Google Research dsculley@google.com
**Sebastian Nowozin** Google Research nowozin@google.com
**Joshua V. Dillon** Google Research jvdillon@google.com
**Balaji Lakshminarayanan**[‡] DeepMind balajiln@google.com
**Jasper Snoek**[‡] Google Research jsnoek@google.com

**Abstract**

Modern machine learning methods including deep learning have achieved great success in predictive accuracy for supervised learning tasks, but may still fall short in giving useful estimates of their predictive *uncertainty*. Quantifying uncertainty is especially critical in real-world settings, which often involve input distributions that are shifted from the training distribution due to a variety of factors including sample bias and non-stationarity. In such settings, well calibrated uncertainty estimates convey information about when a model's output should (or should not) be trusted. Many probabilistic deep learning methods, including Bayesian-and non-Bayesian methods, have been proposed in the literature for quantifying predictive uncertainty, but to our knowledge there has not previously been a rigorous large-scale empirical comparison of these methods under dataset shift. We present a large-scale benchmark of existing state-of-the-art methods on classification problems and investigate the effect of dataset shift on accuracy and calibration. We find that traditional post-hoc calibration does indeed fall short, as do several other previous methods. However, some methods that marginalize over models give surprisingly strong results across a broad spectrum of tasks.

**1 Introduction**

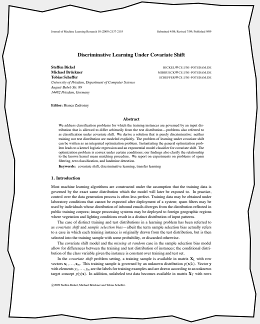
Recent successes across a variety of domains have led to the widespread deployment of deep neural networks (DNNs) in practice. Consequently, the predictive distributions of these models are increasingly being used to make decisions in important applications ranging from machine-learning aided medical diagnoses from imaging (Esteva et al., 2017) to self-driving cars (Bojarski et al., 2016). Such high-stakes applications require not only point predictions but also accurate quantification of predictive uncertainty, i.e. meaningful confidence values in addition to class predictions. With sufficient independent labeled samples from a target data distribution, one can estimate how well

[*]Equal contribution
[†]AI Resident
[‡]Corresponding authors

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Shankar and Garcia et al.

# Data Distribution Shift


Journal of Machine Learning Research 10 (2009) 2137-2155 Submitted 4/08; Revised 7/09; Published 9/09

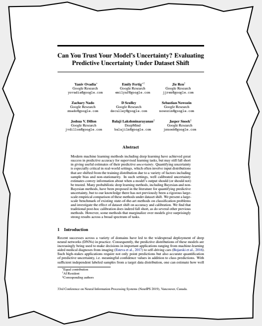
**Discriminative Learning Under Covariate Shift**

Steffen Bickel BICKEL@CS.UNI-POTSDAM.DE
Michael Brückner MIBRUECK@CS.UNI-POTSDAM.DE
Tobias Scheffer SCHEFFER@CS.UNI-POTSDAM.DE
*University of Potsdam, Department of Computer Science*
*August-Bebel-Str. 89*
*14482 Potsdam, Germany*

**Editor:** Bianca Zadrozny

**Abstract**

**1. Introduction**


**Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift**

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, Jasper Snoek

**Abstract**

**1 Introduction**

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.


