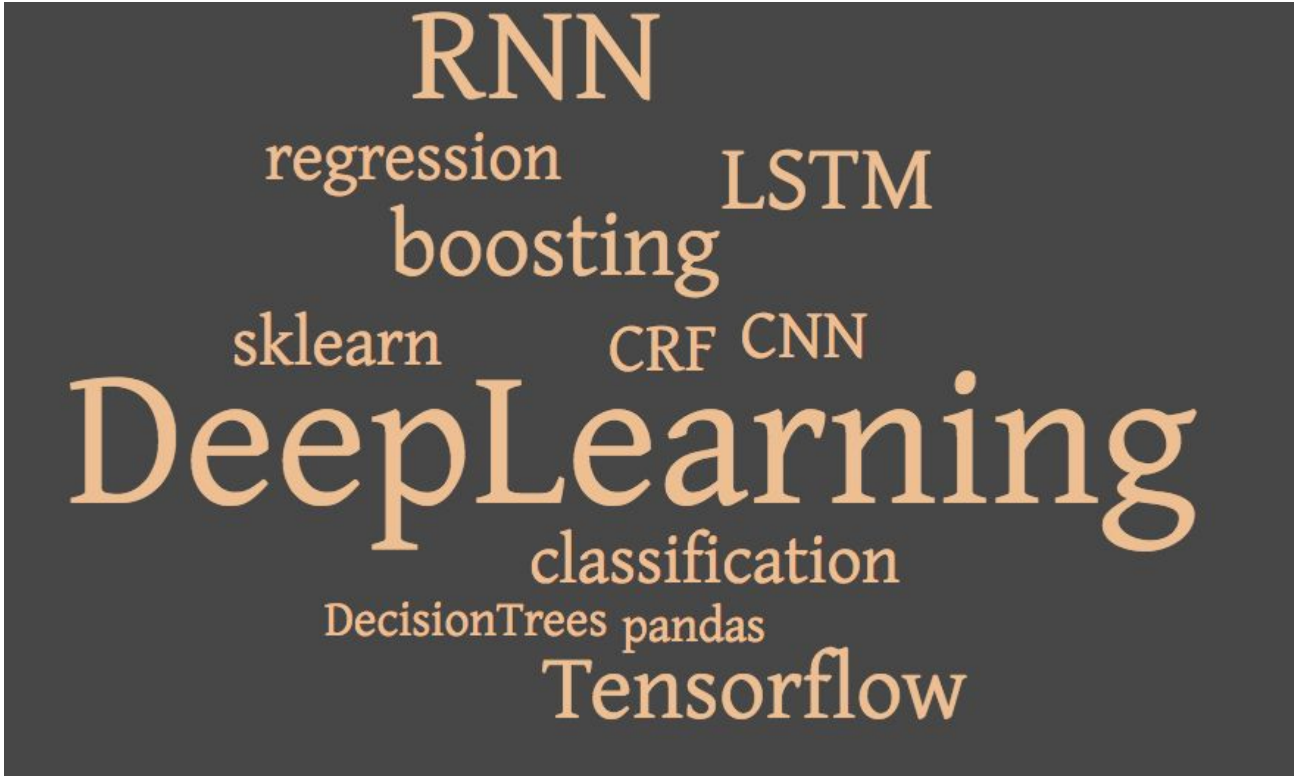
DOORDASH

Engineering Systems for Real-time Predictions at DoorDash

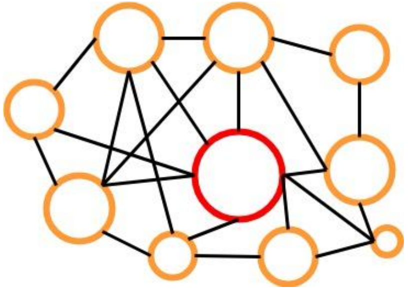
Raghav Ramesh
Machine Learning Engineer, DoorDash
June 2018

