

# Software Engineering for ML/AI

Claire Le Goues

**Michael Hilton**

# Administrivia

- Homework 3 out

# Interview: Josh Gardner



# Software Engineering and ML

# Traditional Software Development

“It is easy. You just chip  
away the stone that  
doesn’t look like David.”

–(probably not) Michelangelo

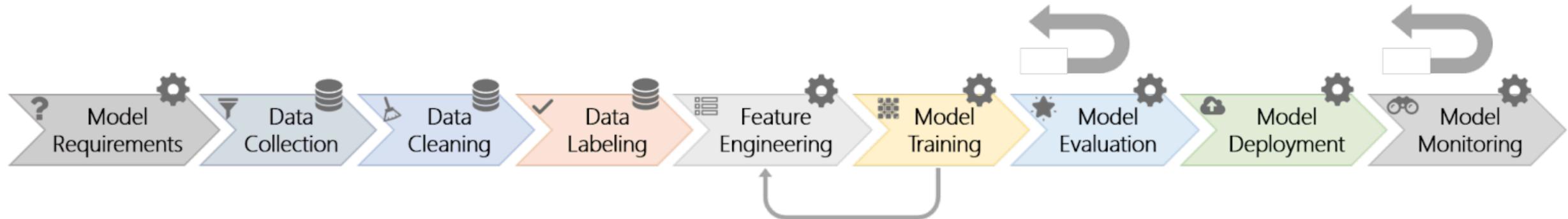


# ML Development

- Observation
- Hypothesis
- Predict
- Test
- Reject or Refine Hypothesis



# Microsoft's view of Software Engineering for ML



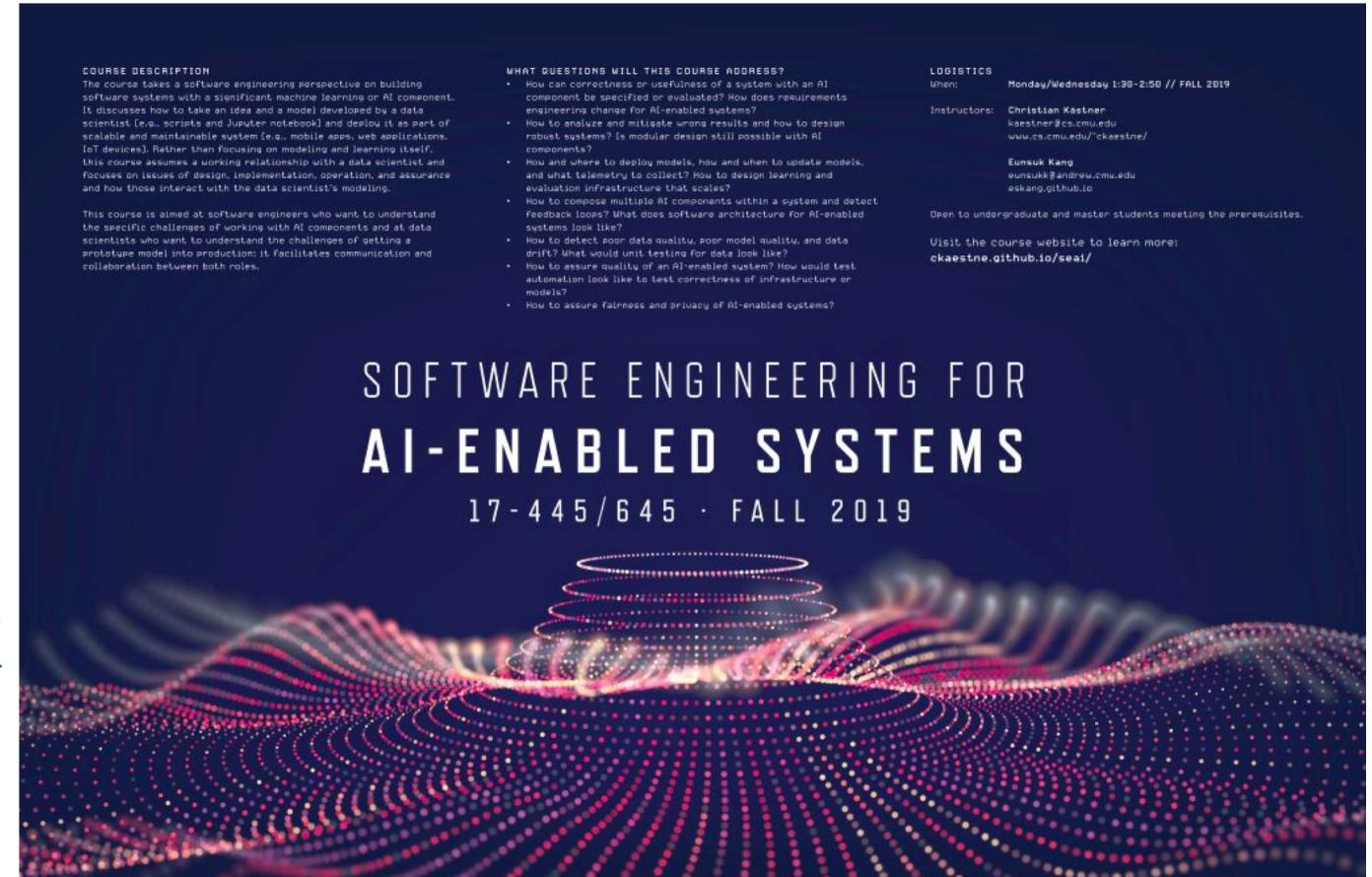
# Three Fundamental Differences:

- Data discovery and management
- Customization and Reuse
- ML Modularity

Case study developed by  
Christian Kästner

<https://ckaestne.github.io/seai/>

## CASE STUDY



The screenshot shows the course website for "Software Engineering for AI-Enabled Systems". The page includes a "Course Description" section, a "What Questions Will This Course Address?" section with a bulleted list of 12 items, and a "Logistics" section with information about the instructor, Christian Kästner, and Eunsuk Kang. Below the logistics is a note about prerequisites and a link to the course website. The background of the page features a dark blue image of a digital wave composed of numerous small glowing dots.

**COURSE DESCRIPTION**  
The course takes a software engineering perspective on building software systems with a significant machine learning or AI component. It discusses how to take an idea and a model developed by a data scientist (e.g., scripts and Jupyter notebook) and deploy it as part of scalable and maintainable system (e.g., mobile apps, web applications, IoT devices). Rather than focusing on modeling and learning itself, this course assumes a working relationship with a data scientist and focuses on issues of design, implementation, operation, and assurance and how those interact with the data scientist's modeling.

This course is aimed at software engineers who want to understand the specific challenges of working with AI components and at data scientists who want to understand the challenges of getting a prototype model into production; it facilitates communication and collaboration between both roles.

**WHAT QUESTIONS WILL THIS COURSE ADDRESS?**

- How can correctness or usefulness of a system with an AI component be specified or evaluated? How does requirements engineering change for AI-enabled systems?
- How to analyze and mitigate wrong results and how to design robust systems? Is modular design still possible with AI components?
- How and when to deploy models, how and when to update models, and what telemetry to collect? How to design learning and evaluation infrastructure that scales?
- How to compose multiple AI components within a system and detect feedback loops? What does software architecture for AI-enabled systems look like?
- How to detect poor data quality, poor model quality, and data drift? What would unit testing for data look like?
- How to assure quality of an AI-enabled system? How would test automation look like to test correctness of infrastructure or models?
- How to assure fairness and privacy of AI-enabled systems?

**LOGISTICS**

When: Monday/Wednesday 1:30-2:50 // FALL 2019

Instructors: Christian Kästner  
kestner@cs.cmu.edu  
www.cs.cmu.edu/~ckestne/  
  
Eunsuk Kang  
eunsukk@andrew.cmu.edu  
euskang.github.io

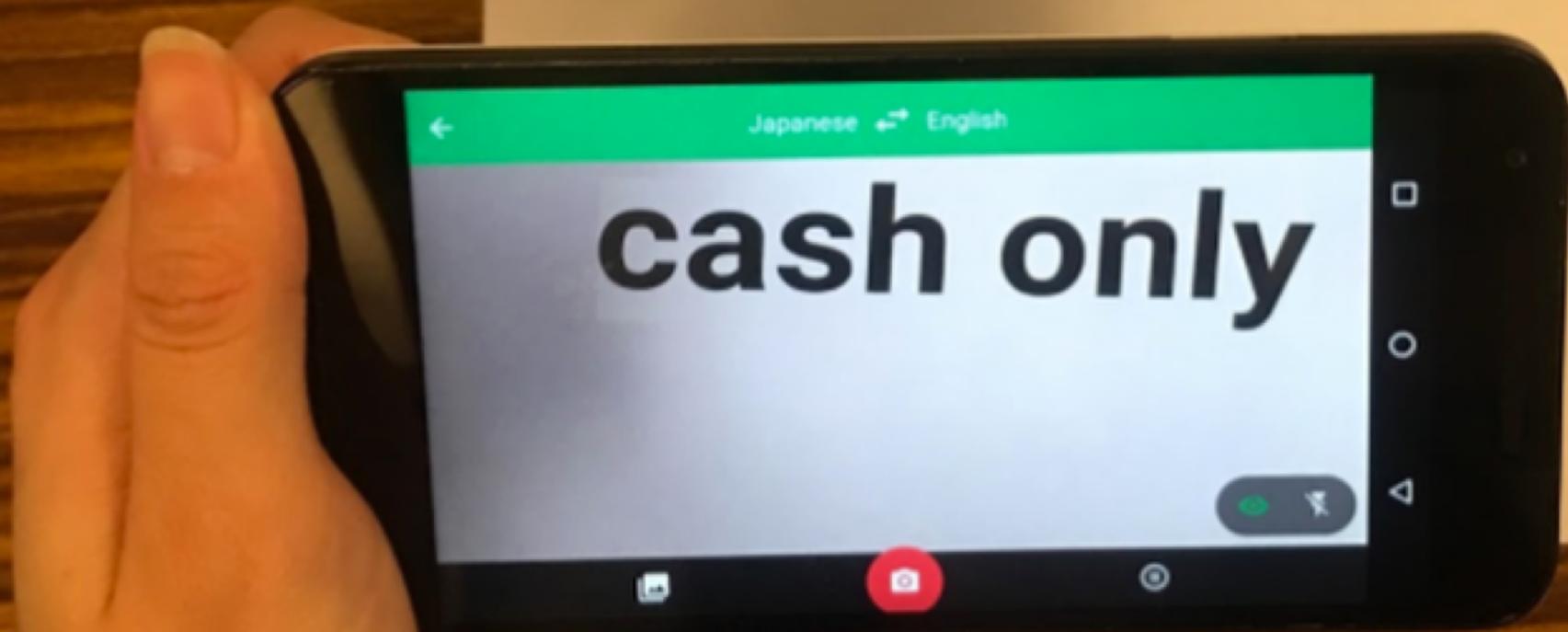
Open to undergraduate and master students meeting the prerequisites.  
Visit the course website to learn more:  
[ckaestne.github.io/seai/](https://ckaestne.github.io/seai/)