Pattern Recognition 45 (2012) 521–530

Pattern Recognition

A unifying view on dataset shift in classification

Jose G. Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, Francisco Herrera

ARTICLE INFO

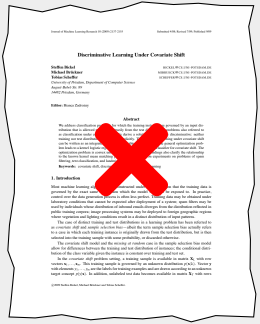
ABSTRACT

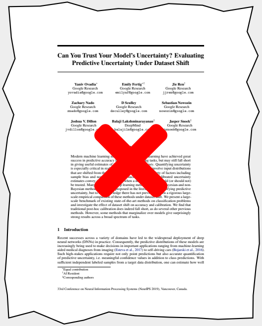
1. Introduction

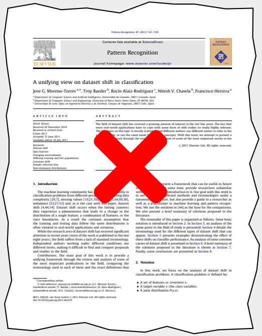
2. Notation

Shankar and Garcia et al.

# Data Distribution Shift




Shankar and Garcia et al.

# Frequently Retrain

Why did we start training daily? As far as I'm aware, we wanted to start simple—we could just have a single batch job that processes new data and we wouldn't need to worry about separate retraining schedules. You don't really need to worry about if your model has gone stale if you're retraining it every day. (P14)

# Frequently Retrain

Why did we start training daily? As far as I'm aware, **we wanted to start simple**—we could just have a single batch job that processes new data and we wouldn't need to worry about separate retraining schedules. **You don't really need to worry about if your model has gone stale if you're retraining it every day.** (P14)

# Maintain Old and Simple Fallback Versions

If the production model drops and the calibration model is still performing within a [specified] range, we'll fall back to the calibration model until someone will fix the production model. (P19)

# Maintain Old and Simple Fallback Versions

If the production model drops and the calibration model is still performing within a [specified] range, we'll **fall back to the calibration model** until someone will fix the production model. (P19)

# Maintain Old and Simple Fallback Versions

If the production model drops and the calibration model is still performing within a [specified] range, we'll **fall back to the calibration model** until someone will fix the production model. (P19)

It's important to keep some model up and running, even if we switch to a less economic model and have to just cut the losses. (P18)

# Maintain Old and Simple Fallback Versions

If the production model drops and the calibration model is still performing within a [specified] range, we'll **fall back to the calibration model** until someone will fix the production model. (P19)

It's important to keep some model up and running, **even if we switch to a less economic model** and have to just cut the losses. (P18)

# Validate Inputs and Outputs 🔔

Set alerts when data is corrupted, e.g.,

- Hard constraints/bounds for features (P2)
- “Common-sense checks” like nonnegativity (P16)
- Schema/type checks (P8)

**Example Feature: User Age**

✔ $< 125$
✔ Nonnegative
✔ Integer

# Sustaining Model Performance

**Myths**

- ✗ Build models “robust” to data distribution shift
- ✗ Always serve the latest model’s predictions
- ✗ Validate only the outputs
- ✗ Set and forget

Shankar and Garcia et al.

# Sustaining Model Performance

**Myths**

- ✗ Build models “robust” to data distribution shift
- ✗ Always serve the latest model’s predictions
- ✗ Validate only the outputs
- ✗ Set and forget

**Tips**

- ✓ Frequently retrain models on latest data
- ✓ Maintain simple models and heuristics for rollback
- ✓ Validate inputs and outputs
- ✓ On-call rotations and SLAs

Shankar and Garcia et al.

# Outline

# Dev-Prod Environment Mismatch 🧦

- Data leakage
- Jupyter notebook usage
- Code reviews

# Dev-Prod Environment Mismatch

I used to see a lot of people complaining that model developers don't follow software engineering [practices]. At this point, I'm feeling more convinced that they don't follow software engineering [practices]—[not] because they're lazy, [but because software engineering practices are] contradictory to the agility of analysis and exploration. (P6)

# Dev-Prod Environment Mismatch

> I used to see a lot of people complaining that model developers don't follow software engineering [practices]. At this point, I'm feeling more convinced that they don't follow software engineering [practices]—[not] because they're lazy, [but because **software engineering practices are] contradictory to the agility of analysis and exploration.** (P6)

# Data Validation: Alert Fatigue 🔔

- Many alerts triggered even when ML performance is fine
- On-call rotations are scary!