ML seems like

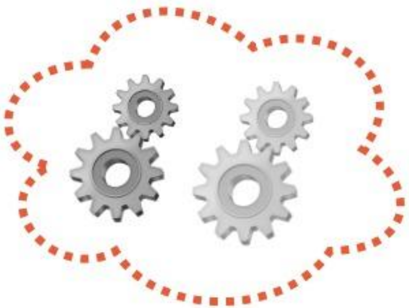


Practically, ML is

Data Pipelines



Model management



Performance monitoring

Real world ML = **10%** algorithms

\+

90% ML systems

ML Systems =

Robust tools

+

Templatizing best practices

ML Systems =

Robust tools

\+

Templatizing best practices

Concepts > Implementation

Agenda

DoorDash Overview

Evolution of ML

Systems for ML

Systems in action

Last mile,
on-demand logistics

Three-sided
marketplace

Restaurant Delivery

1600 cities by end
of 2018

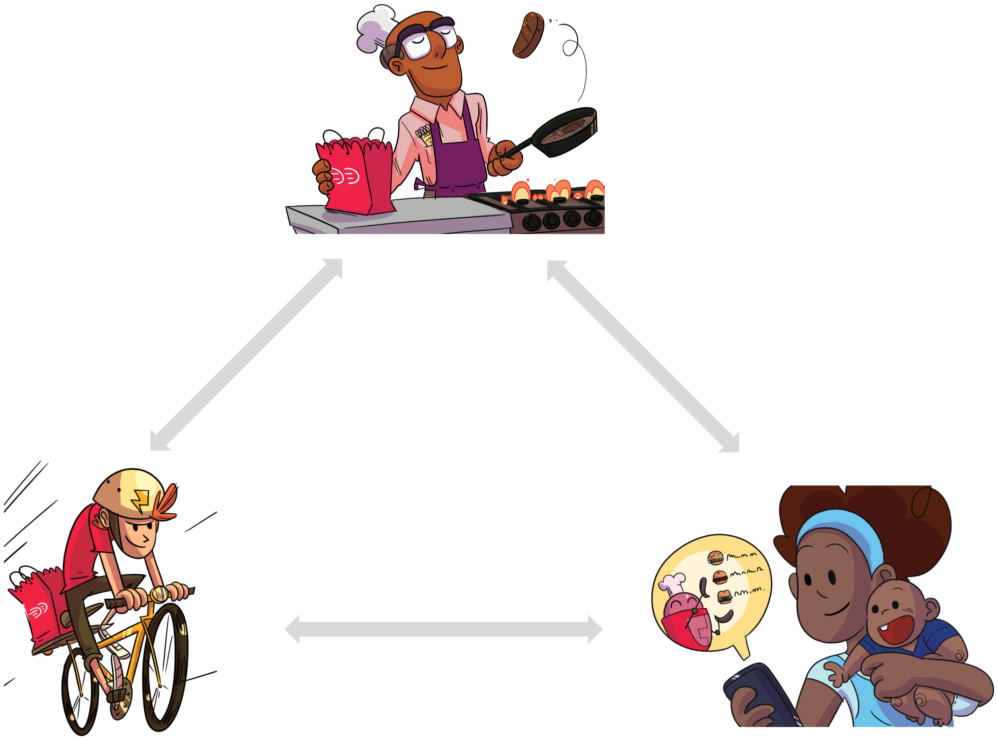
100,000+ Restaurants

300,000+ Dashers

10,000,000s of Deliveries

Marketplace