**SOFTWARE ENGINEERING FOR  
AI-ENABLED SYSTEMS**  
17-445/645 · FALL 2019

# **WHAT CHALLENGES ARE THERE IN BUILDING AND DEPLOYING ML?**



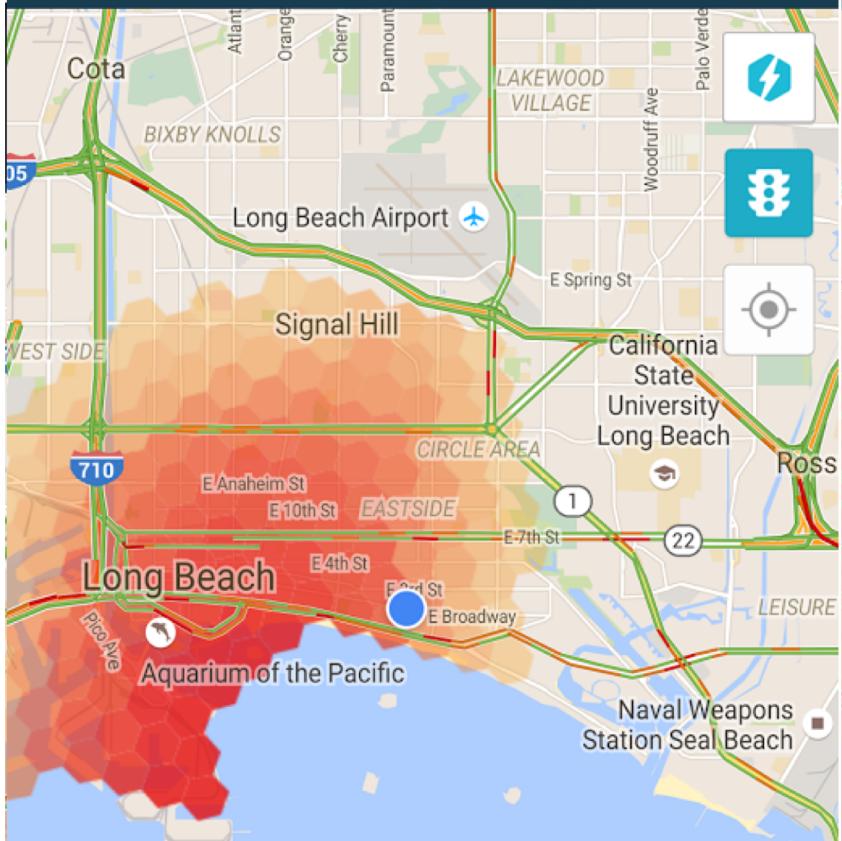
現金のみ





# Qualities of Interest?

GO OFFLINE



Google

CURRENT PROMOTION



HOME



EARNINGS

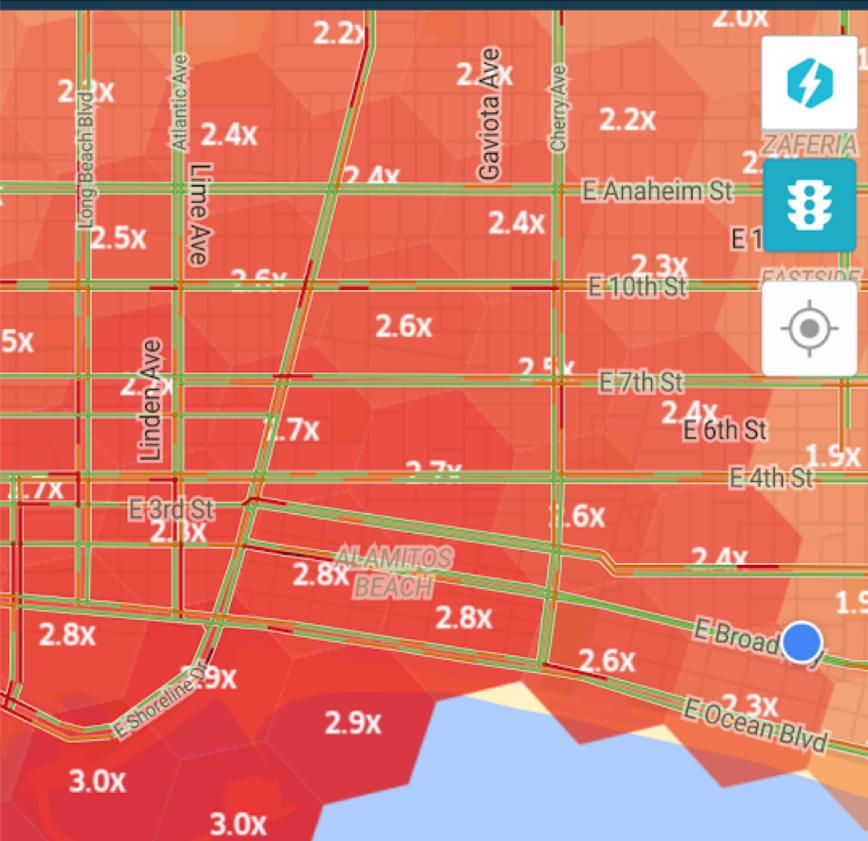


RATINGS



ACCOUNT

GO OFFLINE

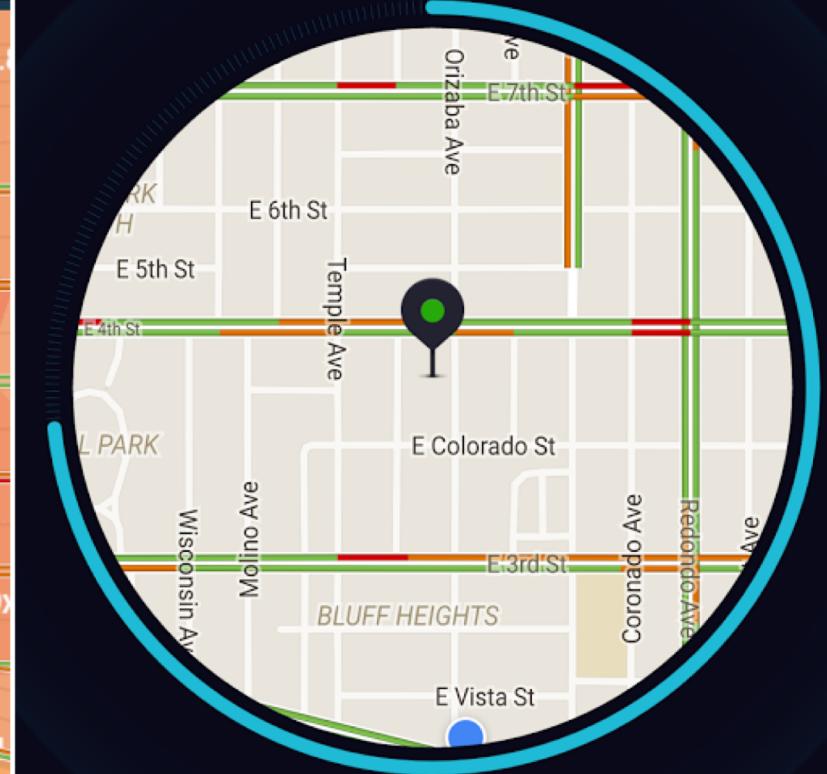


CURRENT PROMOTION



4 MINUTES  
[REDACTED] Ave, Long Beach, CA  
90814, USA

5.0★ | POOL | ⚡ 1.9X



# Qualities of Interest?

# MACHINE LEARNING PIPELINE

# Typical ML Pipeline

- Static
  - Get labeled data
  - Identify and extract features
  - Split data into training and evaluation set
  - Learn model from training data
  - Evaluate model on evaluation data
  - Repeat, revising features
- With production data
  - Evaluate model on production data; monitor
  - Select production data for retraining
  - Update model regularly

# Example Data

<b>UserId</b>	<b>PickupLocation</b>	<b>TargetLocation</b>	<b>OrderTime</b>	<b>PickupTime</b>
5	....	...	18:23	18:31
...				