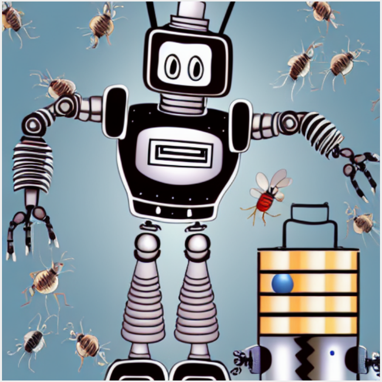
# Data Validation: Alert Fatigue

You typically ignore most alerts...I guess on record I'd say 90% of them aren't immediate. You just have to acknowledge them [internally], like just be aware that there is something happening. (P18)

# Data Validation: Alert Fatigue

**You typically ignore most alerts**...I guess on record I'd say 90% of them aren't immediate. You just have to acknowledge them [internally], like just be aware that there is something happening. (P18)

# Taming the Long Tail of ML Bugs



- “I just sort of poked around until, at some point, I figured [it] out.” (P16)
- Slicing and dicing data
- Paranoia induced by debugging trauma

Shankar and Garcia et al.

# Taming the Long Tail of ML Bugs

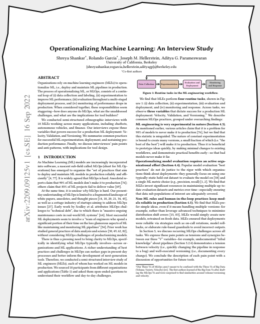
ML [bugs] don't get caught by tests or production systems and just silently cause errors [that manifest as] slight reductions in performance. This is why [you] need to be paranoid when you're writing ML code. (P1)

# Taming the Long Tail of ML Bugs

ML [bugs] don't get caught by tests or production systems and just **silently cause errors** [that manifest as] slight reductions in performance. This is why **[you] need to be paranoid when you're writing ML code.** (P1)

# Outline

# MLOps Research Roadmap

arXiv:2209.09125v1 [cs.SE] 16 Sep 2022

**Operationalizing Machine Learning: An Interview Study**

Shreya Shankar*, Rolando Garcia*, Joseph M. Hellerstein, Aditya G. Parameswaran
University of California, Berkeley
{shreyashankar,rogarcia,hellerstein,adityagp}@berkeley.edu
*Co-first authors

**ABSTRACT**

Organizations rely on machine learning engineers (MLEs) to operationalize ML, i.e., deploy and maintain ML pipelines in production. The process of operationalizing ML, or MLOps, consists of a continual loop of (i) data collection and labeling, (ii) experimentation to improve ML performance, (iii) evaluation throughout a multi-staged deployment process, and (iv) monitoring of performance drops in production. When considered together, these responsibilities seem staggering—how does anyone do MLOps, what are the unaddressed challenges, and what are the implications for tool builders?

We conducted semi-structured ethnographic interviews with 18 MLEs working across many applications, including chatbots, autonomous vehicles, and finance. Our interviews expose three variables that govern success for a production ML deployment: Velocity, Validation, and Versioning. We summarize common practices for successful ML experimentation, deployment, and sustaining production performance. Finally, we discuss interviewees' pain points and anti-patterns, with implications for tool design.

**1 INTRODUCTION**

As Machine Learning (ML) models are increasingly incorporated into software, a nascent sub-field called *MLOps* (short for ML Operations) has emerged to organize the "set of practices that aim to deploy and maintain ML models in production reliably and efficiently" [4, 77]. It is widely agreed that MLOps is hard. Anecdotal reports claim that 90% of ML models don't make it to production [76]; others claim that 85% of ML projects fail to deliver value [69].

