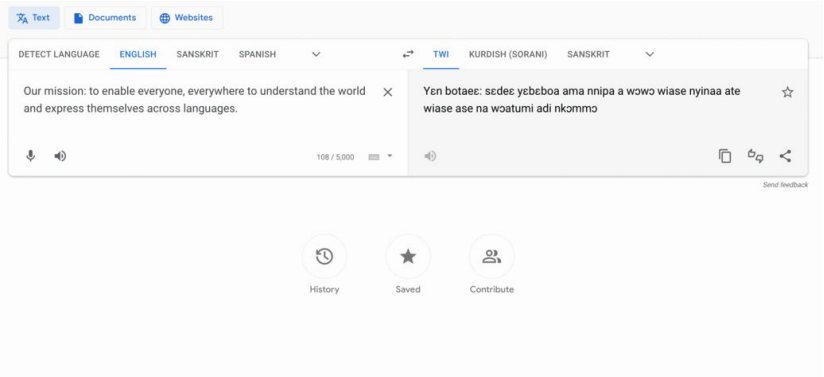
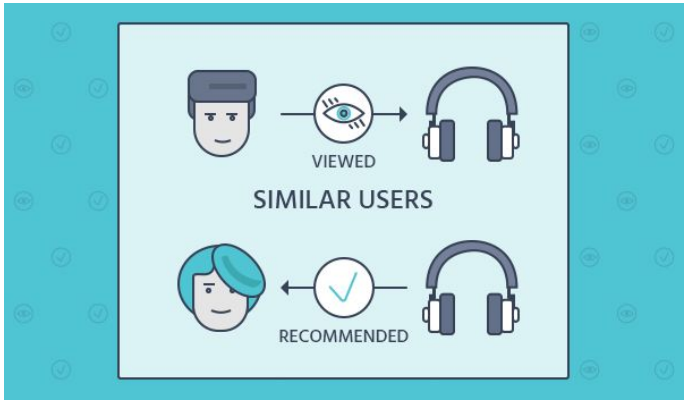
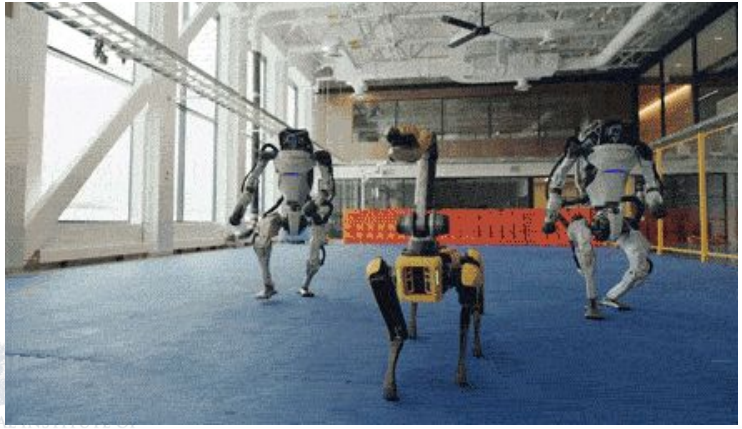
Modern ML Systems



Google translate



Recommender systems



Boston Dynamics

Tesla car in Autopilot

Modern ML Systems

GPT-4

What is an ML-enabled system?

An "ML-enabled system" refers to a system that integrates Machine Learning (ML) techniques to enhance its capabilities, automate tasks, or make data-driven decisions. In essence, ML enables a system to learn from data rather than explicit programming, allowing it to adapt over time as it processes more data. Here's what characterizes an ML-enabled system:

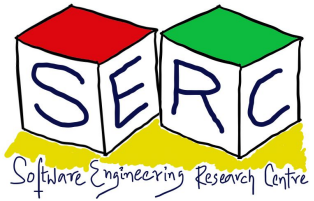
1. **Adaptive Behavior**: ML-enabled systems can modify their behavior based on the patterns they identify in data. This adaptability is the fundamental essence of machine learning.
2. **Data Dependency**: These systems heavily rely on data. The quality and quantity of this data can greatly influence their performance. Given accurate and relevant data, an ML system can provide valuable insights or achieve complex tasks.
3. **Model Training**: Before deploying, the system's model usually undergoes a training phase using a dataset. During training, the model learns patterns, relationships, and structures within the data.
4. **Continuous Learning**: Some ML-enabled systems can continue to learn and adapt even after their initial training. This is often seen in systems that implement online learning or utilize feedback loops.

https://www.indiatoday.in/technology/news/story/openai-could-go-bankrupt-by-2024-as-chatgpt-costs-over-rs-5-crore-to-operate-everyday-2420302-2023-08-13

Observability and Maintainability of ML-enabled Systems

Karthik Vaidhyanathan

https://karthik.vaidhyanathan.com



About me

Assistant Professor, SERC
IIIT Hyderabad
Hyderabad, India
https://karthikvaidhyanathan.com

Education

AMRITA VISHWA VIDYAPEETHAM

GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

Industry

Infosys

Knowledge Lens
A ROCKWELL AUTOMATION COMPANY

FOUNDING MINDS

choira

Community

ICSA 23
13-17 MARCH 2023 | L'AQUILA - ITALY

ICSA 2024

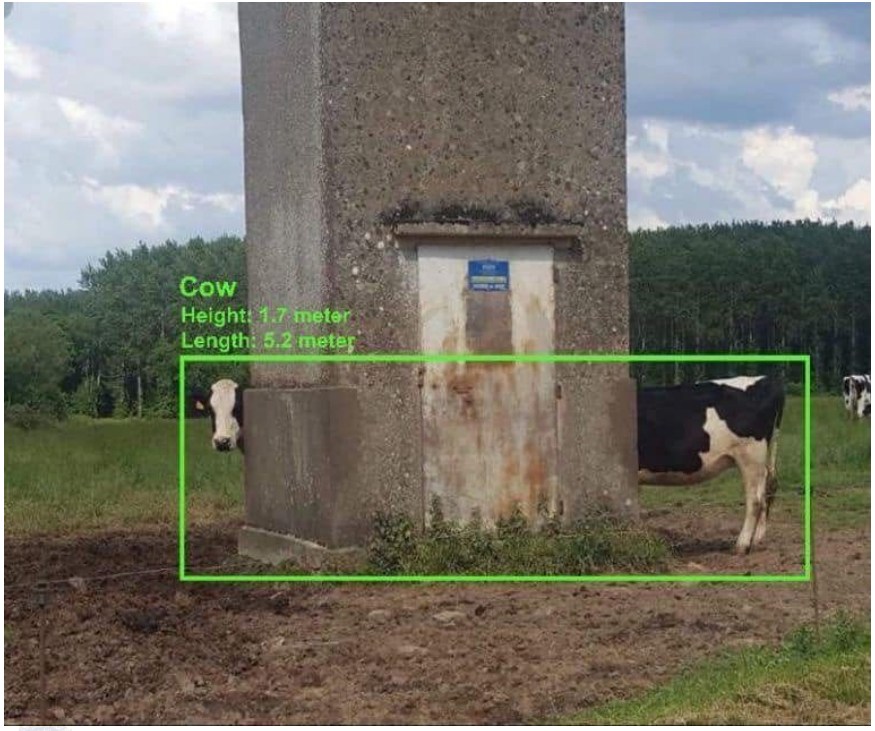
CAIN 2023

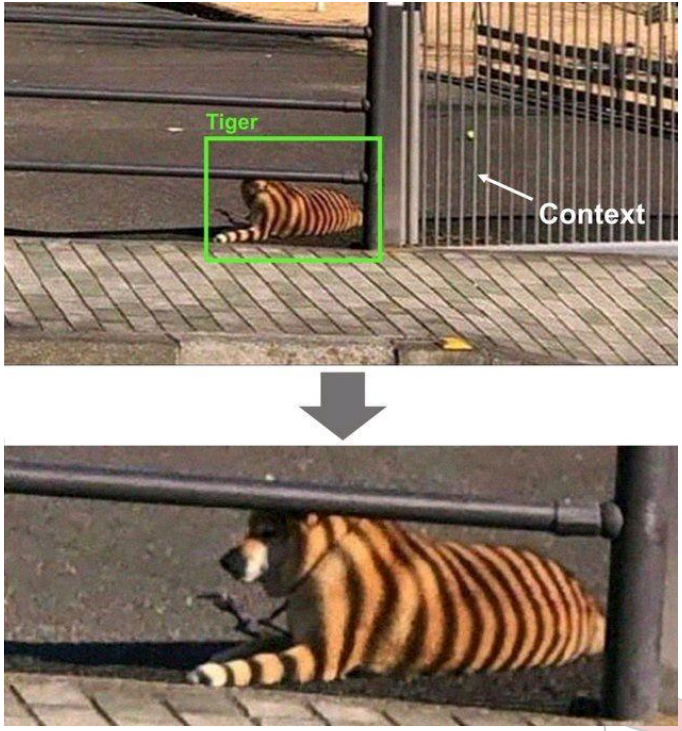
ACM CompEd 2023

14-18 SEPTEMBER 2020 L'AQUILA | ITALY VIRTUAL CONFERENCE
ECSA 2020

Various Journals

ML is not Always Right!





Beware - This can happen

‘skiing’ (.937) → ‘snow’ (.982)

‘girl’ (.660) → ‘photography’ (.738)

‘tennis’ (.982) → ‘sports’ (.989)

‘neighbourhood’ (.925) → ‘blue’ (.927)

Classification model (November 2018 to March 2019)

Cummaudo, Alex, Scott Barnett, Rajesh Vasa, John Grundy, and Mohamed Abdelrazek. **Beware the evolving ‘intelligent’web service! An integration architecture tactic to guard AI-first components**. FSE 2020

ML is not Always Right!

Vox recode

Tesla needs to fix its deadly Autopilot problem

Tesla is facing heat from federal officials following an investigation into a fatal crash involving its Autopilot.