# Example Data

OCR Helper Tool

Input Image: C:\tmp\MyHandWriting.jpg (Re)Process

Model Params: Load Model

0 Blobs selected

Hover controls for tooltips

Show Binarized Image

Show Rows

Binarization Threshhold: 200

Height Merge Sensitivity: 15

Width Merge Sensitivity: 10

Pre Merge Filter Size: 10

Post Merge Filter Size: 100

Extracted Back Color: 0

Move Selected Blobs

Interval: 2

Export

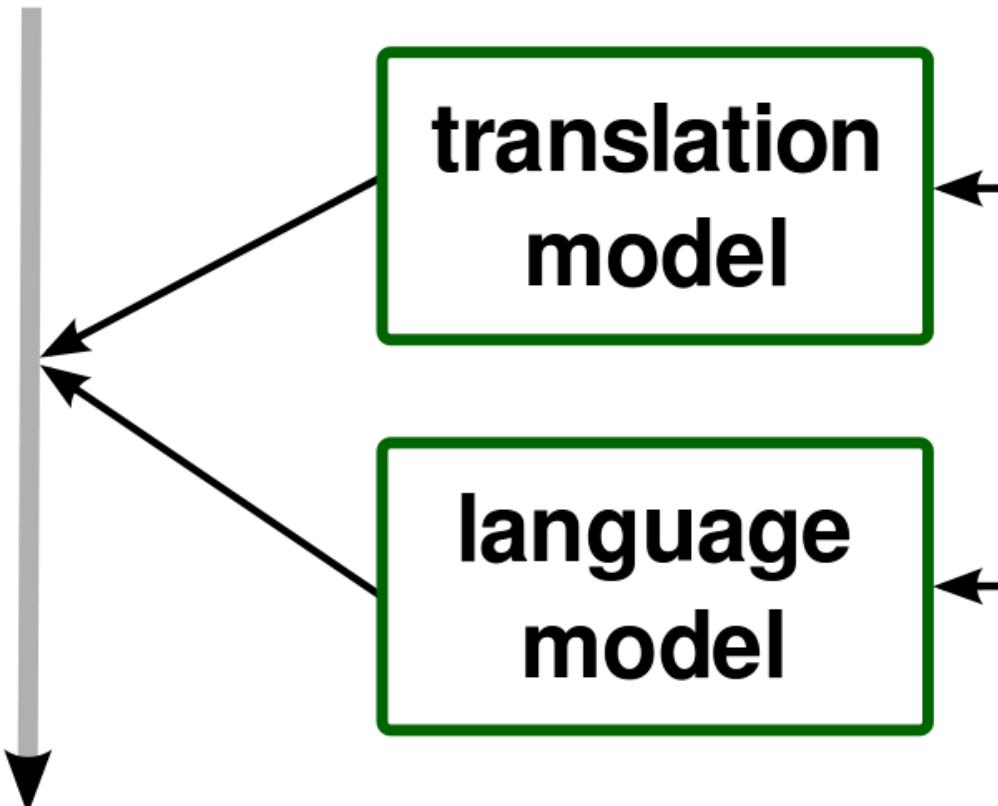
Export Size (W/H): 20

Output:

Export Blobs

# Learning Data

似乎格式有問題



## parallel corpus

网站资讯分析网数  
据显示的主域名为  
全世界访问量最高  
的站点除此之外搜  
索在其他国家或地  
区域名下的多个站  
点等等及旗下的等

The corporation has been estim  
to run more than one million pag  
in data centers around the world  
to process over one billion searc  
requests and about twenty-four i  
of user-generated data each dat  
December 2012 Alexa listed as

## monolingual corpus

started functioning in 1928 and established the tradition of  
large exhibitions and trade fairs held in Brno, and nowadays  
also ranks among the sights of the city. Brno is also  
known for hosting big motorbike and other races on the  
Masaryk Circuit, a tradition established in 1930 in which  
the Road Racing World Championship Grand Prix is  
one of the most prestigious races. Another notable cultural  
tradition is an international fireworks competition.

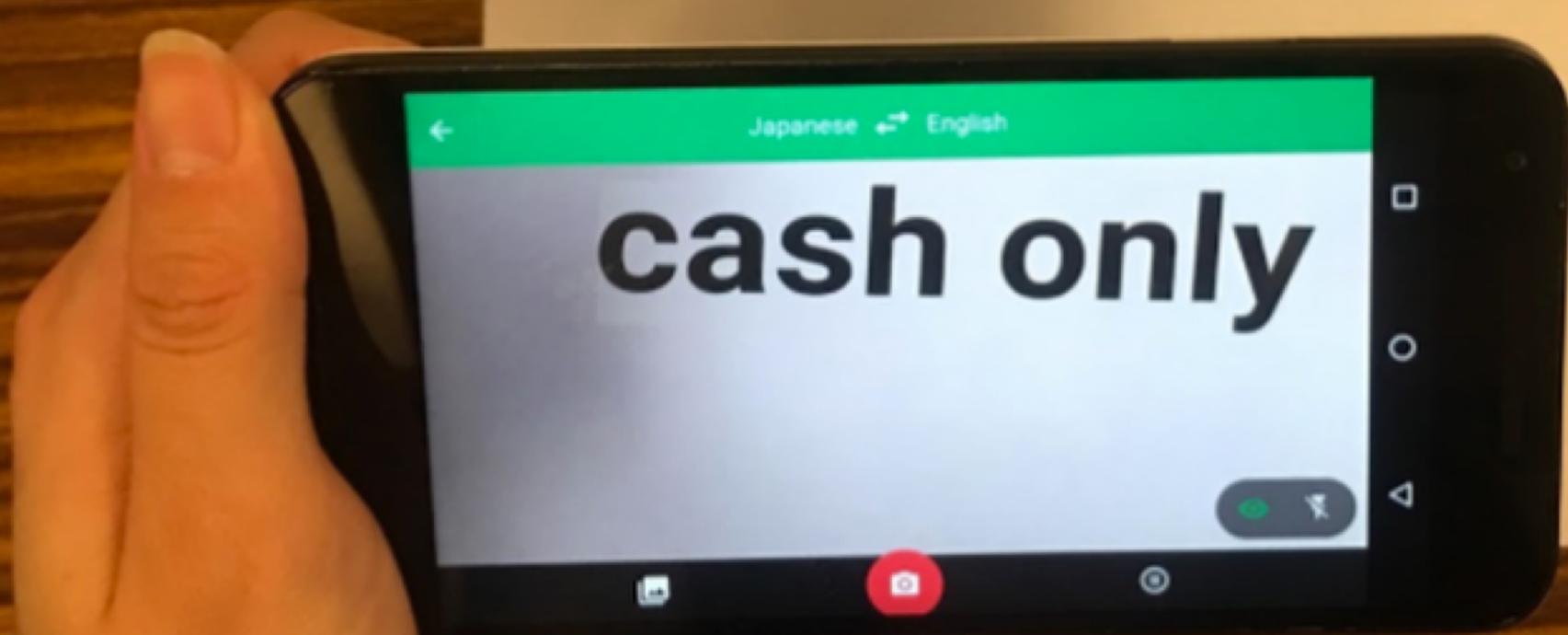
# Training Data

# Feature Engineering

- Identify parameters of interest that a model may learn on
- Convert data into a useful form
- Normalize data
- Include context
- Remove misleading things
- In OCR/translation:

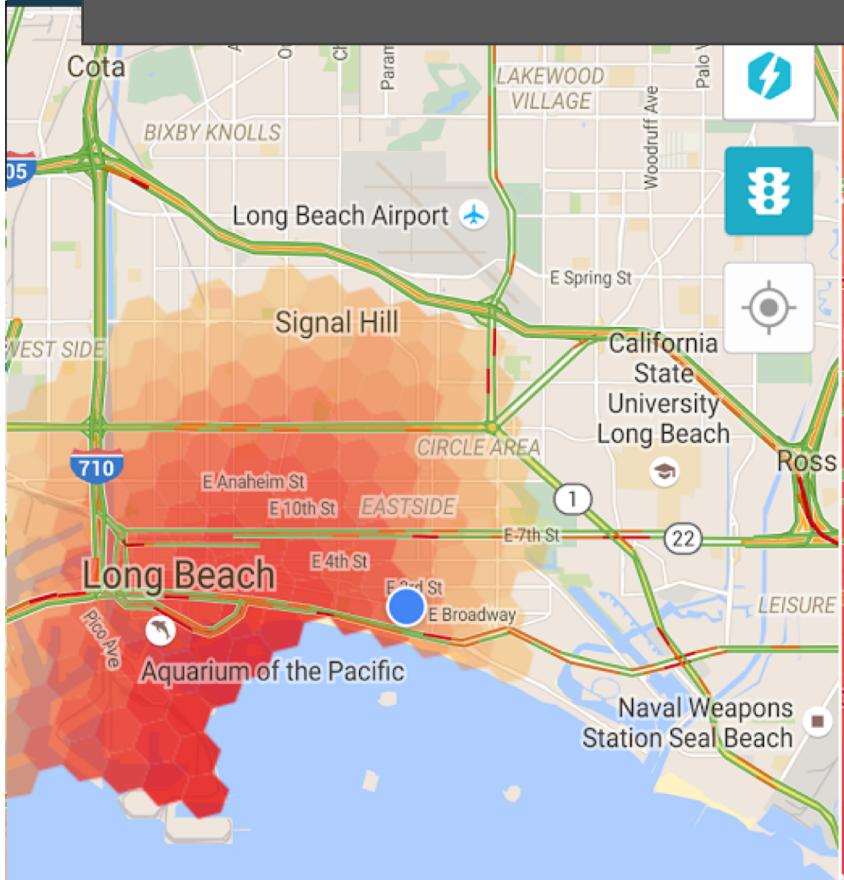
Features?