At the same time, it is unclear why MLOps is hard. Our present-day understanding of MLOps is limited to a fragmented landscape of white papers, anecdotes, and thought pieces [14, 18, 20, 21, 34, 45], as well as a cottage industry of startups aiming to address MLOps issues [27]. Early work by Sculley et al. attributes MLOps challenges to "technical debt", due to which there is "massive ongoing maintenance costs in real-world ML systems" [64]. Most successful ML deployments seem to involve a "team of engineers who spend a significant portion of their time on the less glamorous aspects of ML like maintaining and monitoring ML pipelines" [54]. Prior work has studied general practices of data analysis and science [30, 49, 62, 82], without considering MLOps challenges of productionizing models.

There is thus a pressing need to bring clarity to MLOps, specifically in identifying what MLOps typically involves—across organizations and ML applications. A richer understanding of best practices and challenges in MLOps can surface gaps in present-day processes and better inform the development of next-generation tools. Therefore, we conducted a semi-structured interview study of ML engineers (MLEs), each of whom has worked on ML models in production. We sourced 18 participants from different organizations and applications (Table 1) and asked them open-ended questions to understand their workflow and day-to-day challenges.

Data Collection → Experimentation → Evaluation and Deployment → Monitoring and Response

**Figure 1: Routine tasks in the ML engineering workflow.**

We find that MLEs perform **four routine tasks**, shown in Figure 1: (i) data collection, (ii) experimentation, (iii) evaluation and deployment, and (iv) monitoring and response. Across tasks, we observe **three variables** that dictate success for a production ML deployment: Velocity, Validation, and Versioning.[1] We describe common MLOps practices, grouped under overarching findings:

**ML engineering is very experimental in nature (Section 4.3).** As mentioned earlier, various articles claim that it is a problem for 90% of models to never make it to production [76], but we find that this statistic is misguided. The nature of constant experimentation is bound to create many versions, a small fraction of which (i.e. "the best of the best") will make it to production. Thus it is beneficial to prototype ideas quickly, by making minimal changes to existing workflows, and demonstrate practical benefits early—so that bad models never make it far.

**Operationalizing model evaluation requires an active organizational effort (Section 4.4).** Popular model evaluation "best practices" do not do justice to the rigor with which organizations think about deployments: they generally focus on using one typically-static held-out dataset to evaluate the model on [38] and a single ML metric choice (e.g., precision, recall) [1, 2]. We find that MLEs invest significant resources in maintaining multiple up-to-date evaluation datasets and metrics over time—especially ensuring that data sub-populations of interest are adequately covered.

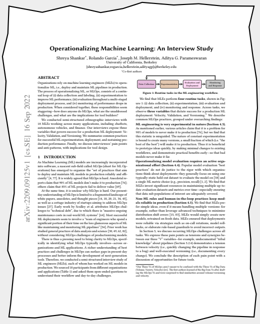
**Non-ML rules and human-in-the-loop practices keep models reliable in production (Section 4.5).** We find that MLEs prefer simple ideas, even if it means handling multiple versions: for example, rather than leverage advanced techniques to minimize distribution shift errors [15, 83], MLEs would simply create new models, retrained on fresh data. MLEs ensured that deployments were reliable via strategies such as on-call rotations, model rollbacks, or elaborate rule-based guardrails to avoid incorrect outputs.

In Section 5, we discuss recurring MLOps challenges across all tasks. We express these pain points as tensions and synergies between our three "V" variables—for example, undocumented "tribal knowledge" about pipelines (Section 5.2.4) demonstrates a tension between velocity (i.e., quickly changing the pipeline in response to a bug) and well-executed versioning (i.e., documenting every change). We conclude the description of each pain point with a discussion of opportunities for future tools.

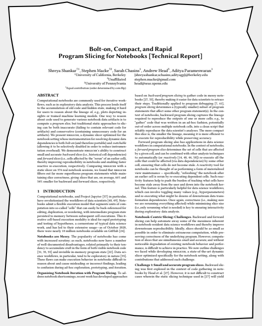
[1] Our Three Vs of MLOps aren't meant to be confused with the Three Vs of Big Data (Volume, Variety, Velocity) [61]. The first authors learned of the Big Data Vs after drafting the MLOps Vs and were surprised to find similarities around volume/versioning and velocity.