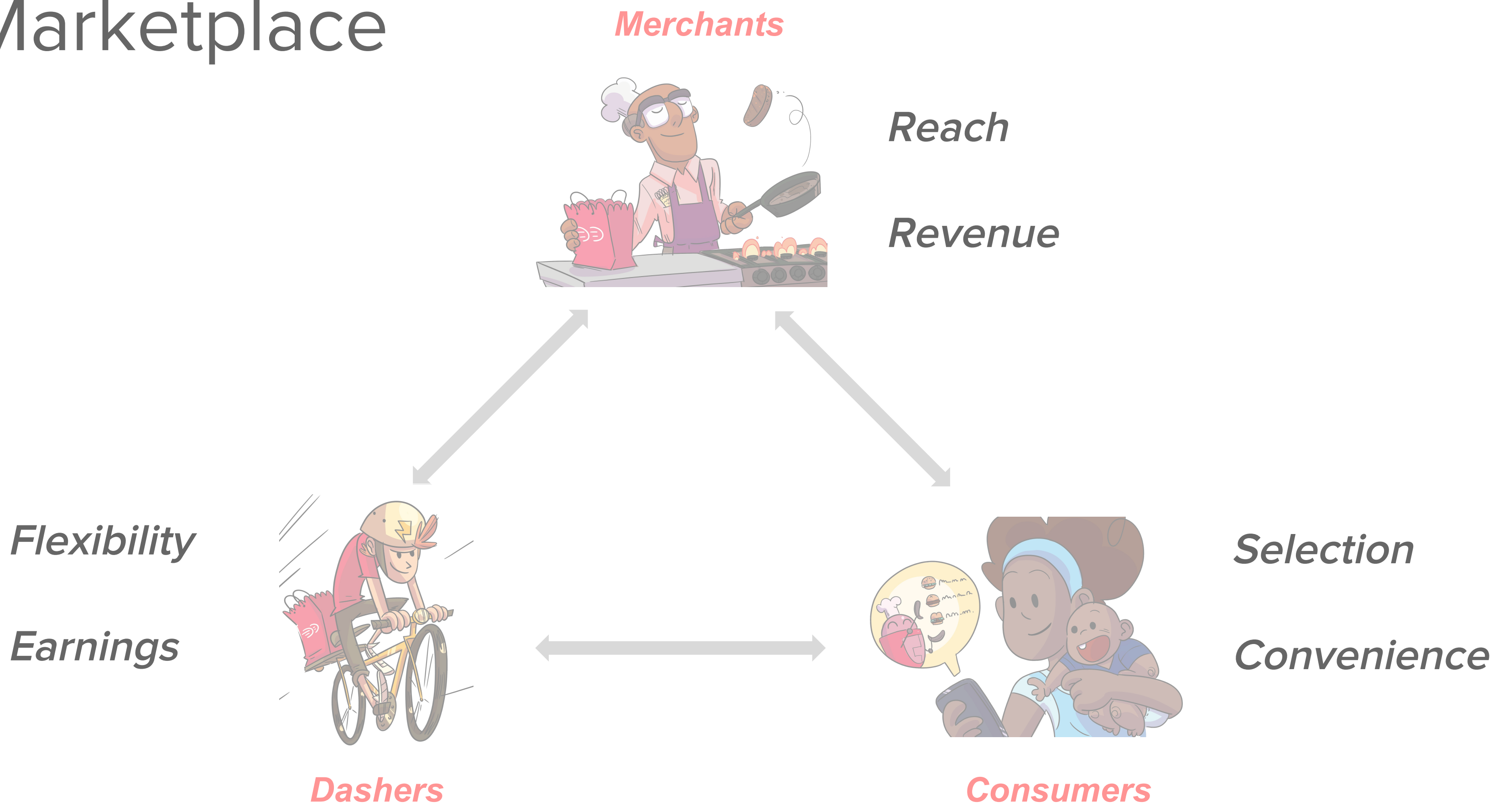
Merchants



Dashers

Consumers

Marketplace

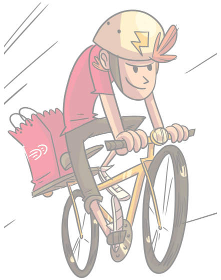


ML @ DoorDash

Merchants

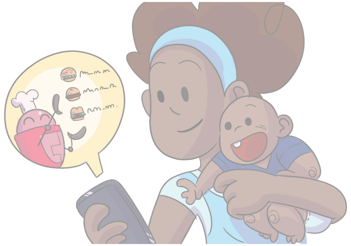
Core Dispatch
Batching algorithms
Hotspots

Recommendations / Personalization
Search ranking
Demand distribution



Dashers

Supply/Demand
Dynamic Pricing
Delivery Time



Consumers

ML @ DoorDash

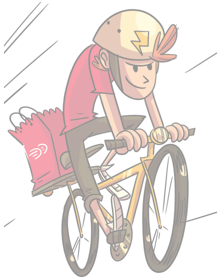
Merchants

Food prep time
Selection intelligence
Parking prediction

Core Dispatch
Batching algorithms
Hotspots

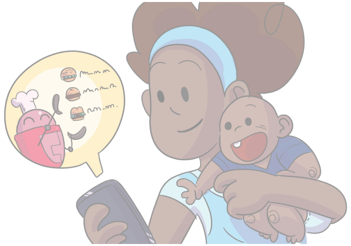
Recommendations / Personalization
Search ranking
Demand distribution