現金のみ



# Features?

GO OFFLINE



Google

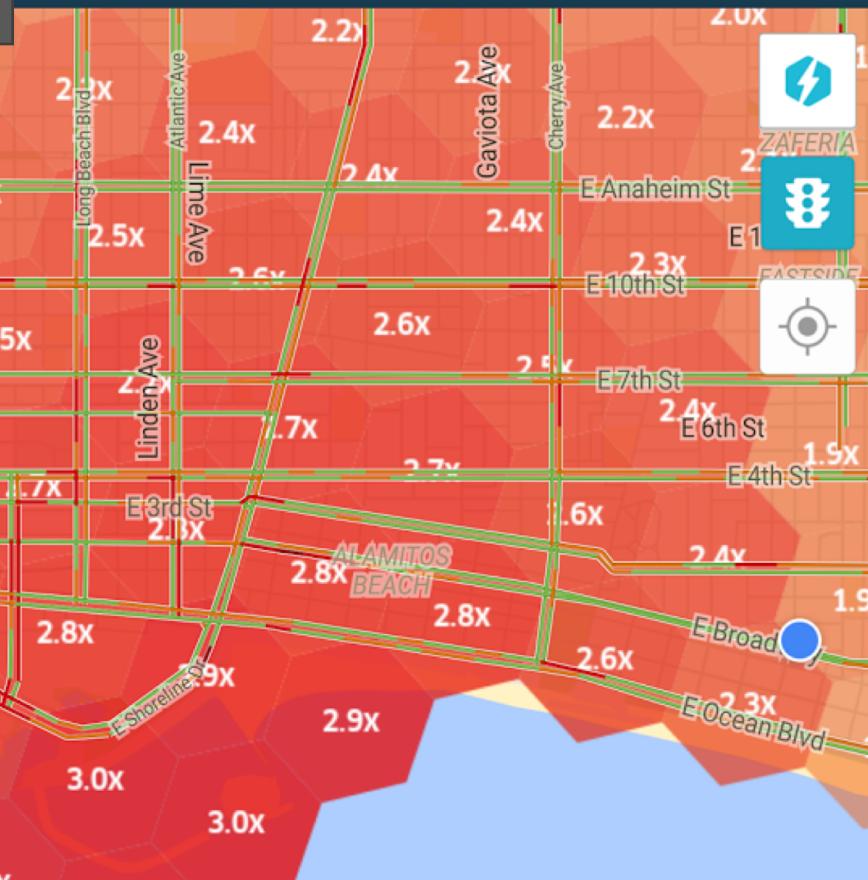
CURRENT PROMOTION

HOME

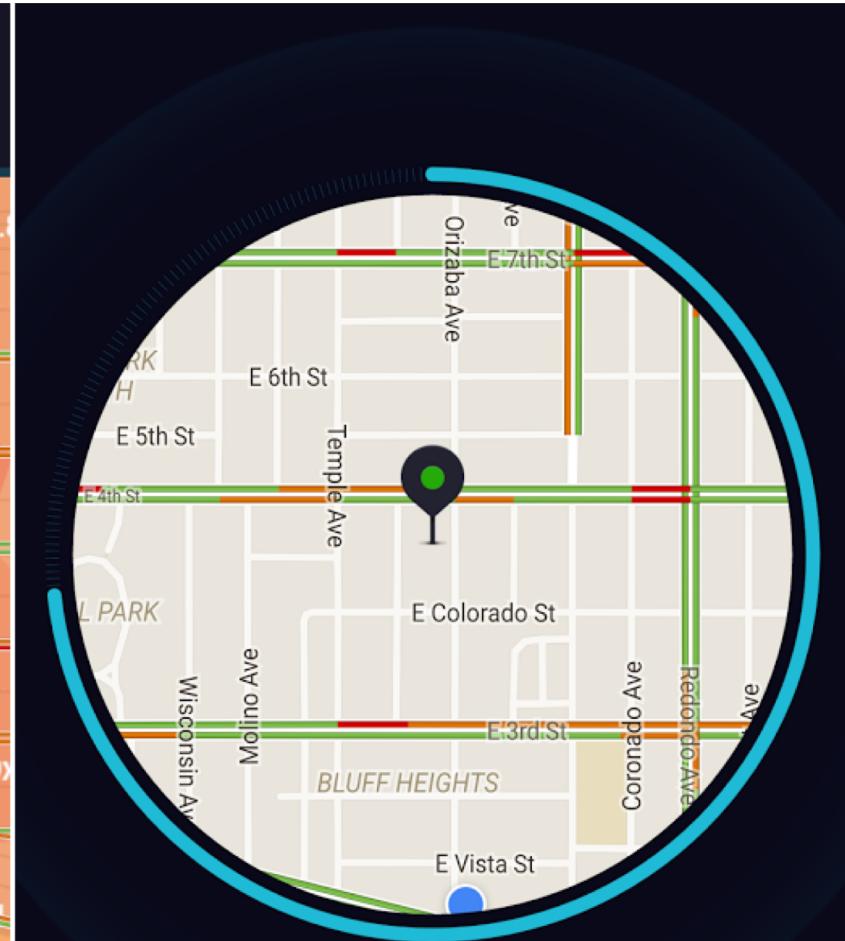
EARNINGS

RATINGS

ACCOUNT



CURRENT PROMOTION



4 MINUTES

[REDACTED] Ave, Long Beach, CA  
90814, USA

5.0★ | POOL | ⚡ 1.9X

# Feature Extraction

- In surge prediction:
  - Location and time of past surges
  - Events
  - Number of people traveling to an area
  - Typical demand curves in an area
  - Demand in other areas

# Data Cleaning

- Removing outliers
- Normalizing data
- Missing values
- ...

# Learning

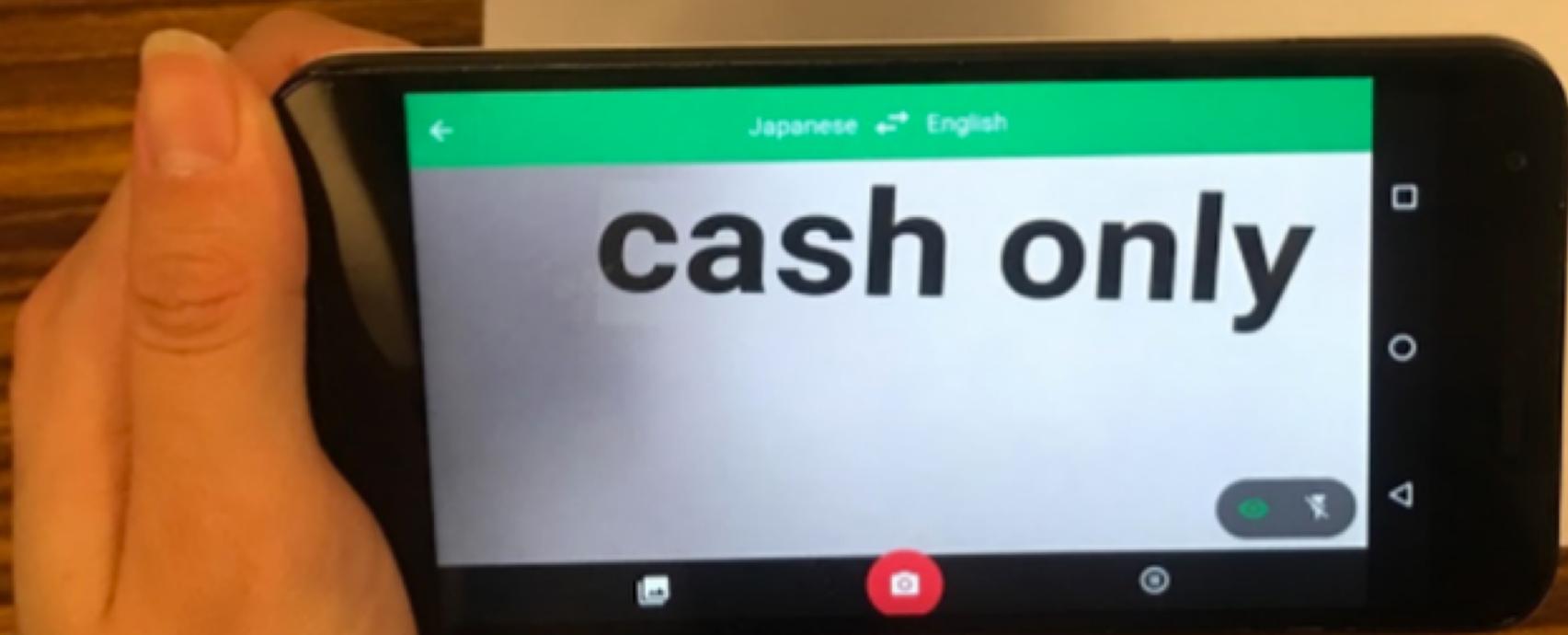
- Build a predictor that best describes an outcome for the observed features

# Evaluation

- Prediction accuracy on learned data vs
- Prediction accuracy on unseen data
  - Separate learning set, not used for training
- For binary predictors: false positives vs false negatives, recall, precision
- For numeric predictors: average (relative) distance between real and predicted value
- For ranking predictors: topK etc

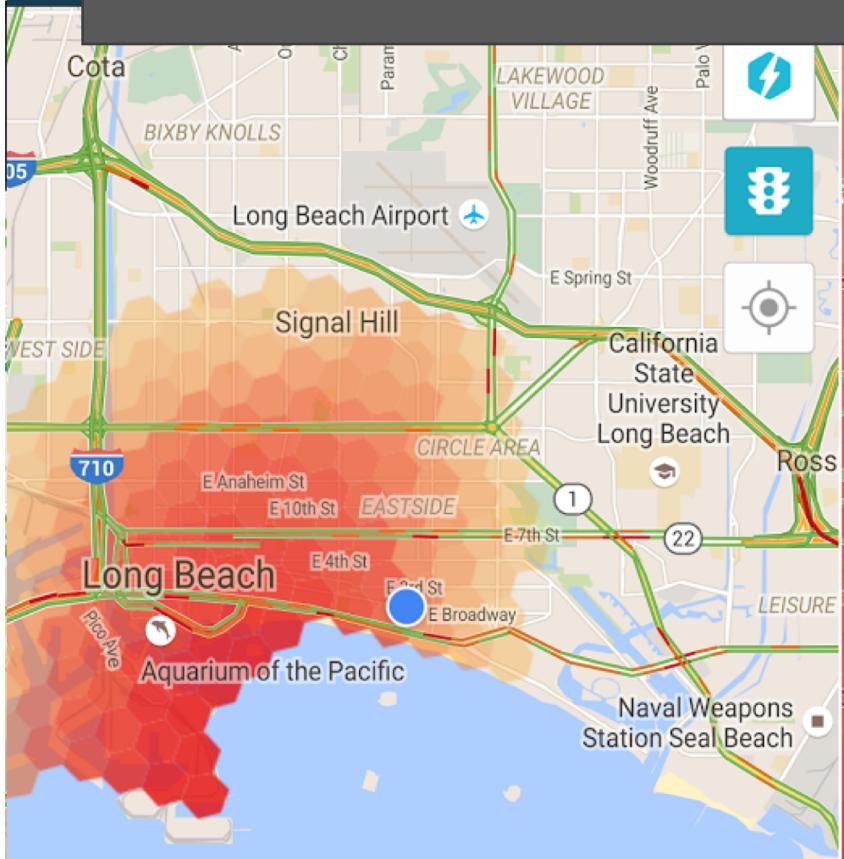
Evaluation Data?