By Rebecca Heilweil | Feb 26, 2020, 1:50pm EST

MIT Technology Review Topics

IAN WALDIE/GETTY IMAGES

Tech policy / AI Ethics

AI is sending people to jail—and getting it wrong

Using historical data to train risk assessment tools could mean that machines are copying the mistakes of the past.

by Karen Hao January 21, 2019

THE VERGE TECH REVIEWS SCIENCE CREATORS ENTERTAINMENT VIDEO MORE

TECH ARTIFICIAL INTELLIGENCE

TL;DR

AI camera operator repeatedly confuses bald head for soccer ball during live stream

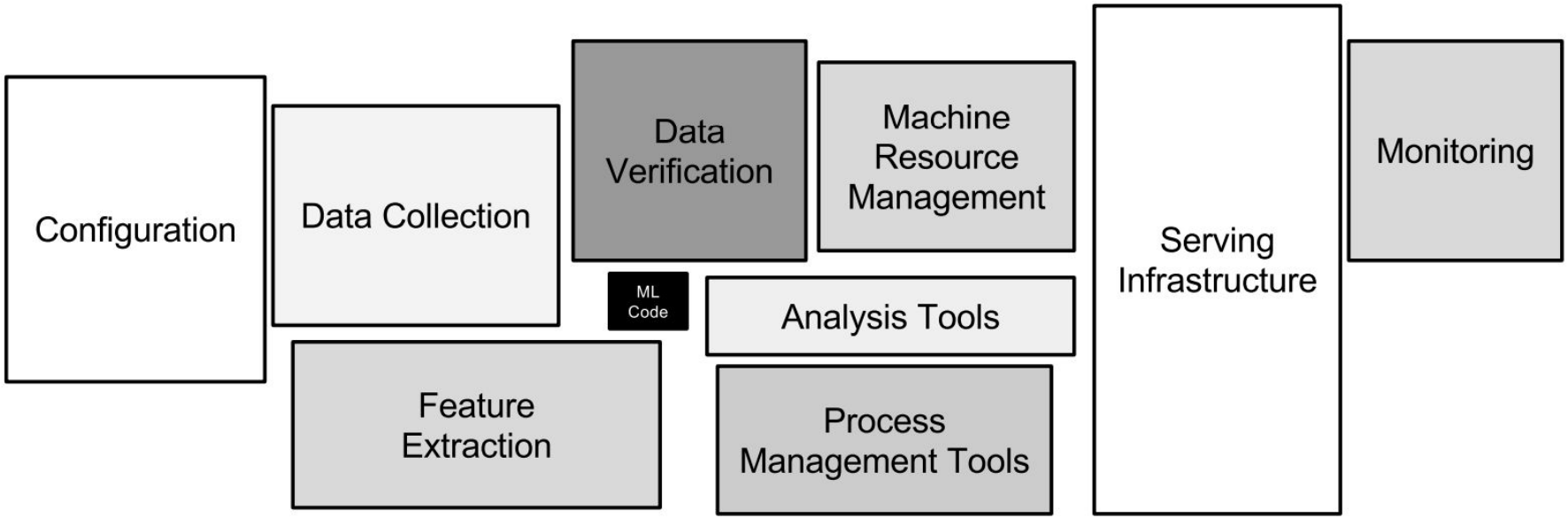
Like a distracted AI with a crush

By James Vincent | Nov 3, 2020, 8:07am EST

1. https://www.vox.com/recode/2020/2/26/21154502/tesla-autopilot-fatal-crashes
2. https://www.technologyreview.com/2019/01/21/137783/algorithms-criminal-justice-ai/
3. https://www.theverge.com/tldr/2020/11/3/21547392/ai-camera-operator-football-bald-head-soccer-mistakes

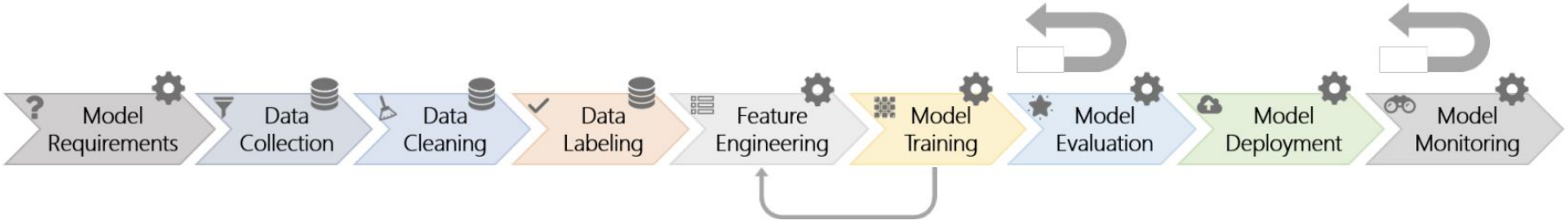
So What is the Problem?

ML System: What it Means?



Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.F. and Dennison, D., 2015. **Hidden technical debt in machine learning systems**. *Advances in neural information processing systems*, *28*.

A Typical ML Workflow

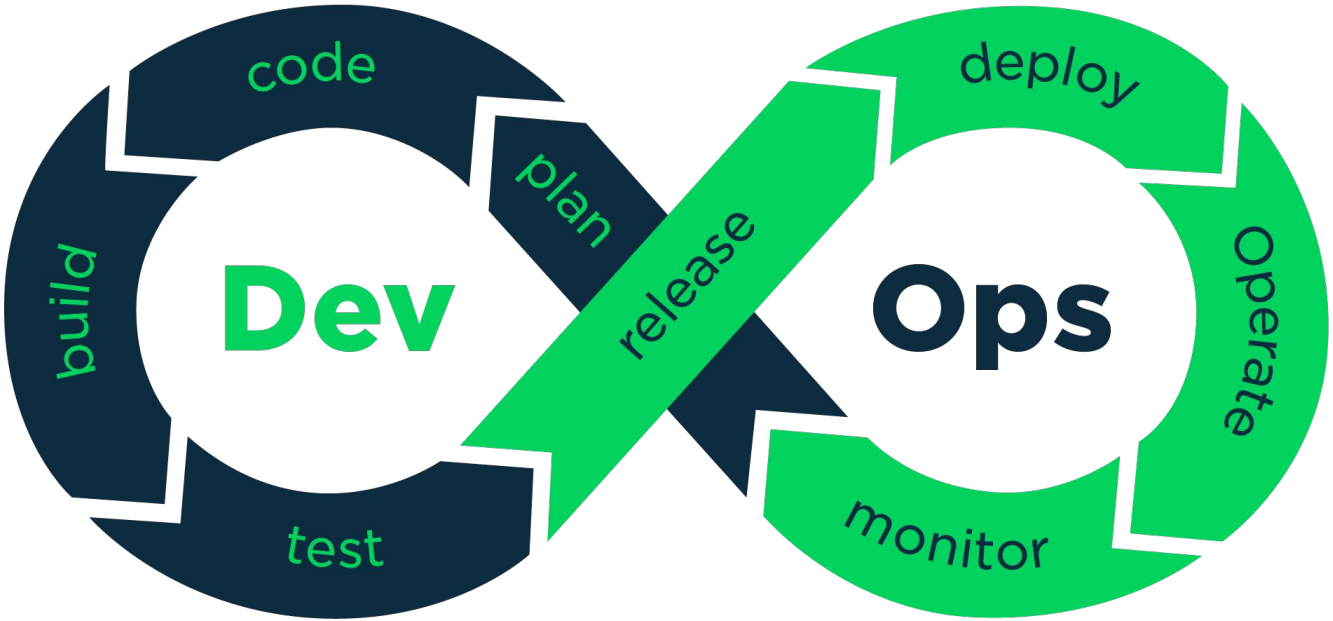


Nine stages of ML workflow

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B. and Zimmermann, T., 2019, May. **Software engineering for machine learning: A case study**. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (pp. 291-300). IEEE

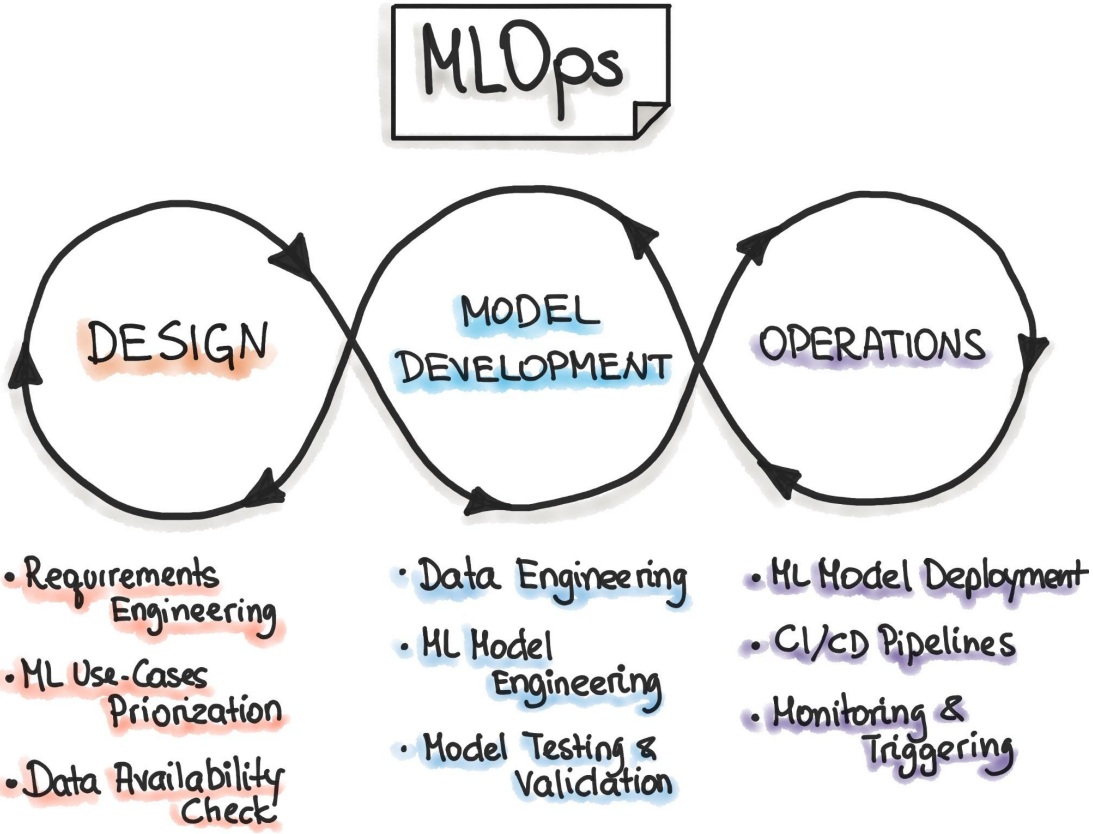
The key part: How to deploy, monitor and maintain?

What did the Software Community do?