Pay calculation
Supply forecasting
Incentives



Dashers

Delivery Time Predictions
Supply/Demand Management
Dynamic Pricing



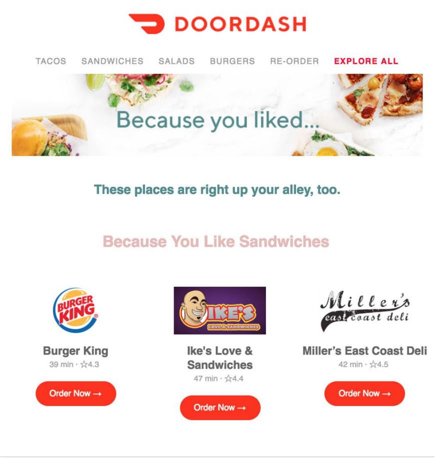
Consumers

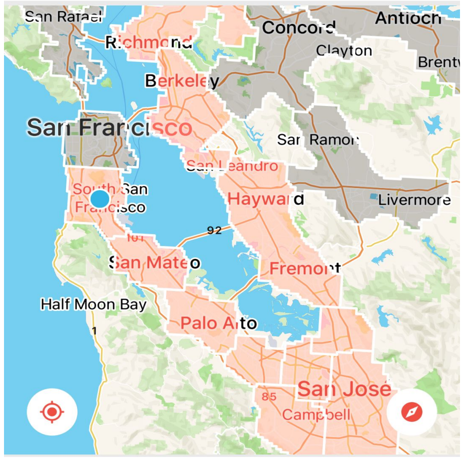
Lifetime value
Acquisition
Promotions

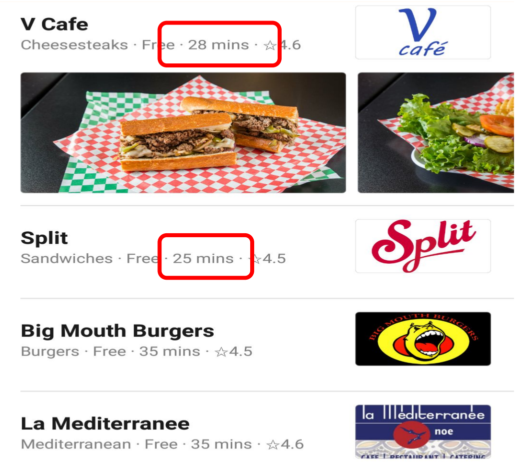
ML @ DoorDash

Offline

Online







Evolution of ML

How application of ML evolves in real world products

Evolution of ML

Evolution of ML

~~Sophisticated model~~ Simple Rules

Evolution of ML

~~Sophisticated model~~ Simple Rules

Manhattan → 40 mins

Austin → 35 mins

San Jose → 31 mins

Evolution of ML

~~Sophisticated model~~ Simple Rules

↓

Learnt coefficients

ETA =
25 minutes
+ **0.1** * supply_factor
+ **0.05** * order_value

Coefficients → Production monolith

Evolution of ML

~~Sophisticated model~~ Simple Rules

↓

Learnt coefficients

↓

First real ML model

Scikit-learn
Feature extractors

Evolution of ML

~~Sophisticated model~~ Simple Rules

↓

Learnt coefficients

↓

First real ML model

↓

Many models, each with its own stack

Different models for

ETA, Prep time,
Demand forecast, etc

Independent models
Duplicated code

Evolution of ML

~~Sophisticated model~~ Simple Rules

↓

Learnt coefficients

↓

First real ML model

↓

Many models, each with its own stack

↓

???

Evolution of ML

~~Sophisticated model~~ Simple Rules

↓

Learnt coefficients

↓

First real ML model

↓

Many models, each with its own stack

↓

???

As your product grows, performance of ML becomes mission critical

- Accuracy SLAs
- System SLAs
- Debuggability
- Time to ship

Evolution of ML

~~Sophisticated model~~ Simple Rules

Learnt coefficients

First real ML model

Many models, each with its own stack

???

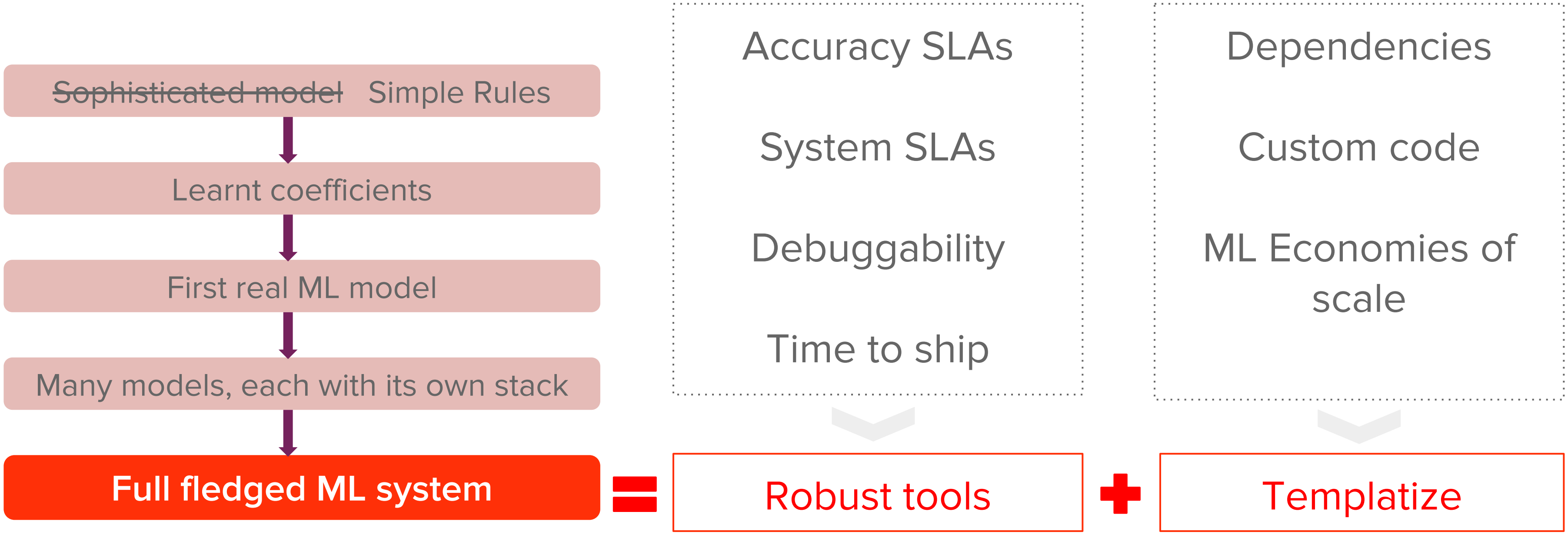
As your product grows, performance of ML becomes mission critical