Shankar et al. [2022a] (under review)

# MLOps Research Roadmap

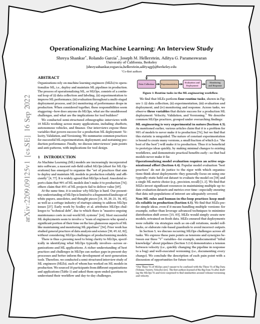

Shankar et al. [2022a] (under review)



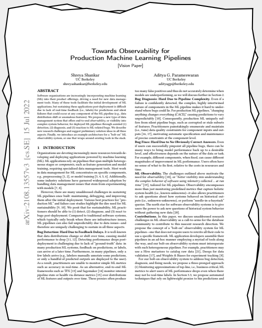

Shankar et al. [2022b] (VLDB 2023)

Shankar and Garcia et al.

# MLOps Research Roadmap


Shankar et al. [2022a] (under review)

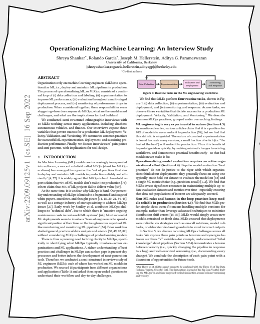



Shankar et al. [2022b] (VLDB 2023)

Shankar and Parameswaran [2022] (VLDB 2023)

Shankar and Garcia et al.

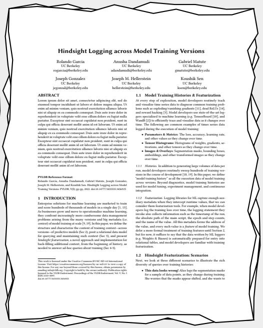
# MLOps Research Roadmap

Operationalizing Machine Learning: An Interview Study

Shreya Shankar*, Rolando Garcia*, Joseph M. Hellerstein, Aditya G. Parameswaran

Shankar et al. [2022a] (under review)



Hindsight Logging across Model Training Versions

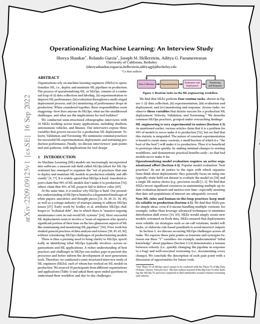
Shankar et al. [2022b] (VLDB 2023)

Shankar and Parameswaran [2022] (VLDB 2023)

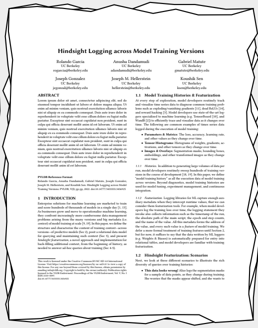
Hindsight logging in model training (in progress)

Shankar and Garcia et al.

# MLOps Research Roadmap


Shankar et al. [2022a] (under review)




Shankar et al. [2022b] (VLDB 2023)

Shankar and Parameswaran [2022] (VLDB 2023)

Hindsight logging in model training (in progress)


Automatic data validation (in progress)

Mining feedback delays and estimating unlabeled accuracy (in progress)

Shankar and Garcia et al.

# Thank you!

Shreya Shankar and Aditya Parameswaran. Towards observability for production machine learning pipelines, 2022. URL https://arxiv.org/abs/2108.13557.

Shreya Shankar, Rolando Garcia, Joseph M. Hellerstein, and Aditya G. Parameswaran. Operationalizing machine learning: An interview study, 2022a. URL https://arxiv.org/abs/2209.09125.

Shreya Shankar, Stephen Macke, Sarah Chasins, Andrew Head, and Aditya Parameswaran. Bolt-on, compact, and rapid program slicing for notebooks [technical report]. 2022b.