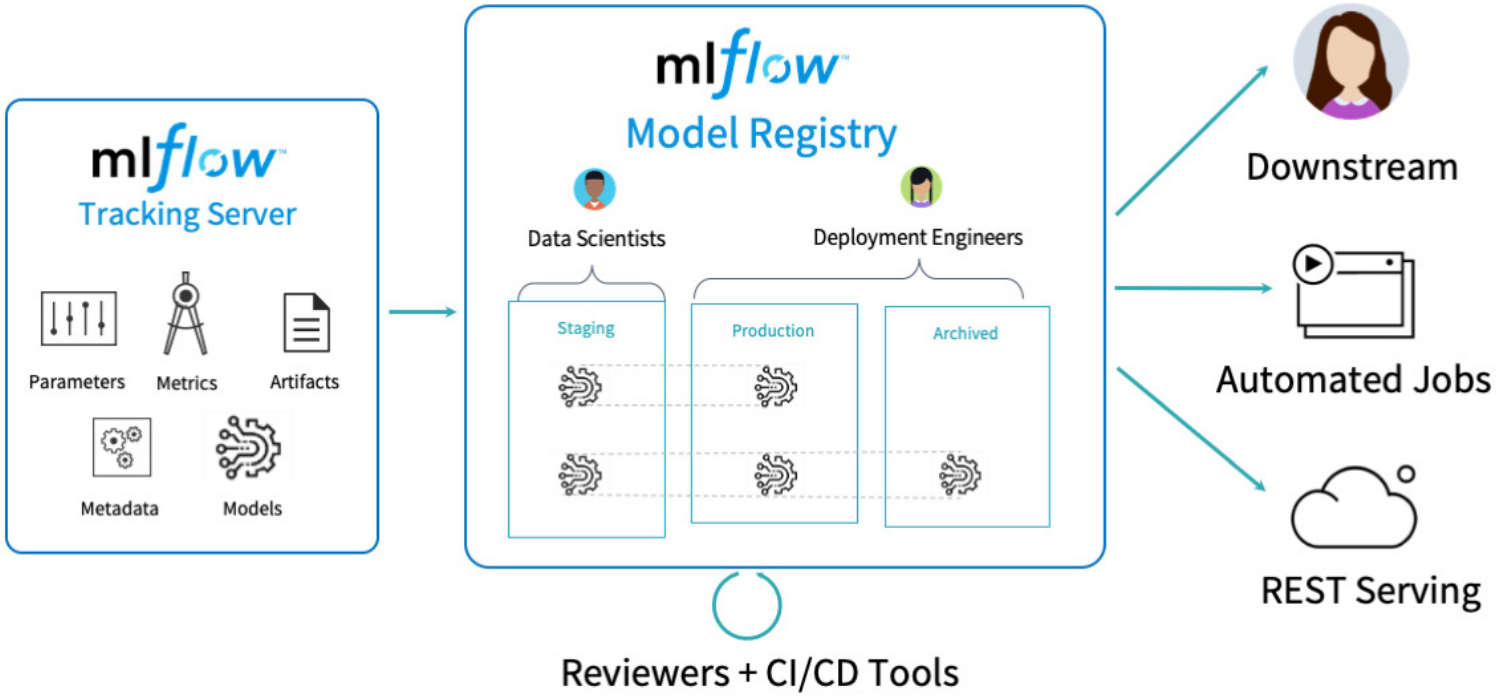
Source: https://devopedia.org/devops

What the ML development teams are doing?



Source: https://Ml-ops.org

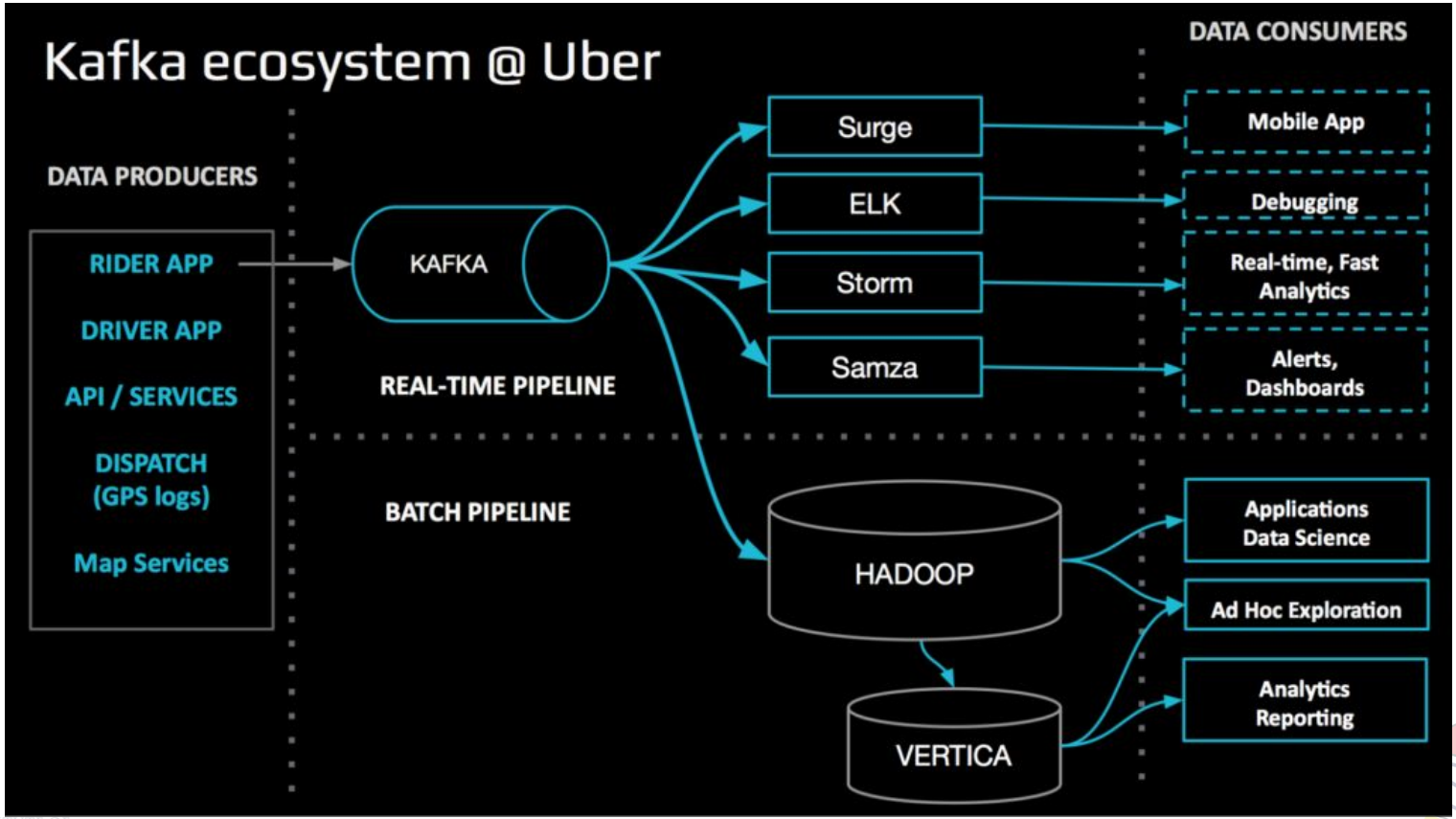
What the ML development teams are doing?



Source: How to Share and Control ML Model Access with MLflow Model Registry - Databricks, MLflow

Is that very Straightforward?

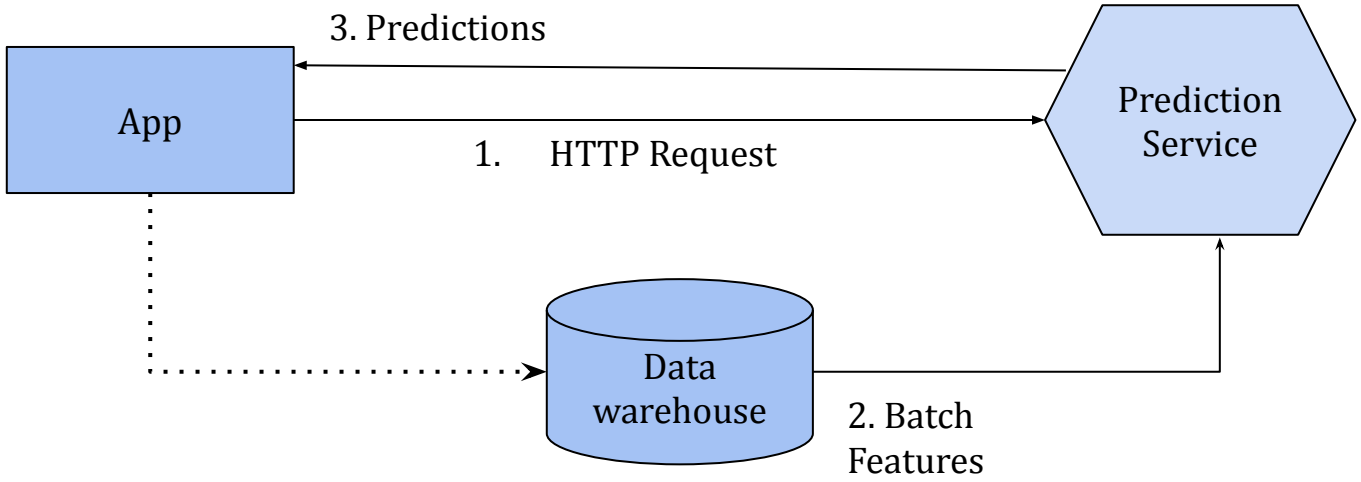
A ML Pipeline flow in Uber



https://databricks.com/blog/2017/04/26/processing-data-in-apache-kafka-with-structured-streaming-in-apache-spark-2.html