- Accuracy SLAs
- System SLAs
- Debuggability
- Time to ship

… and as you build more ML

- Dependencies
- Custom code
- ML Economies of scale

Evolution of ML



ML Systems

Principles

Principles

Templatize best practices

Principles

Templatize best practices

Ease of use

Principles

Templatize best practices

Ease of use

Simple tools

Principles

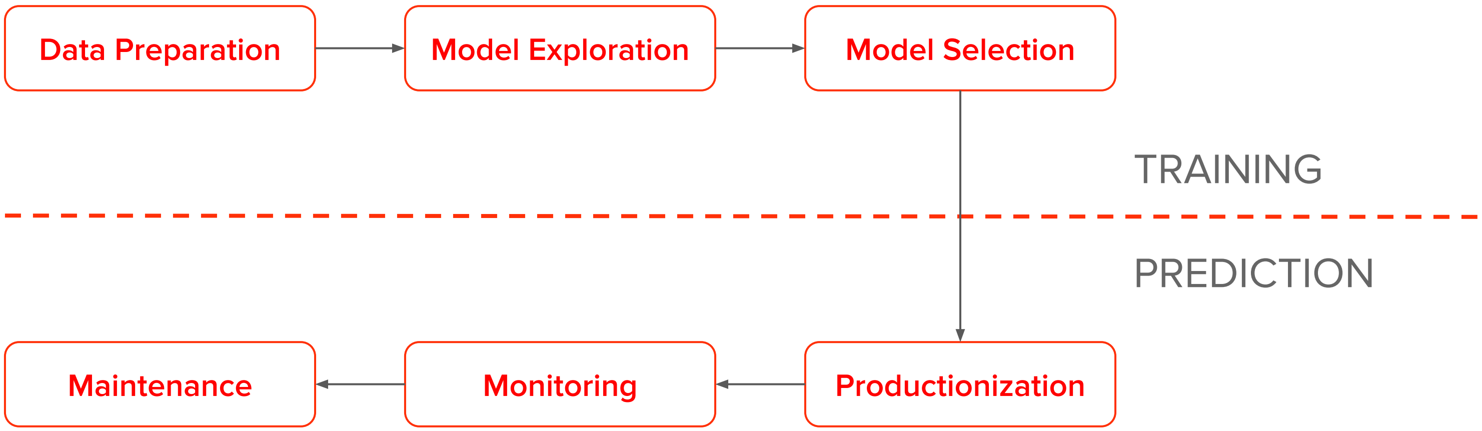
Templatize best practices

Ease of use

Simple tools

Integrate into ecosystem

ML Systems



Data Preparation
Model Exploration
Model Selection
TRAINING
PREDICTION
Maintenance
Monitoring
Productionization