現金のみ



# Evaluation Data?

GO OFFLINE



Google

CURRENT PROMOTION



HOME



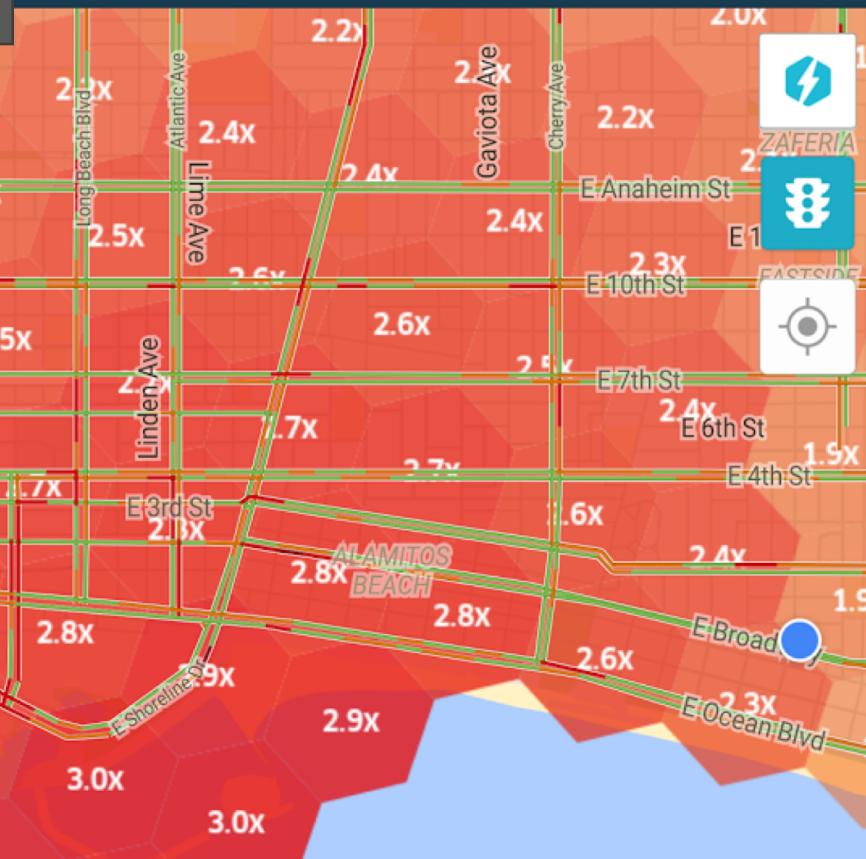
EARNINGS



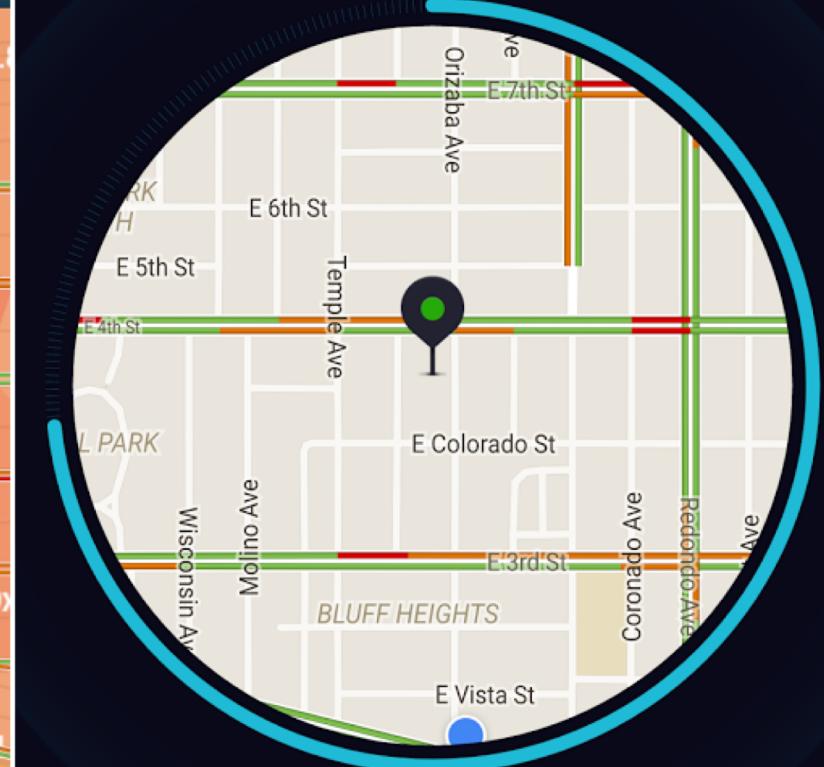
RATINGS



ACCOUNT



CURRENT PROMOTION



4 MINUTES



[REDACTED] Ave, Long Beach, CA  
90814, USA

5.0★ | POOL | ⚡ 1.9X

# Learning and Evaluating in Production

- Beyond static data sets, **build telemetry**
- Design challenge: identify mistakes in practice
- Use sample of live data for evaluation
- Retrain models with sampled live data regularly
- Monitor performance and intervene

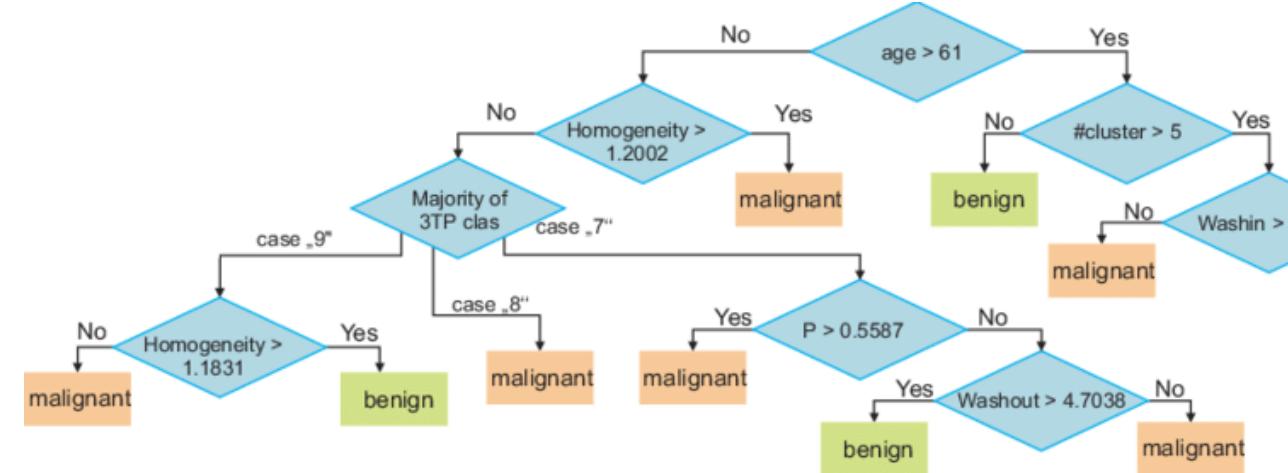
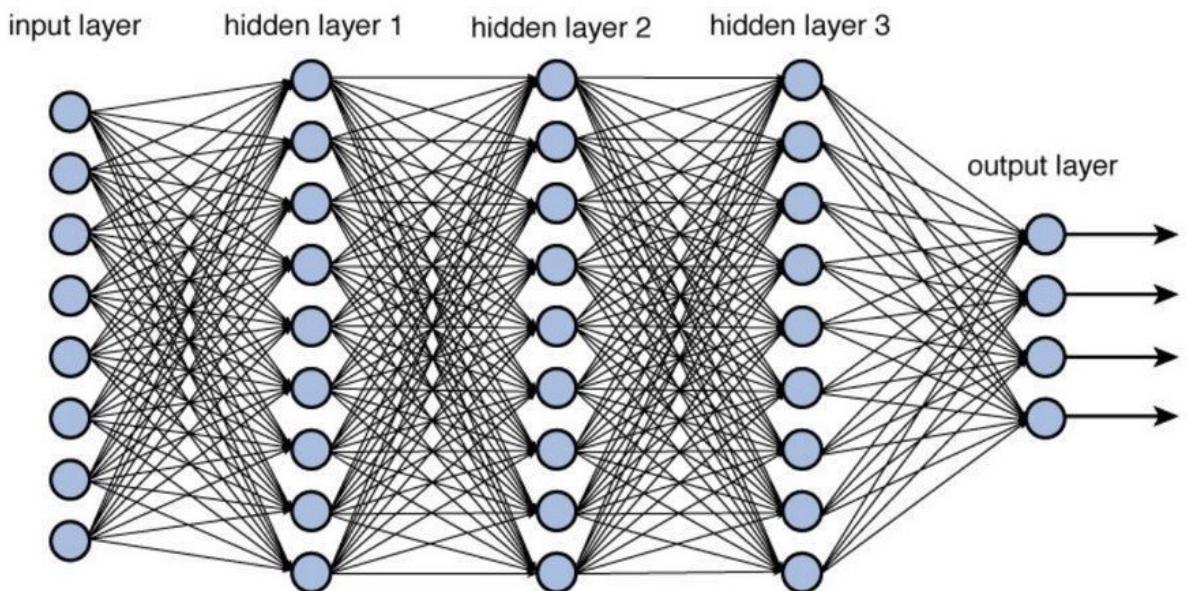
# ML COMPONENT TRADEOFFS

# Qualities of ML Components

- Accuracy
- Capabilities (e.g. classification, recommendation, clustering...)
- Amount of training data needed
- Inference latency
- Learning latency; incremental learning?
- Model size
- Explainable? Robust?
- ...

# Understanding Capabilities and Tradeoffs

- Deep Neural Networks
- Decision Trees



# **SYSTEM ARCHITECTURE CONSIDERATIONS**

# Where should the model live?

Glasses

Phone

Cloud

OCR  
Component

Translation  
Component

# Where should the model live?

Car

Phone

Cloud

Surge  
Prediction

# Considerations

- How much data is needed as input for the model?
- How much output data is produced by the model?
- 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?