Batch Versus Online Predictions

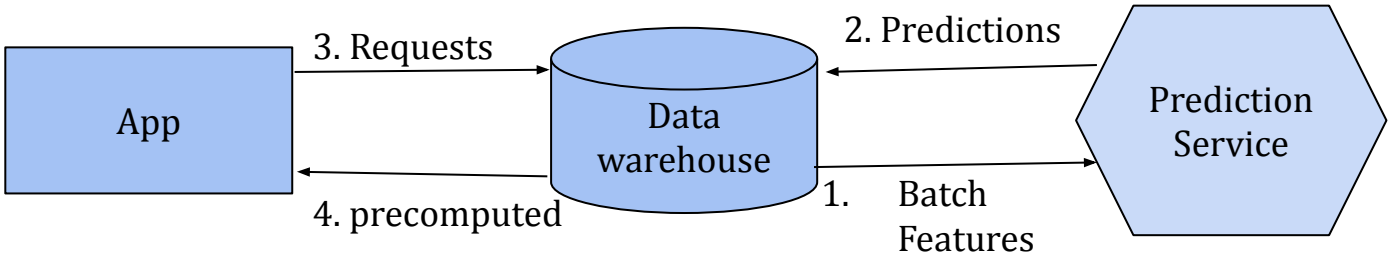
Online Predictions



Example: Near-real time translation in Google Translate

Batch Versus Online Predictions

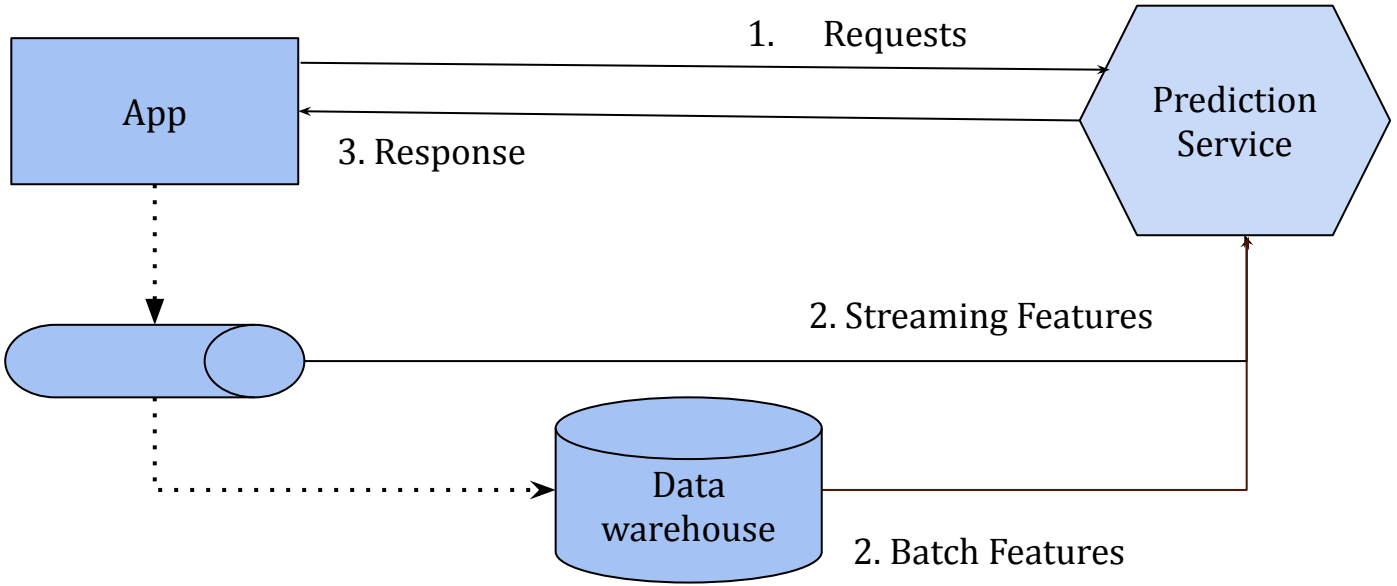
Batch Predictions



Example: Recommendations in Netflix

Batch Versus Online Predictions

Online Predictions (Streaming)



Example: Google maps traffic forecasting

Batch Versus Online Predictions

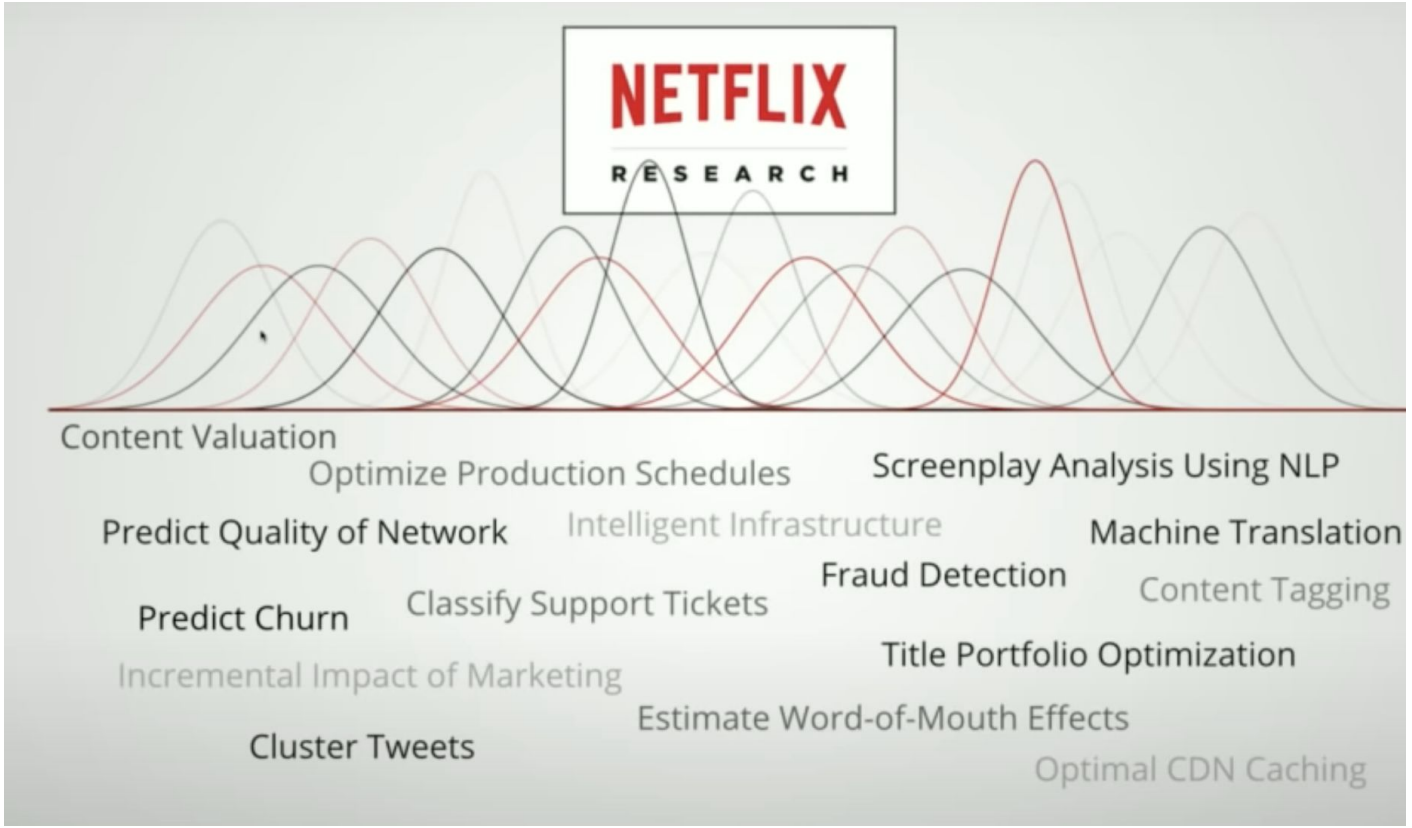
	Batch prediction (asynchronous)	Online prediction (synchronous)
Frequency	Periodical, such as every four hours	As soon as requests come
Useful for	Processing accumulated data when you don't need immediate results (such as recommender systems)	When predictions are needed as soon as a data sample is generated (such as fraud detection)
Optimized for	High throughput	Low latency

Eg: uber eats recommendations

Myth 1: Only one or two models

- Build a model with available data using notebooks
- Often production environment is not just about deploying 1 or 2 models
- In reality companies have many ML models: eg: Uber
  - Models for Ride demand, driver availability, ETA, etc.
  - If operational in 20 countries, each requires its model(s)
  - 20 countries, 10 ML tasks => 200 models!!!
- Need for best practices!

Myth 1: Only one or two models



Different tasks that leverage ML at Netflix

Myth 2: Model Performance remains Same

- Software doesn't age like wine - it ages very poorly
- Things become even worse with ML systems - distribution shifts, concept drift, etc.
- ML model performs wells right after the training then degradation starts!

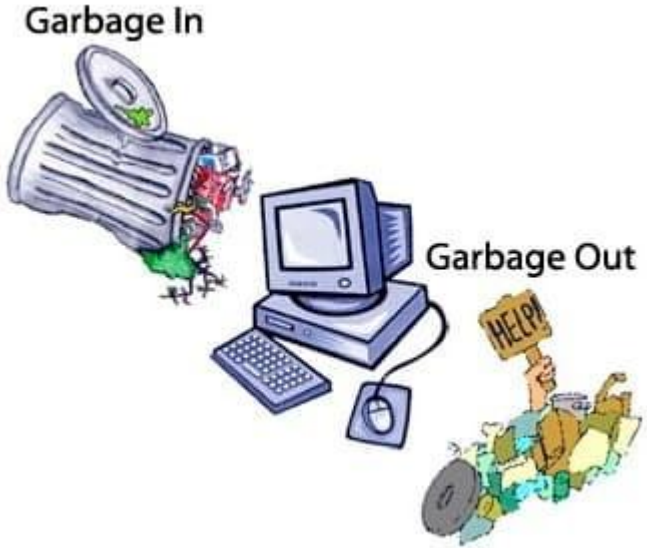
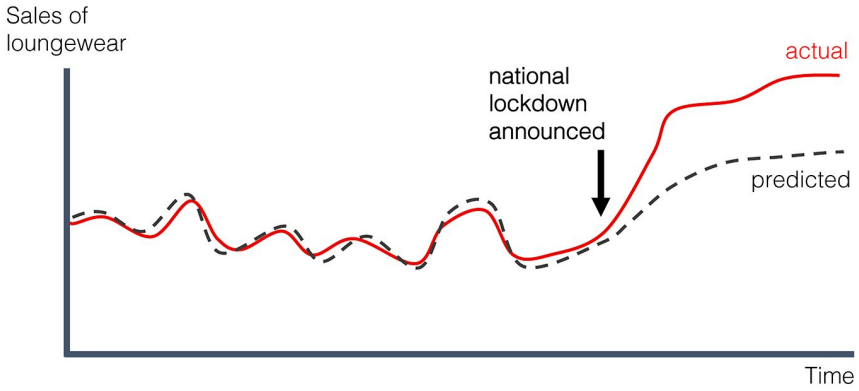