ML Systems: Training

Data Preparation → Model Exploration → Model Selection

TRAINING

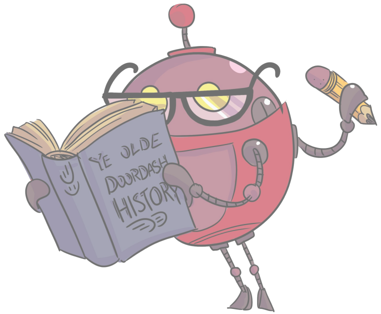
PREDICTION

Maintenance ← Monitoring ← Productionization

Offline

Model agnostic

Iteration speed



ML Systems: Prediction

Data Preparation → Model Exploration → Model Selection

TRAINING

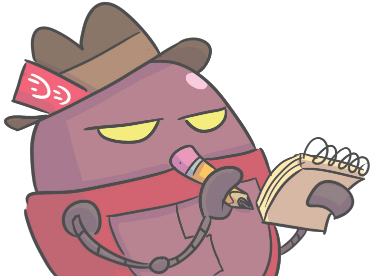
PREDICTION

Maintenance ← Monitoring ← Productionization

Real-time

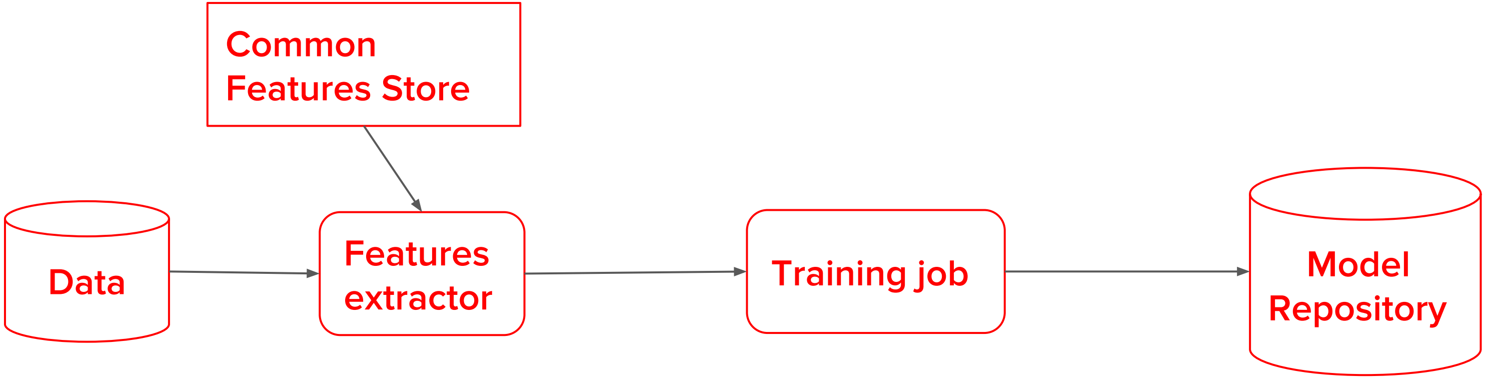
Production systems

Scale



Training

Training Pipeline



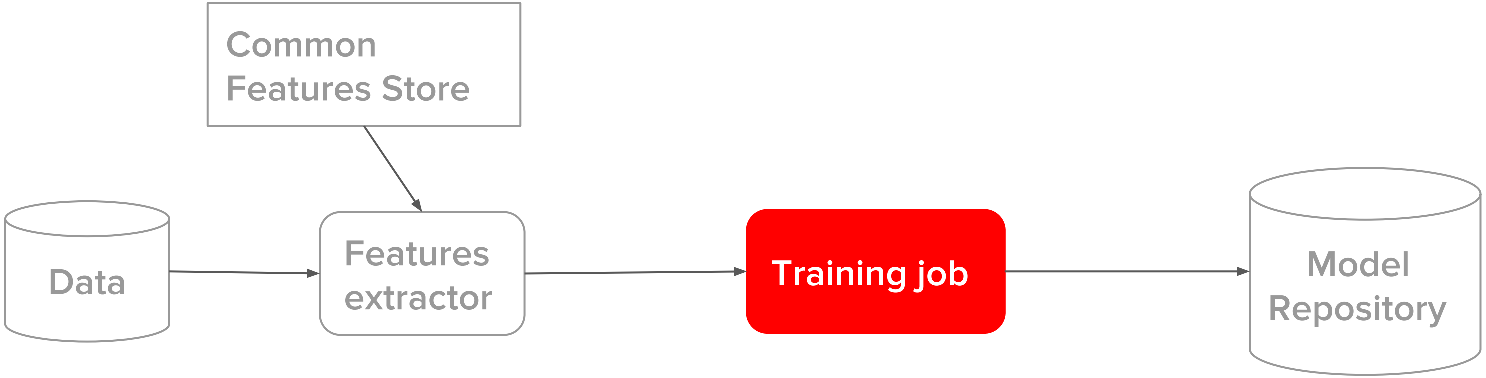