# Typical Designs

- Static intelligence in the product
  - difficult to update
  - good execution latency
  - cheap operation
  - offline operation
  - no telemetry to evaluate and improve
- Client-side intelligence
  - updates costly/slow, out of sync problems
  - complexity in clients
  - offline operation, low execution latency

# Typical Designs

- Server-centric intelligence
  - latency in model execution (remote calls)
  - easy to update and experiment
  - operation cost
  - no offline operation
- Back-end cached intelligence
  - precomputed common results
  - fast execution, partial offline
  - saves bandwidth, complicated updates
- Hybrid models

# Other Considerations

- Coupling of ML pipeline parts
- Coupling with other parts of the system
- Ability for different developers and analysts to collaborate
- Support online experiments
- Ability to monitor

# Reactive System Design Goals

- Responsive
  - consistent, high performance
- Resilient
  - maintain responsive in the face of failure, recovery, rollback
- Elastic
  - scale with varying loads

# Common Design Strategies

- Message-driven, lazy computation, functional programming
  - asynchronous, message passing style
- Replication, containment, supervision
  - replicate and coordinate isolated components, e.g. with containers
- Data streams, “infinite data”, immutable facts
  - streaming technologies, data lakes
- See “big data systems” and “cloud computing”

# Making Decisions

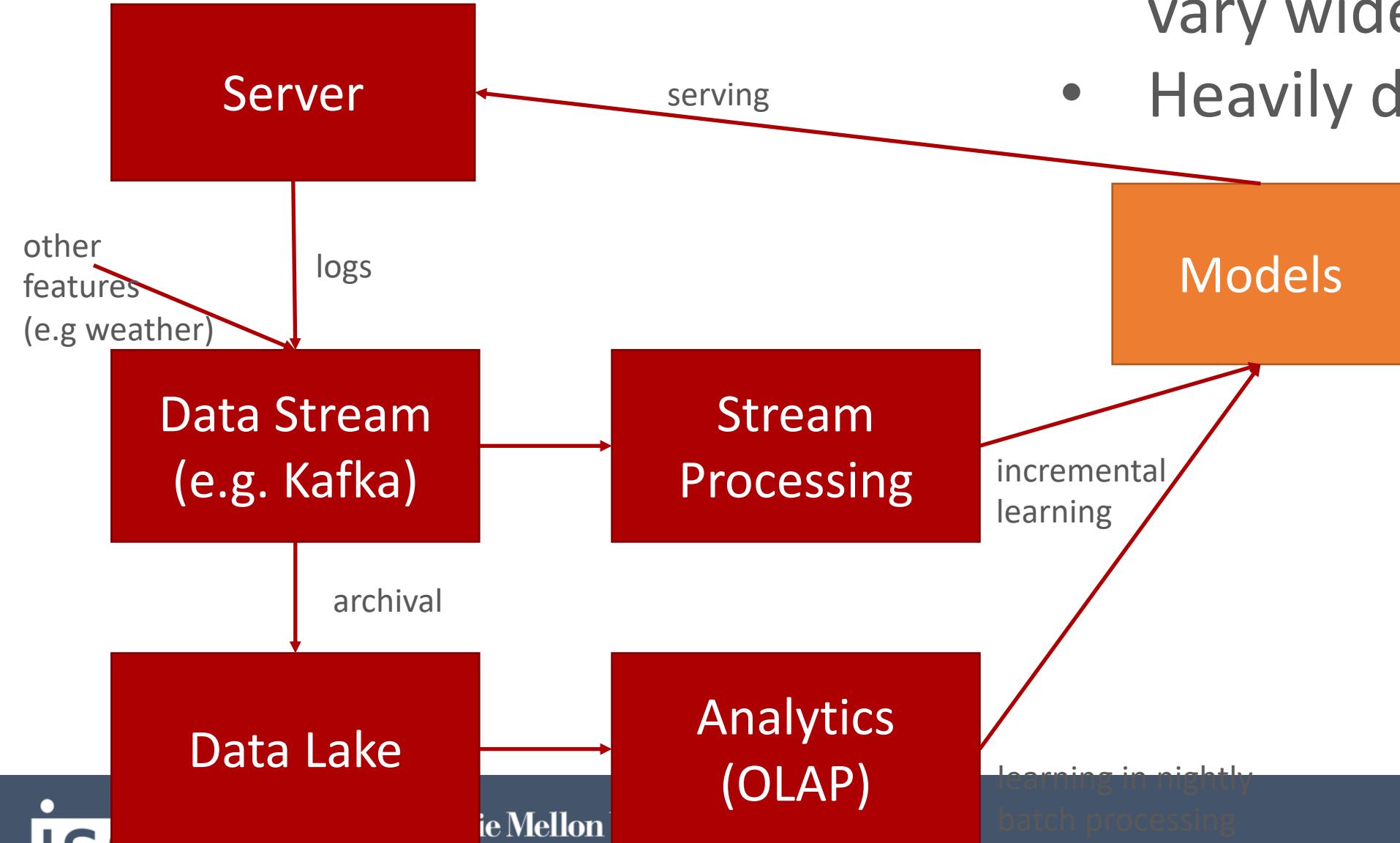
- What steps to take?
- What information to collect?

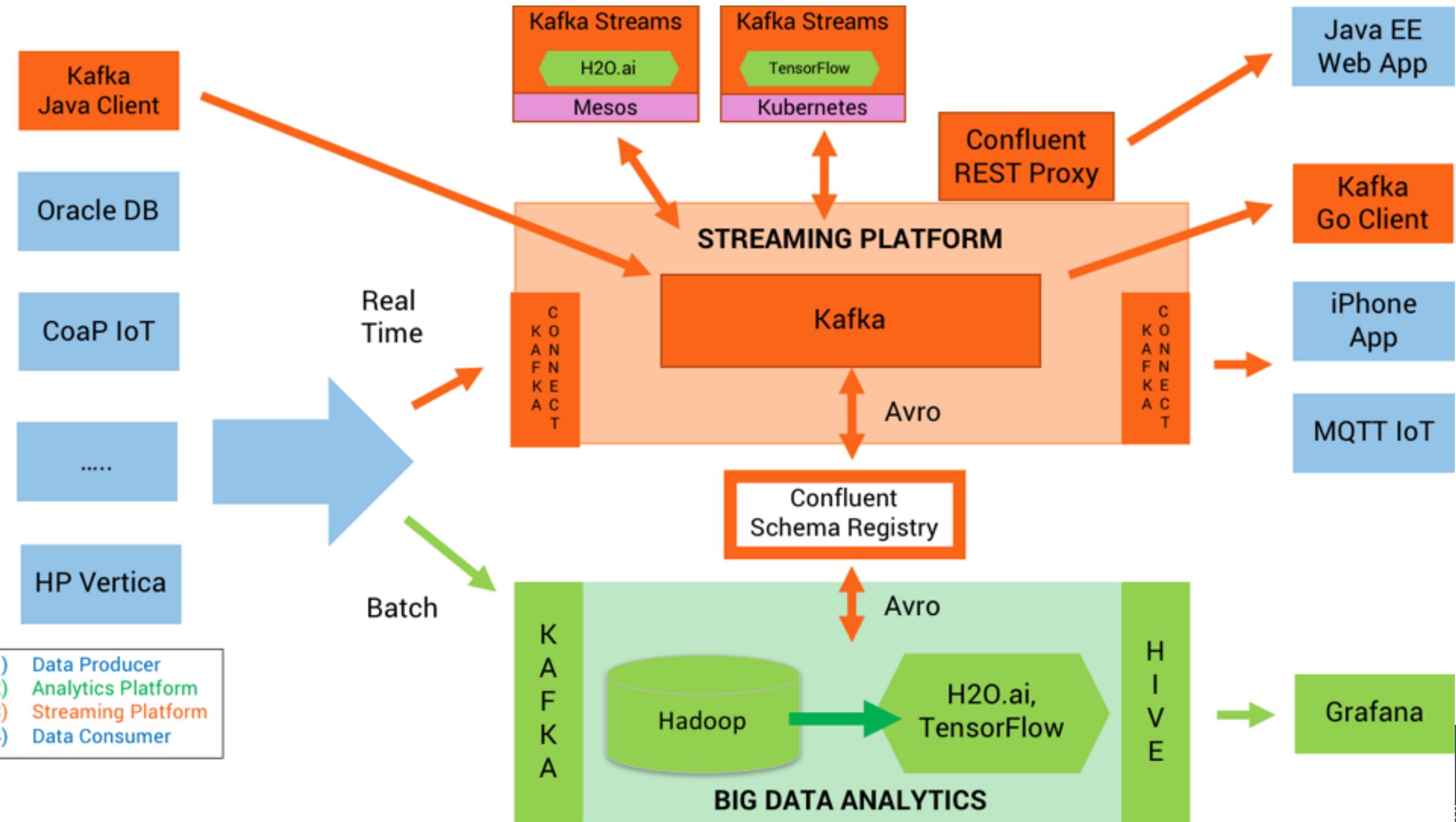
# UPDATING MODELS

# Updating Models

- Models are rarely static outside the lab
- Data drift, feedback loops, new features, new requirements
- When and how to update models?
- How to version? How to avoid mistakes?

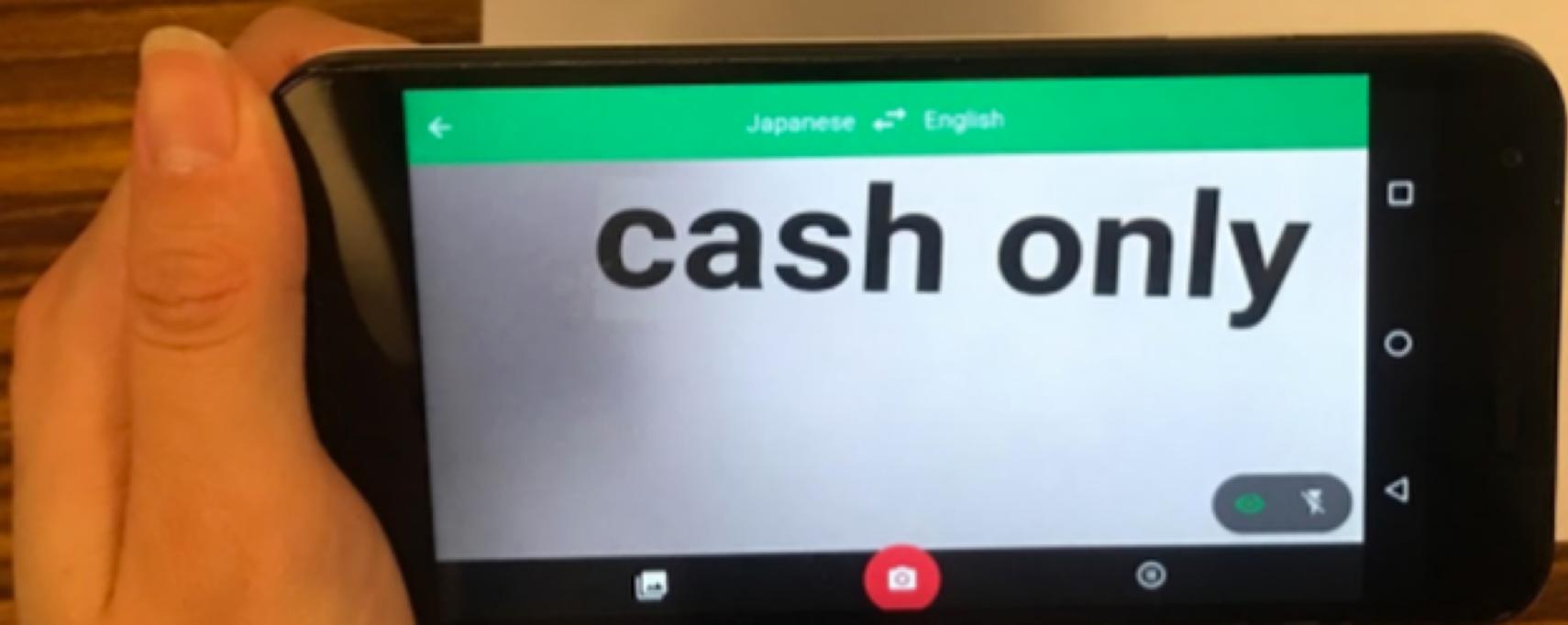
- Latency and automation vary widely
- Heavily distributed