image source: google images

Myth 2: Model Performance remains Same



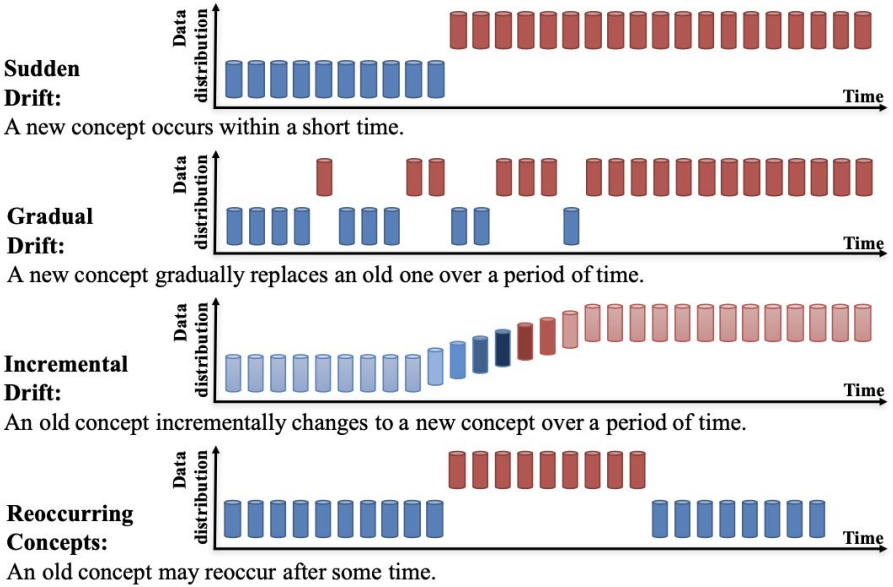


image source: evidentlyai.com, https://arxiv.org/pdf/2004.05785.pdf

Myth 3: Not Many Model Updates

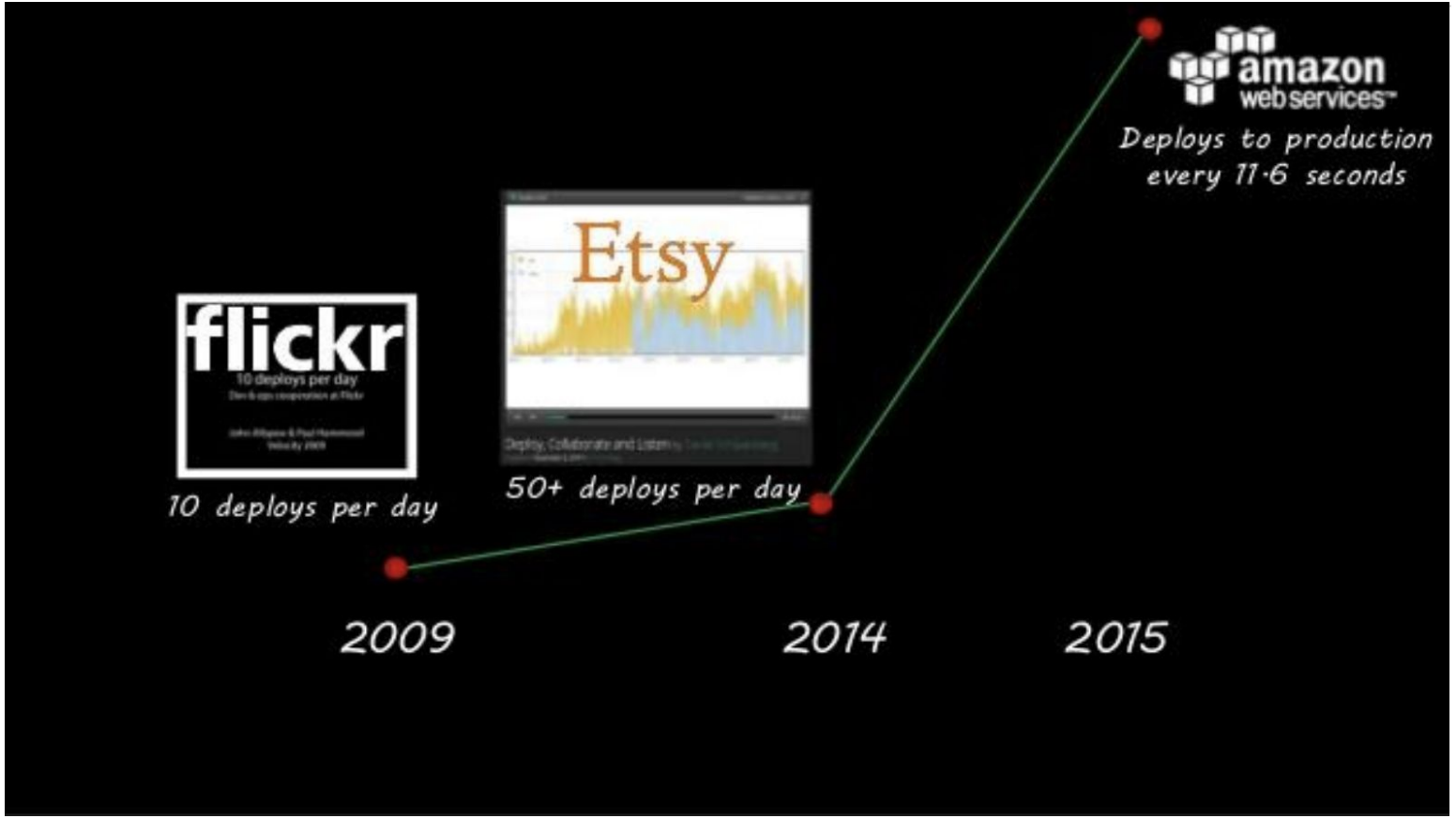


image source: Talk by Josh Wills, QCon.ai

Myth 3: Not Many Model Updates

- Question: How often *should* I update my models? **- Wrong!!**

- Rather: How often can I update my models? - **As often as possible!!**

- ML performance decays over time - **Don't wait!**

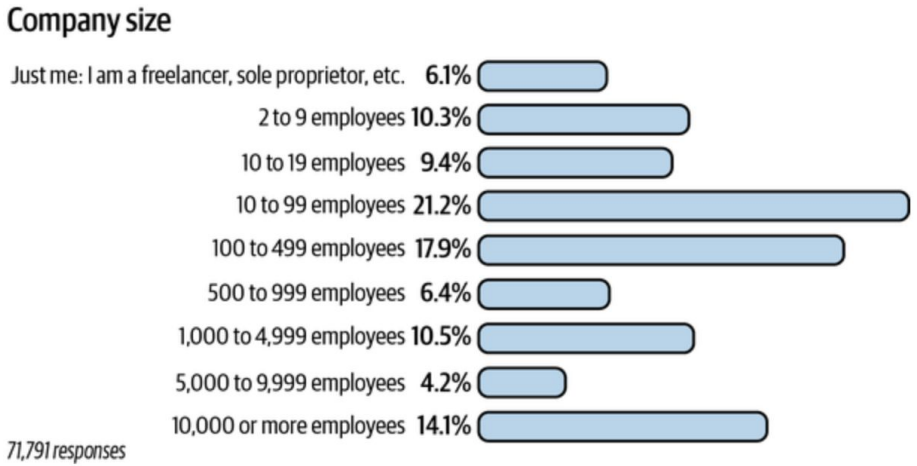
- ML community can take inspiration from DevOps practices
  - Some companies update models once a month
  - Some update every 10 minutes (eg: weibo, Alibaba, Bytedance (TikTok))

image source: evidentlyai.com, https://arxiv.org/pdf/2004.05785.pdf

Myth 4: ML Engineers need not worry about scale

- Scale is a very relative term - users, application domain, etc.
  - Serves 100s of queries per second
  - Serves millions of users per month

- It's not just about Google, Microsoft, Facebook, etc.



Think scale!!

Why Don’t We Monitor the System?

Wait! What do we Monitor?

In a typical Software System, following needs to be monitored:

1. Latency - Time taken to serve a request
2. Traffic - Demand on the system
3. Error - Rate of HTTP request failures, any internal server errors, etc.
4. Saturation - How full a program/service is?

How can this be monitored?

We need Observability!

Observability: What is it about?

Which of these apple would you prefer and why?

Observability: What is it about?

- Observability is a quality that any software system must possess
- Extent to which we can understand the internal state of the system from external outputs
- The more observable the system is, the more easier to understand and solve issues
- When designing and building ML System(s), ensure that it provides observability

How to do that?