Training Pipeline

- ETL pipelines
- Job Scheduler

Training Pipeline

- Standardized
- Key → Value

Training Pipeline



- ML wrapper
- Airflow

Training Pipeline

- S3
- Metadata

Predictions

Predictions Pipeline

Predictions Pipeline

Predictions Request → **Predictions Service** → Predictions Response

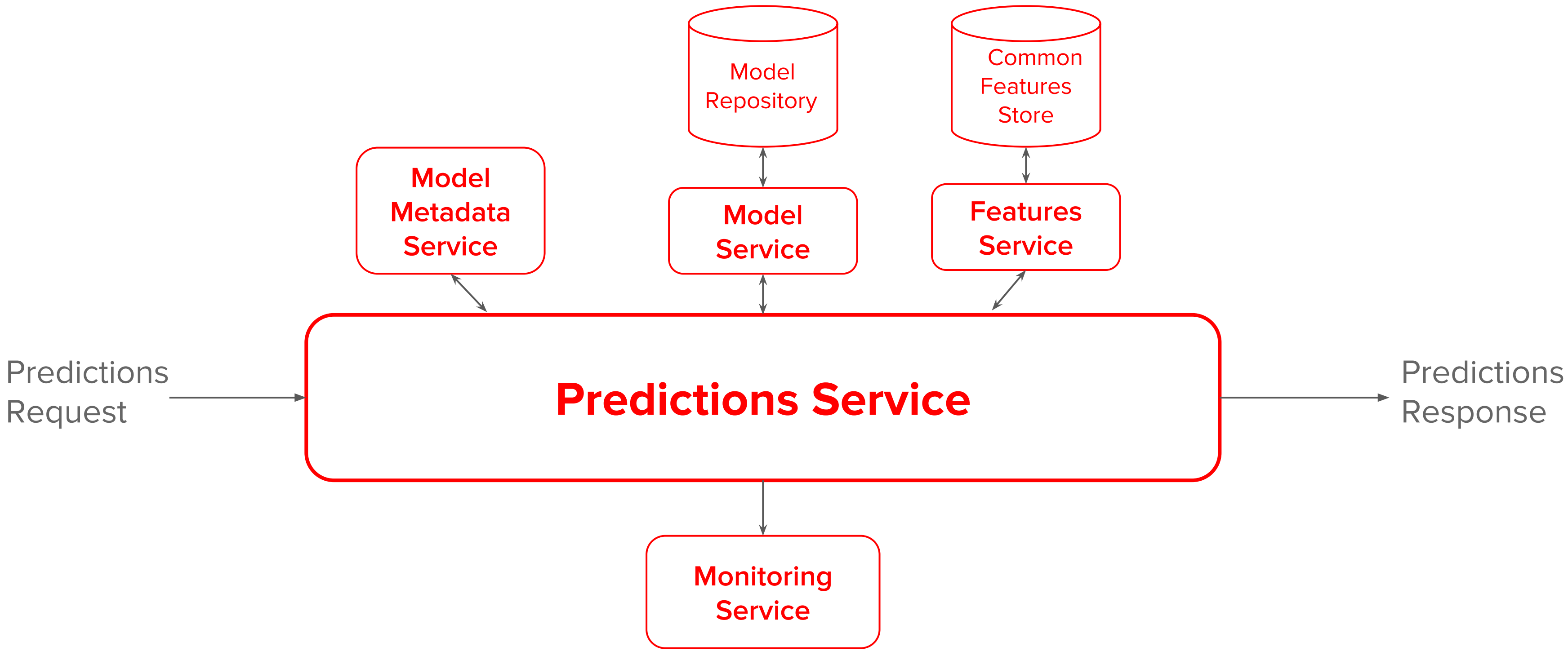
- Multiple predictors
- Low latency
- Load balancer
- HTTP

Predictions Pipeline

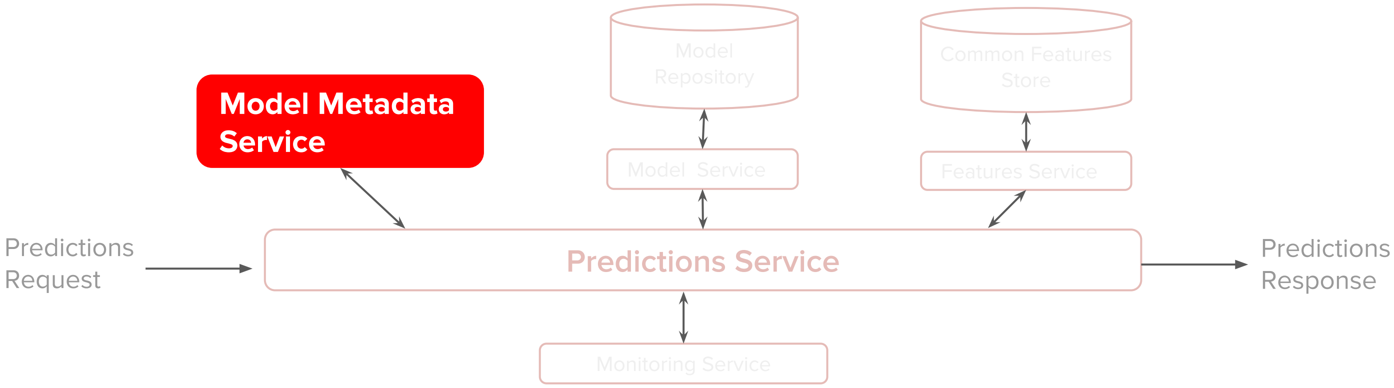
Predictions Request → **Predictions Service** → **Predictions Response**