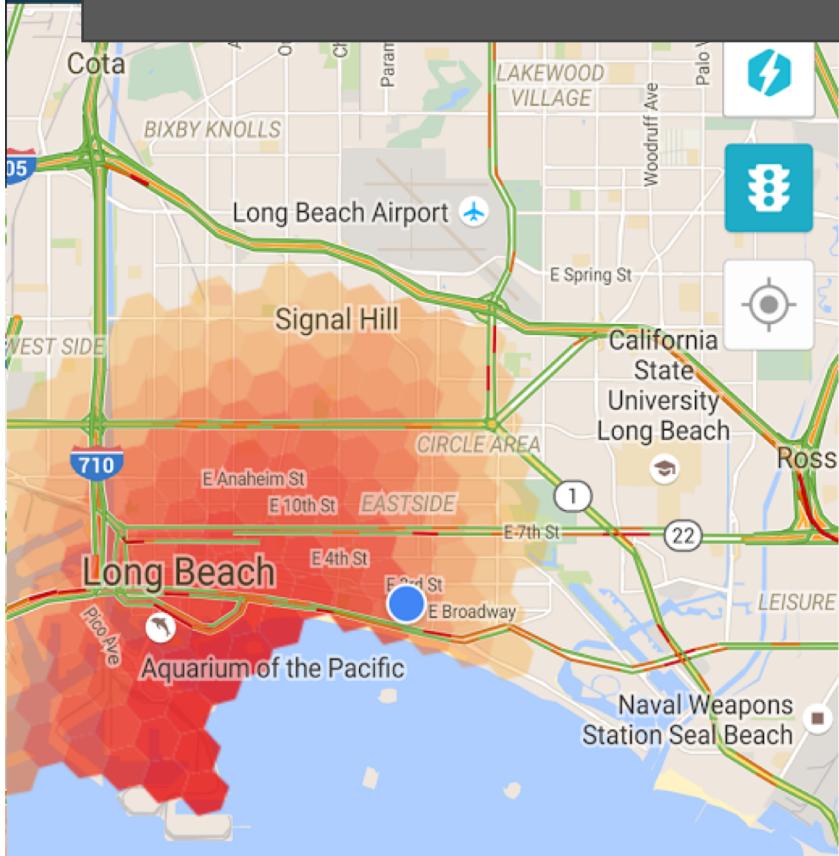
Update Strategy?

現金のみ



# Update Strategy?

GO OFFLINE



Google

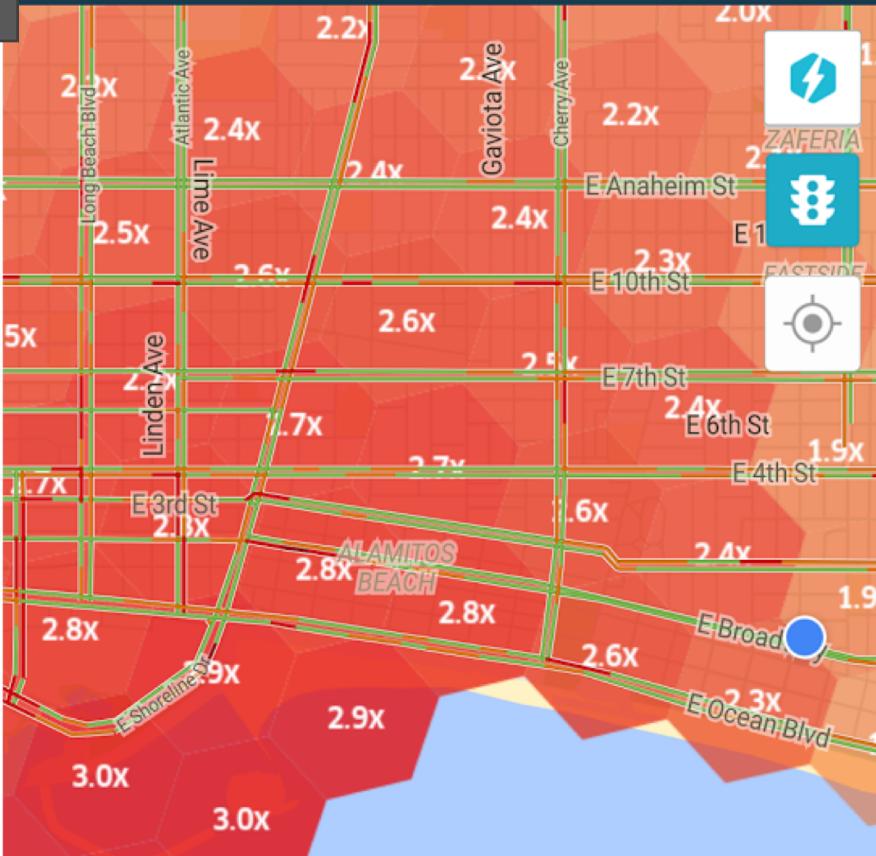
CURRENT PROMOTION

HOME

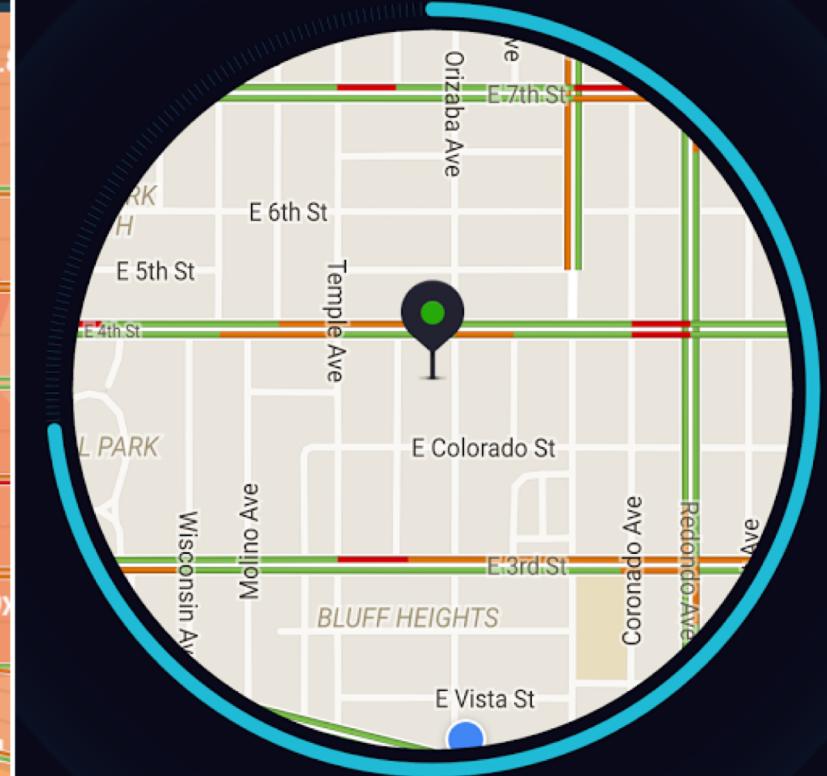
EARNINGS

RATINGS

ACCOUNT



CURRENT PROMOTION



4 MINUTES

[REDACTED] Ave, Long Beach, CA  
90814, USA

5.0★ | POOL | ⚡ 1.9X

# Making Decisions

- What steps to take?
- What information to collect?

# PLANNING FOR MISTAKES

# Mistakes will happen

- No specification
- ML components detect patterns from data (real and spurious)
- Predictions are often accurate, but mistakes always possible
- Mistakes are not predictable or explainable or similar to human mistakes
- Plan for mistakes
- Telemetry to learn about mistakes?

# How Models can Break

- System outage
- Model outage
  - model tested? deployment and updates reliable? file corrupt?
- Model errors
- Model degradation
  - data drift, feedback loops

# Hazard Analysis

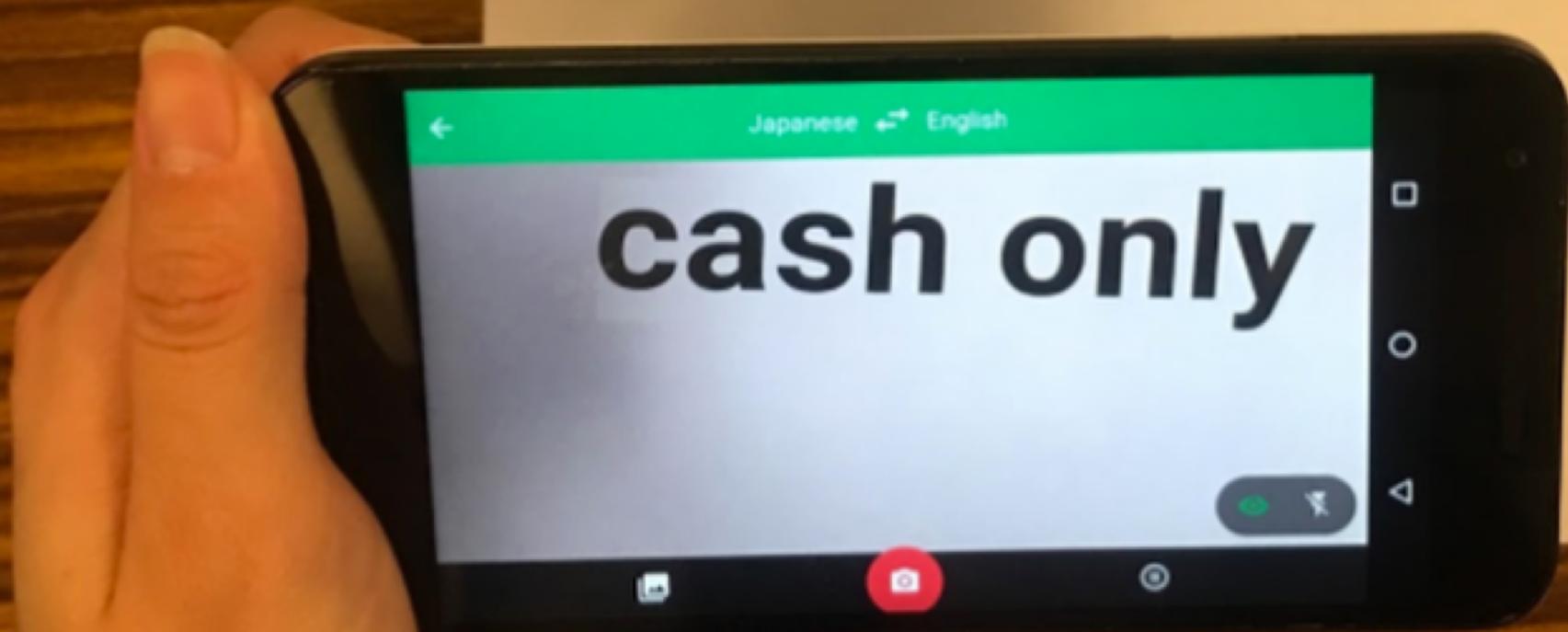
- Worst thing that can happen?
- Backup strategy? Undoable? Nontechnical compensation?