Observability: The Three Pillars

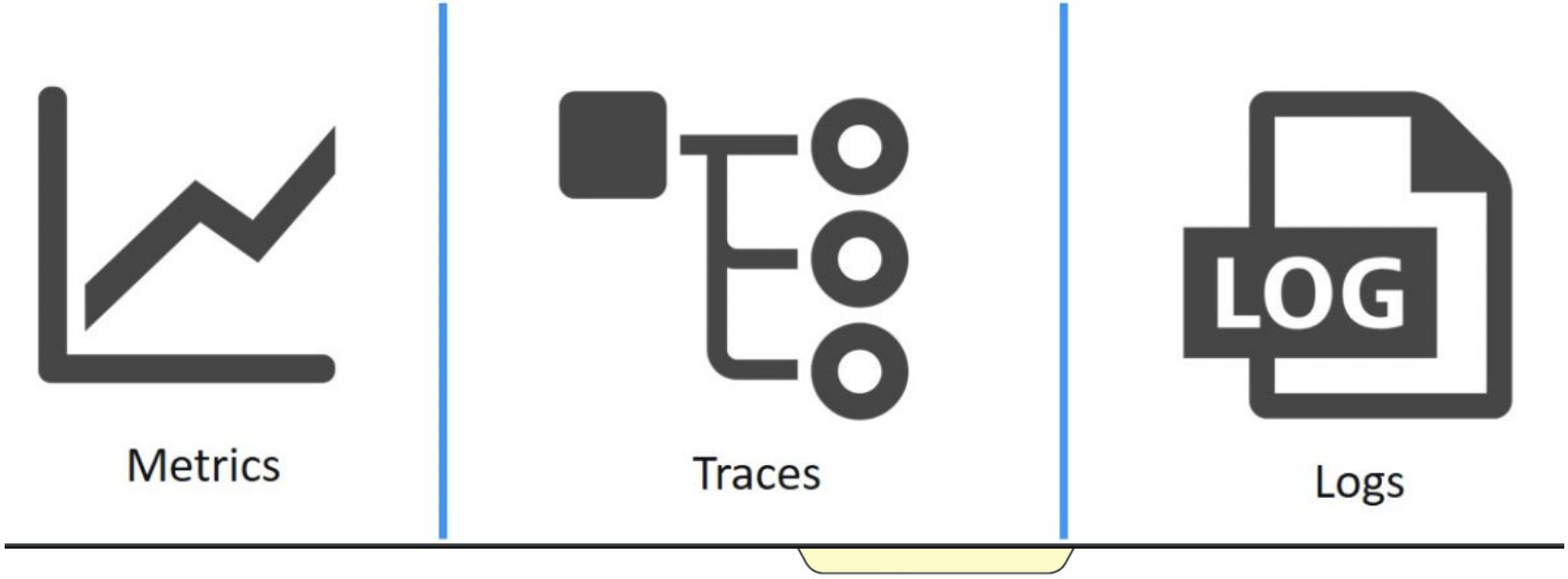
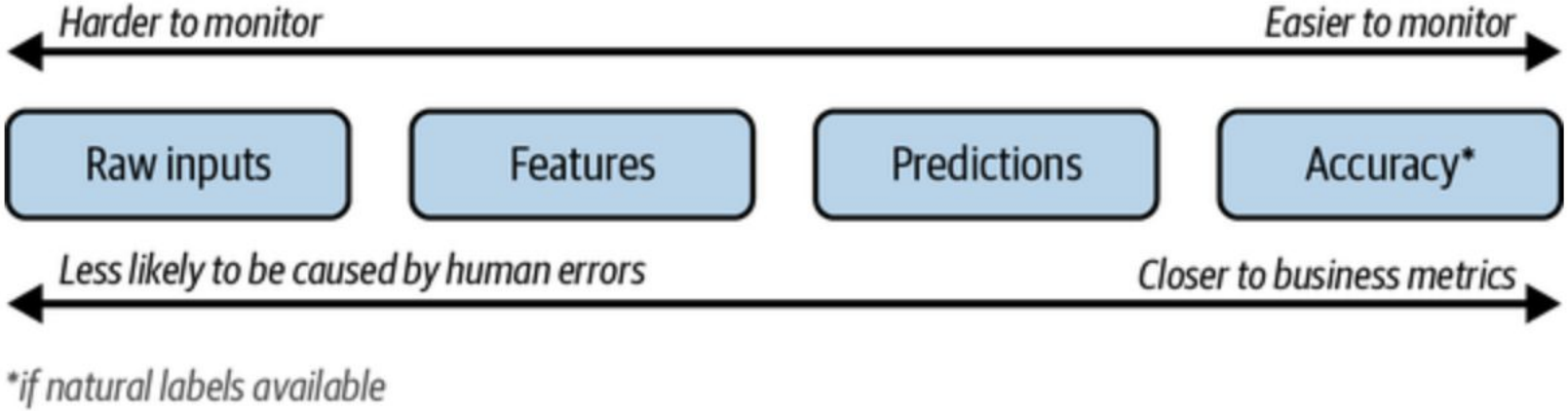


Image Source: medium.com

Metrics to Monitor in Intelligent System

1. Four key artifacts to monitor at different stages of the pipeline
2. The more deeper the artifact - the more transformations - more easy to monitor
3. Errors can be induced in transformation process as well



Credits: designing machine learning systems by Chip Huyen, Oreilly

Monitoring Accuracy Metrics

1. Monitor any user feedback (click, like, upvote, etc.)

2. Natural labels can be inferred from user feedback - **identify degradation!**

3. Youtube recommendation analogy
   a. Track if user clicks on recommended video
   b. Measure how much time user spends on the video and monitor completion rate
   c. If rate drops over time, then time to change the model!

Monitoring Predictions

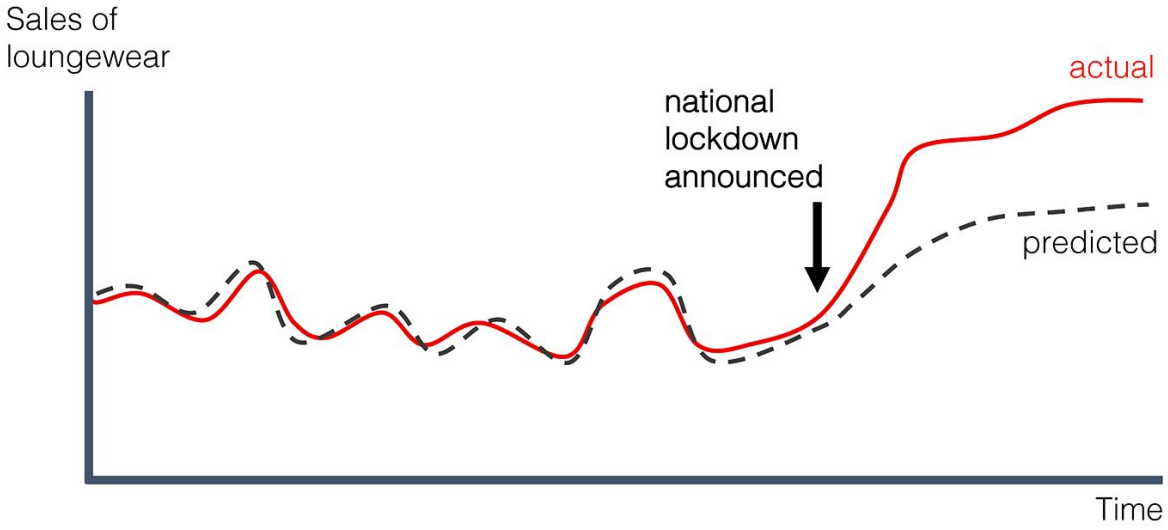
1. Prediction is the most common artifact to monitor

2. Regression task => Continuous value, classification => discrete value

3. Monitor any shifts in the prediction distributions => change in input distribution

4. Odd events could also be identified => continuous wrong prediction, large errors in predictions, etc.

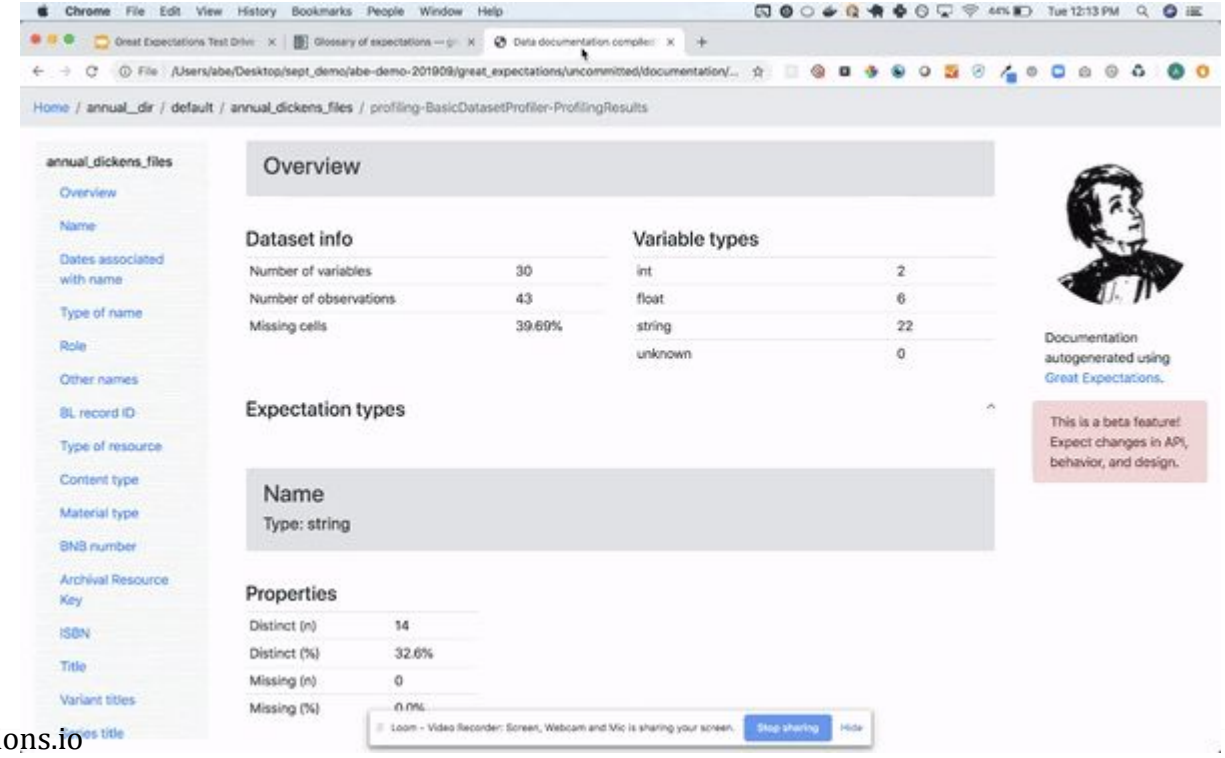
Monitoring Predictions



Source: evidentlyai.com

Monitoring Features

Use tools like Great Expectations - data testing, profiling and documentation



Source: greatexpectations.io

Monitoring Features - Four Concerns

1. A company may have 100s of models and each may have 100s of features
2. Tracking features is good for debugging - Not for model degradation
3. Feature extraction is done as multiple steps in multiple libraries on multiple services
    a. Multiple steps - filling missing values, normalization, etc.
    b. Multiple libraries - Pandas, pyspark, etc.
    c. Multiple services - BigQuery, snowflake, etc.
4. Features schema may change over time - Monitor health of ML systems

Monitoring Raw Inputs

1. Change in features may be due to processing and not due to input data
2. What if Raw input data could be monitored?
   a. Multiple data sources each with different format
   b. Only People with access to data platform can access
   c. Can help in understanding any data drifts - early detection
3. Often complex to achieve

Logging

"If it moves, we track it" - Ian Malpass, Etsy

1. Events that will be produced at runtime needs to be logged
2. Some example log events:
   a. Container starts, memory it takes, etc.
   b. ML processing stage flow, memory, CPU consumption, etc.
   c. Crashes, restarts, errors, etc.
3. Modern software systems generates tremendous amount of logs!!
4. Log management market is estimated around 2.3 billion in 2021 (expected to go upto 4.1 billion by 2026)

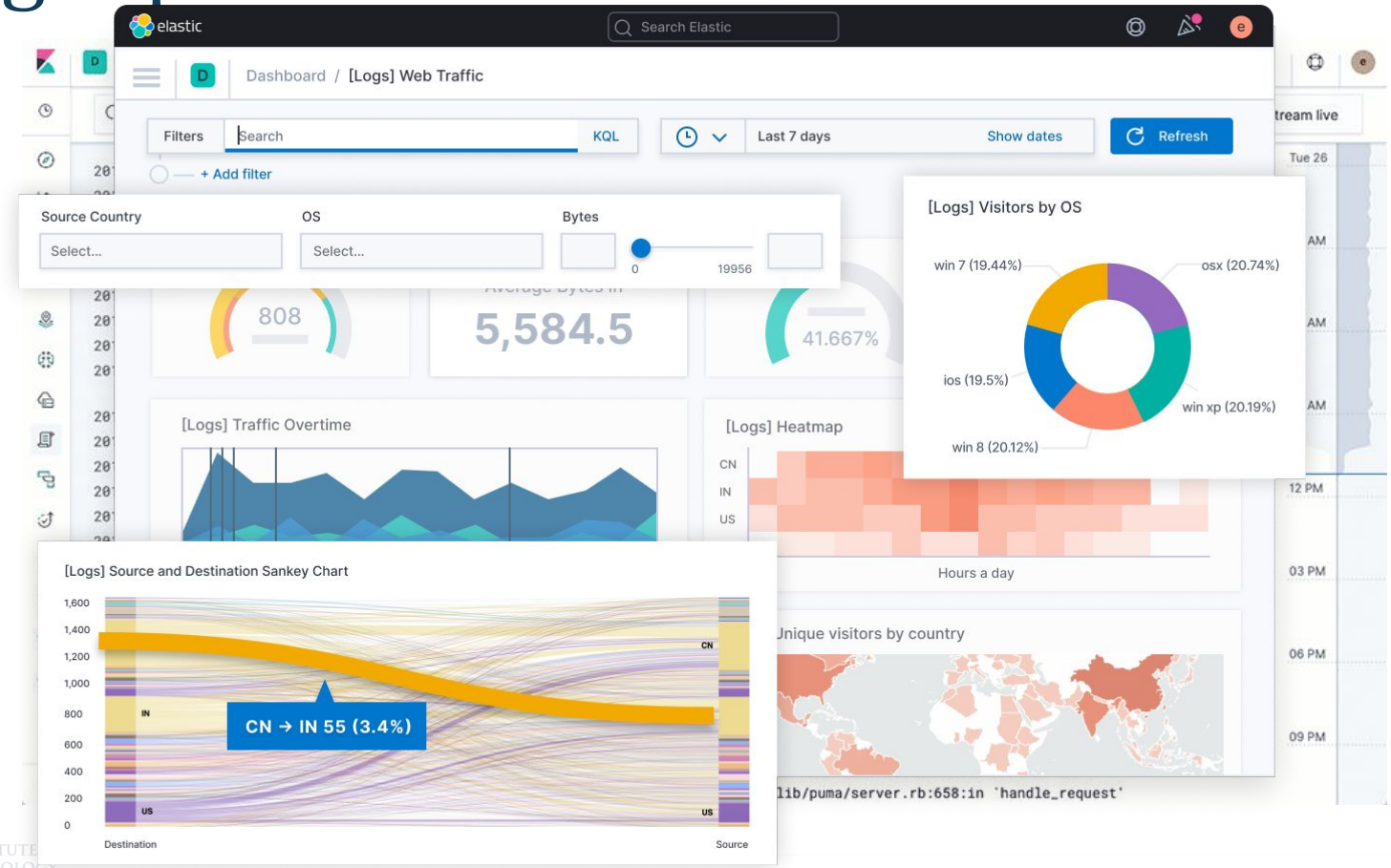
Logging

```
logger.info("Start Cross Validation")

for model_name, model in models.items():
  logger.info("Train {}".format(model_name))

  # cross_val_score for each classifier
  scores = cross_val_score(model, X, y, cv=10, scoring = 'accuracy')

  logger.info("The mean score for {}: {:.3f}".format(model_name, scores.mean()))

  logger.info("------------------------------")
```

```
1  INFO: 2020-02-25 18:32:47,797: Start Cross Validation
2  INFO: 2020-02-25 18:32:47,798: Train KNN
3  INFO: 2020-02-25 18:32:47,906: The mean score for KNN: 0.730
4  INFO: 2020-02-25 18:32:47,907: ------------------------------
5  INFO: 2020-02-25 18:32:47,907: Train RF
6  INFO: 2020-02-25 18:32:49,918: The mean score for RF: 0.777
7  INFO: 2020-02-25 18:32:49,918: ------------------------------
8  INFO: 2020-02-25 18:32:49,919: Train GB
```

Source: medium.com

Log Pipeline



Source: elastic.co

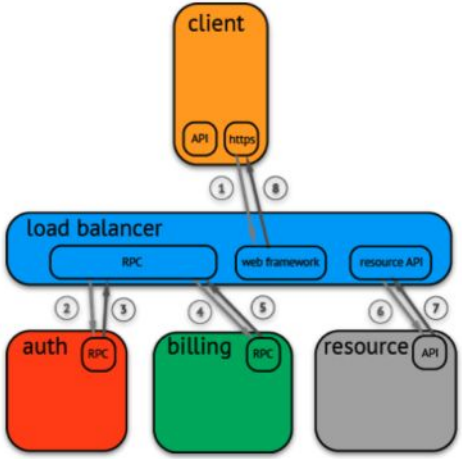
Dashboards help

Source: evidently.ai

Distributed Tracing

Trace tells the story of a transaction or workflow as it propagates through the system

Modern tracing softwares does not take ML explicitly into consideration



Source: epsagon presentation

ML Production Platform

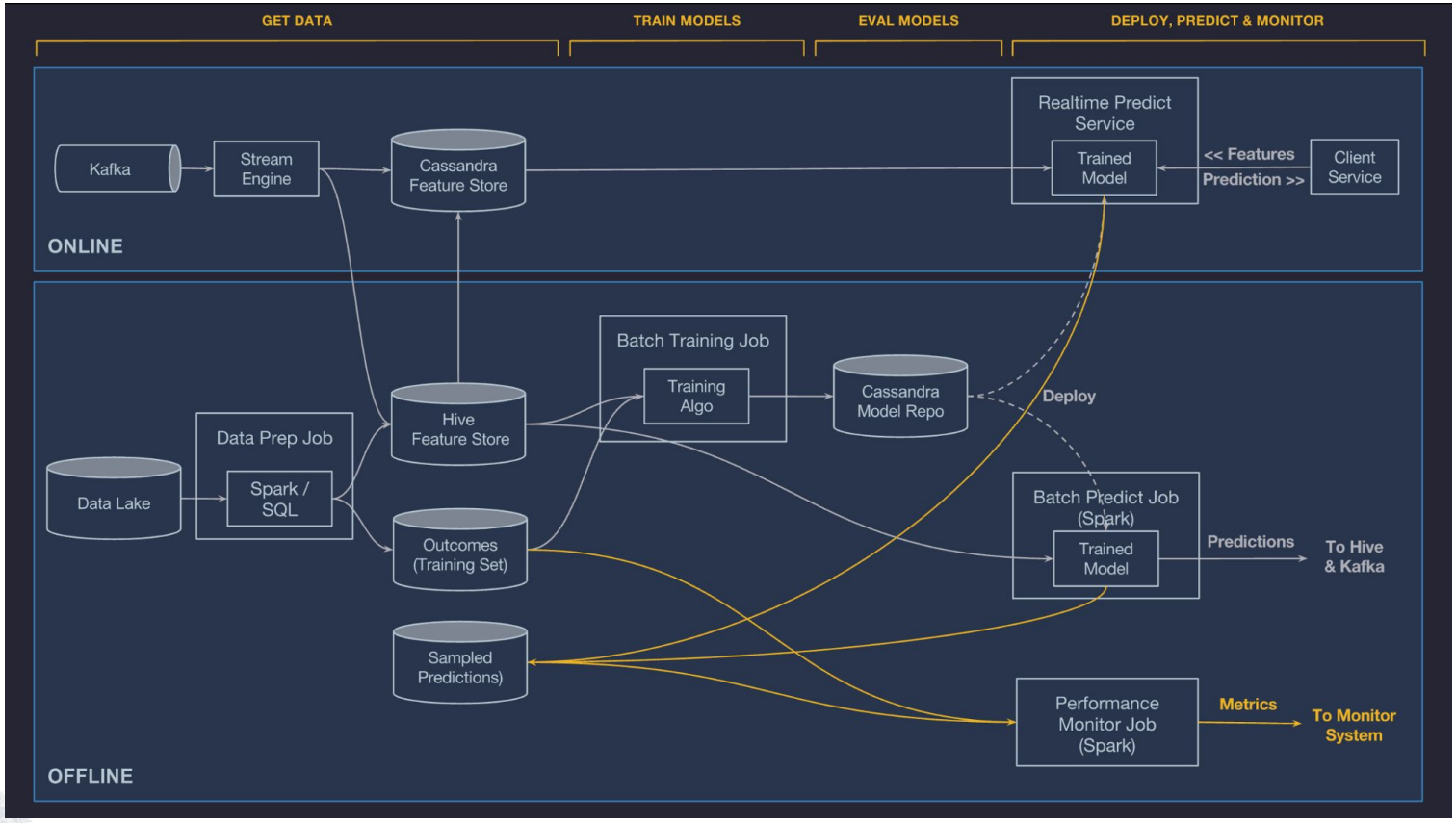


image source: uber.com

What about Maintainability

Maintainability of Software Systems

"*This characteristic represents the degree of* ***effectiveness and efficiency*** *with which a product or system can be* ***modified to improve it, correct it or adapt*** *it to changes in environment, and in requirements*"
[ISO 25000, Software and Data quality]

70-75% of the cost of software goes in maintenance

Maintainability of Intelligent Systems

>50 % of ML systems fails or never make it into production - Gartner

Maintenance of ML systems - Consider software and ML components

60 out of 96 failures were not due to ML itself - Survey on one decade of outages in an ML pipeline at Google, 2020[1]

[1] https://www.youtube.com/watch?v=hBMHohkRgAA

Maintainability Through notion of Technical Debt

What is Technical Debt ?