# Mitigating Mistakes

- Investigating in ML
  - e.g., more training data, better data, better features, better engineers
- Less forceful experience
  - e.g., prompt rather than automate decisions, turn off
- Adjust learning parameters
  - e.g., more frequent updates, manual adjustments
- Guardrails
  - e.g., heuristics and constraints on outputs
- Override errors
  - e.g., hardcode specific results

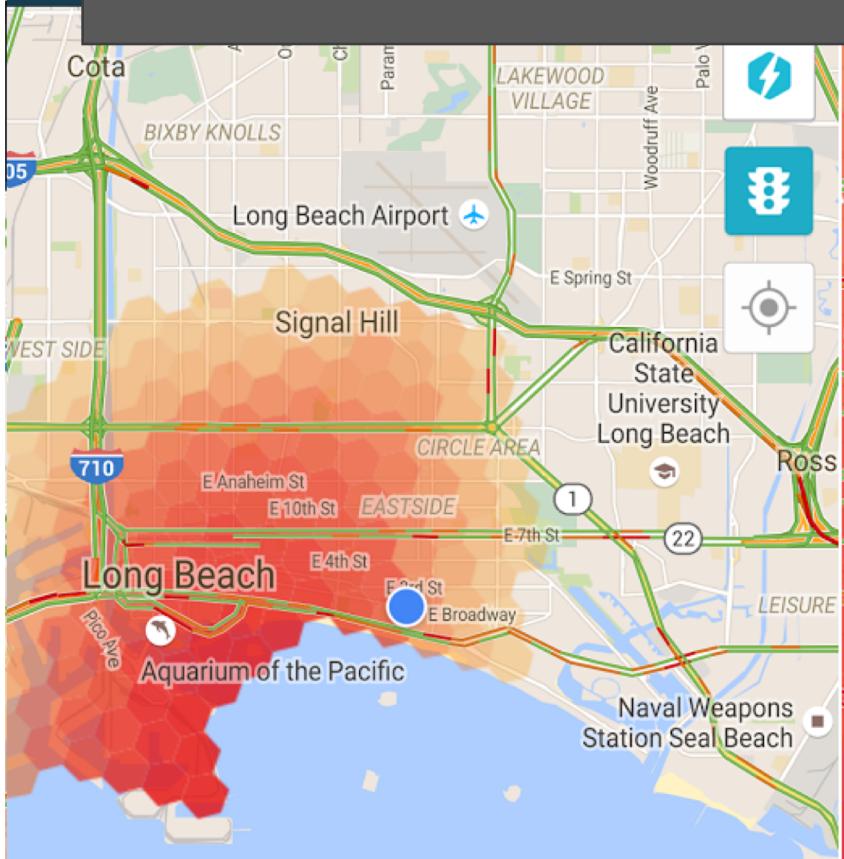
Mistakes?

現金のみ



# Mistakes?

GO OFFLINE



Google

CURRENT PROMOTION



HOME



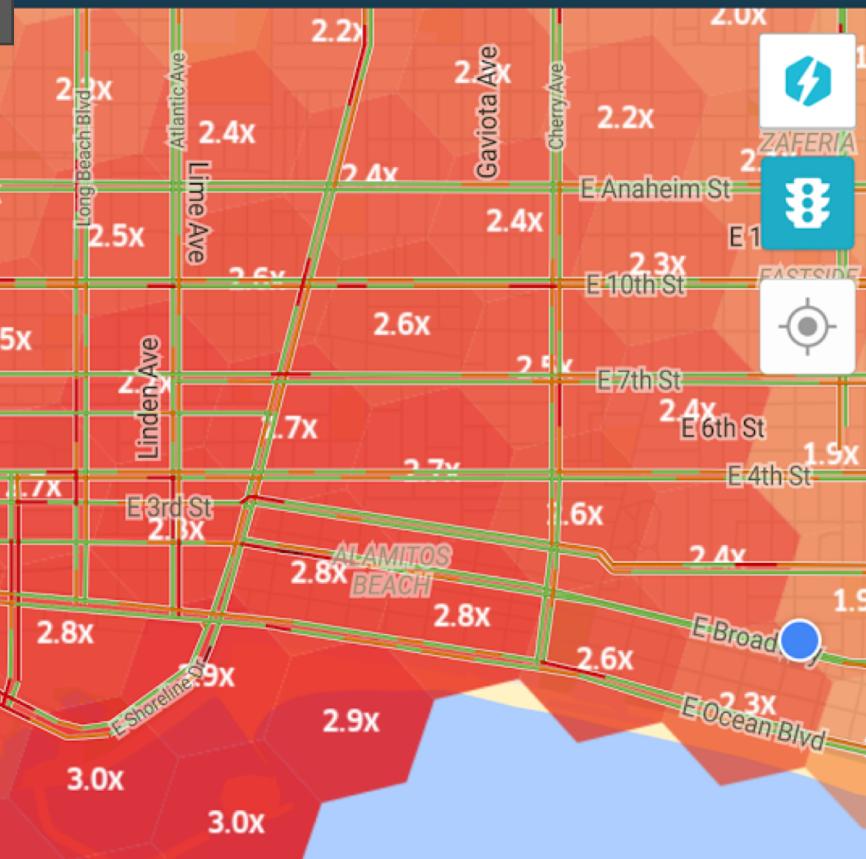
EARNINGS



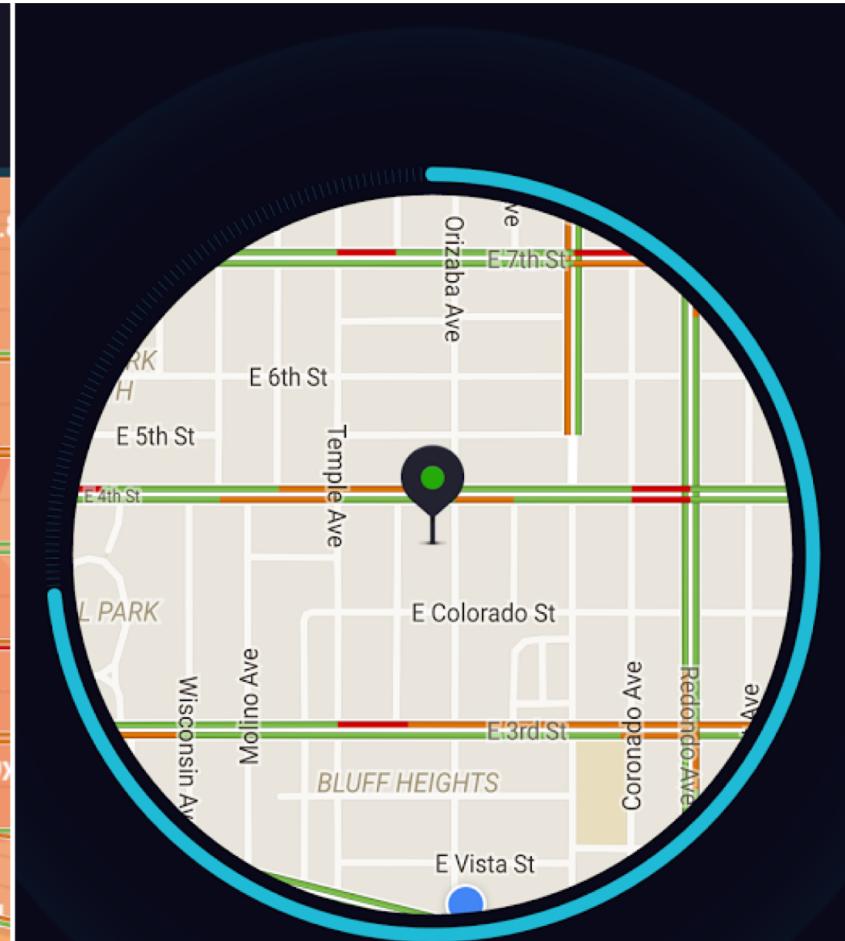
RATINGS



ACCOUNT



CURRENT PROMOTION



4 MINUTES



[REDACTED] Ave, Long Beach, CA  
90814, USA

5.0★ | POOL | ⚡ 1.9X

# Telemetry

- Purpose:
  - monitor operation
  - monitor success (accuracy)
  - improve models over time (e.g., detect new features)
- Challenges:
  - too much data – sample, summarization, adjustable
  - hard to measure – intended outcome not observable? proxies?
  - rare events – important but hard to capture
  - cost – significant investment must show benefit
  - privacy – abstracting data

# Model Orchestration

- Multiple models work together
  - ORC + statistical translation
  - driver activity predictor + demand predictor + event predictor + weather model
  - model + railguards
- Many different composition strategies
  - ensemble learning
  - integrated models
  - model sequencing
  - partitioning

# 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