Software systems are prone to accumulate complexities that makes it difficult to make a change - Cruft

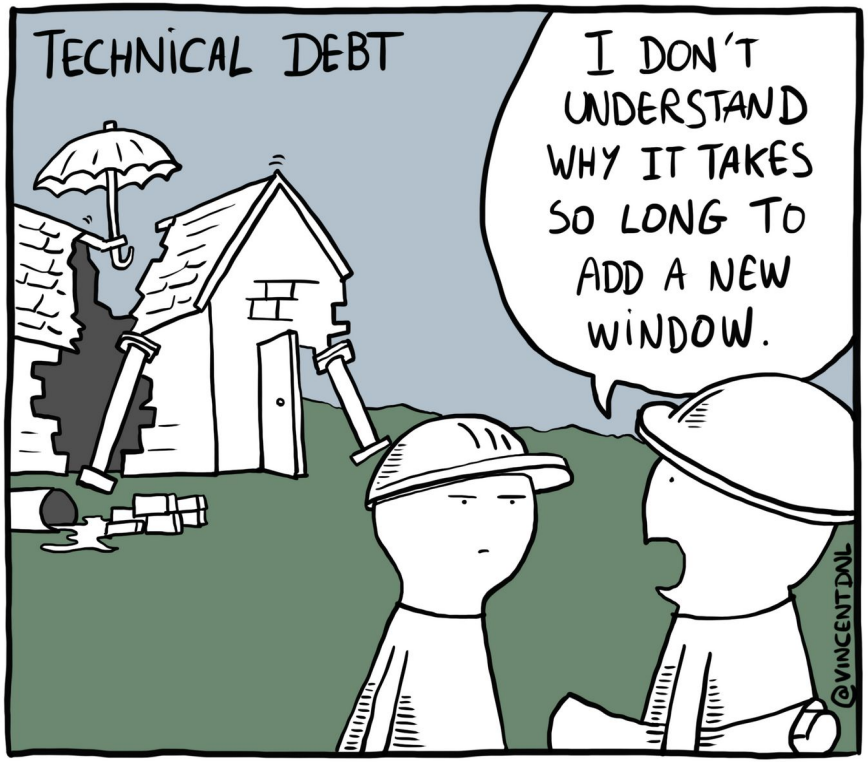
Technical debt - metaphor coined by Ward Cunningham

Think of it as paying financial debt

Extra effort required to build a new feature - interest paid on debt

Source: medium,

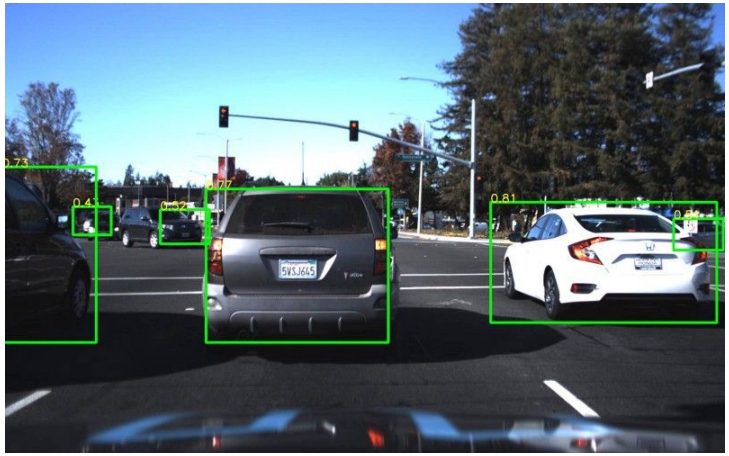
What is Technical Debt ?



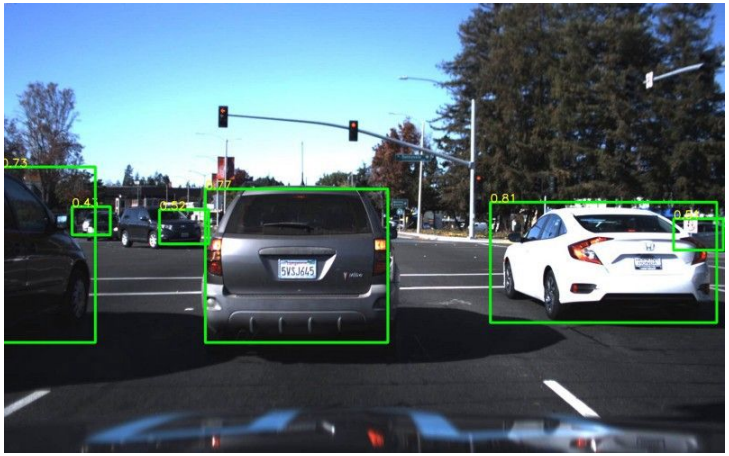
Source: medium,

Complex Models Erode Boundaries

Entanglement



Three cars (Preprocessing A)



Three Vehicles (Preprocessing B)

Change Anything Changes Everything (CACE) Principle (Same input, two different predictions)
Possible solution: Isolate model and serving, Monitor models, keep an eye on data

mage Source: Analytics vidya, the hans india

Complex Models Erode Boundaries

Correction Cascades

Scenario: Create a movie recommendation system. We have a product recommendation algorithm of e-commerce, let's build on that

Problem: Dependency on existing model, leading to improvement deadlocks, more cascading can result in serious problems

Possible Solution: Accept the cost of creating separate model, learn correction within same model to distinguish cases

Complex Models Erode Boundaries

Undeclared Consumers

Scenario: There is a trip forecasting model that provides forecasts on expected trips in future. Let's use that prediction to calculate discount

Problem: Trip forecast model building team may be Unaware of this. Creates a hidden tight coupling

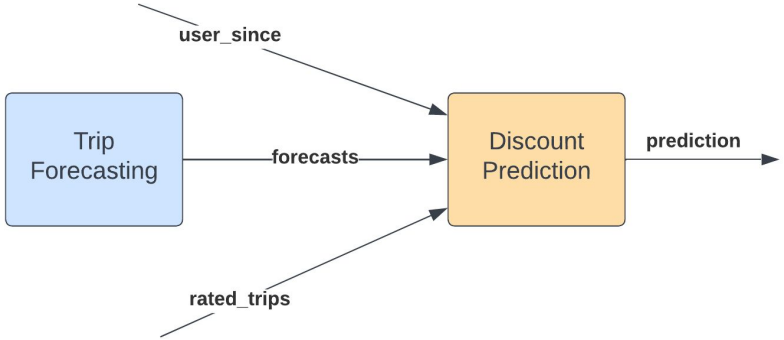
Possible Solution: Access restrictions, strict SLA's.

Costlier Data Dependencies - Dependency Debt

Unstable data dependencies
Prob: Input data keeps changing
Sol: Versioning helps

Underutilized data dependencies
Prob: Many are not needed
Sol: Use only what is needed, leave-one-out feature evaluation.

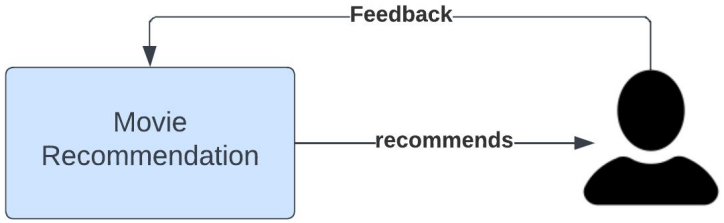
Static analysis of data dependencies can be performed



An example prediction pipeline

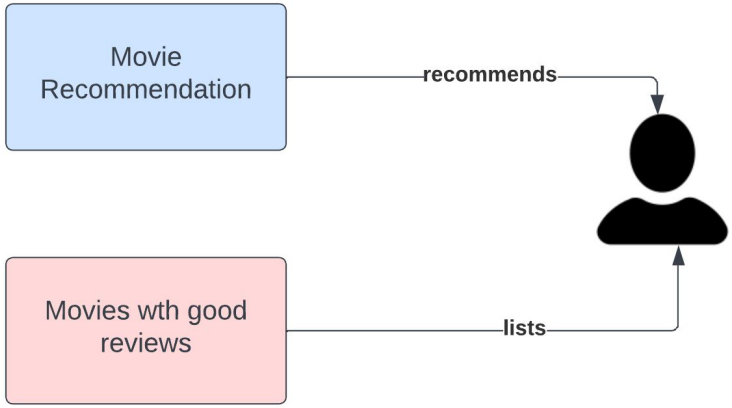
Feedback Loops - Analysis Debt

Direct Feedback Loops
Prob: Prediction influences the next model
Sol: Randomization, isolation of part of data



Hidden Feedback Loops
Prob: Two ML systems influence each indirectly
Two disjoint systems may also influence one another

Sol: Better communication and collaboration between teams



ML Systems - Antipatterns

Glue Code
Using generic libraries can be costly
Try to separate black box and own ML code

Pipeline Jungles
With time => new data sources, more scrape, joins, etc.
Take a clean slate approach when possible

Dead experimental Code Paths
Implementing experimental path in prod code
Knight capital lost 465 million $ in 45 mins
Examine periodically, discard what is not needed

image source: medium

ML Systems - Common Smells

Design smell indicates an underlying problem in code or component

Plain Old Data Type Smell
All data produced/consumed should have clear definition

Multiple Language Smell
Using multiple languages to implement can be costly

Prototype Smell
Prototype solutions should not be used in production

image source: medium

ML Systems - Configuration Debt

ML systems has wide range of configuration options

Some ground rules

1. Easy to specify
2. Hard to make errors
3. Easy to assert and verify
4. Ability to detect unused or redundant setting
5. Code review for configurations

image source: rudder.io

Concluding Thoughts

Engineering ML systems requires careful consideration of some key quality attributes

1. Try to identify the type of architectural pattern that needs to be used
2. Each pattern provides a guarantee on some qualities
3. Ensure system is observable and maintainable (keep debts to minimum)
4. **Lot of research needs to be done**

image source: rudder.io

Thank you

Further queries:

- E-mail: karthik.vaidhyanathan@iiit.ac.in
- karthikv1392@gmail.com
- Web: https://karthikvaidhyanathan.com
- Twitter: @karthi_ishere