- Multiple predictors
- Low latency
- Load balancer
- HTTP
- Kubernetes
- Flask

Predictions Pipeline



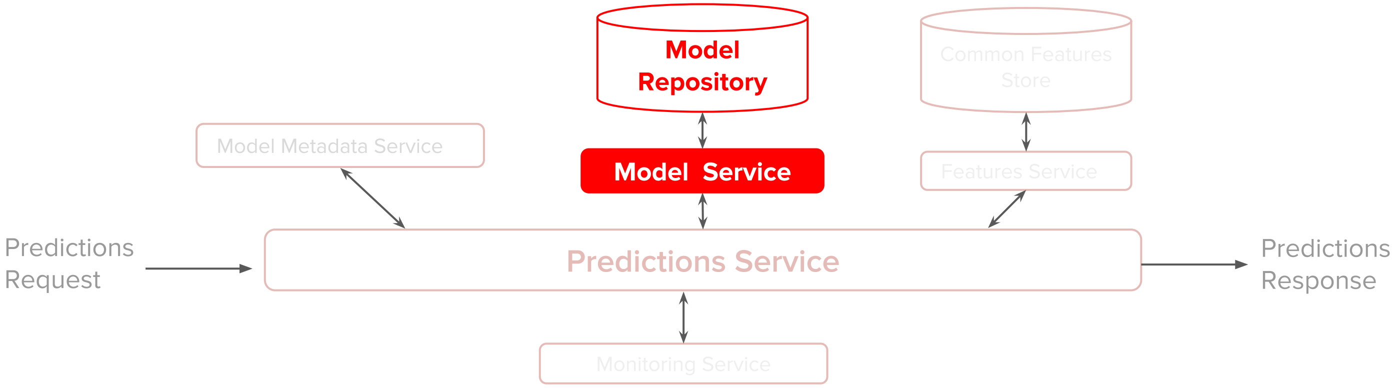
Predictions Pipeline



- **Command Center**
- **Model Status, Type, Version**
- **Registry and Retrieval**

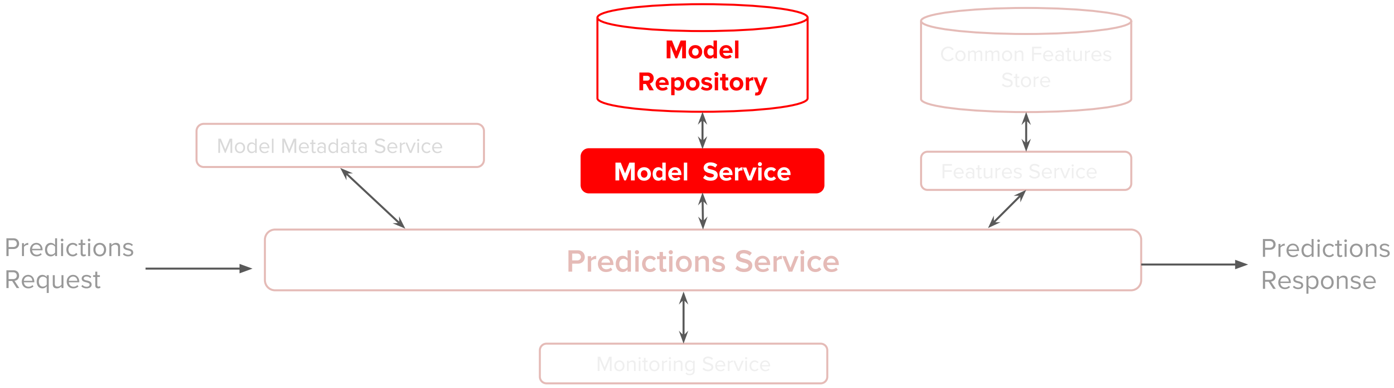
```json
{
    "id": "staging_prep_model_shadow_080402",
    "predictor": "prep_time",
    "is_active": true,
    "is_shadow": true,
    "is_ensemble": false,
    "s3_bucket_name": "doordash-staging-model-buckets",
    "s3_key_path": "prep_time/shadows/model_20180402",
    "predictor_type": "regressor",
    "metadata": { }
}
```

Predictions Pipeline



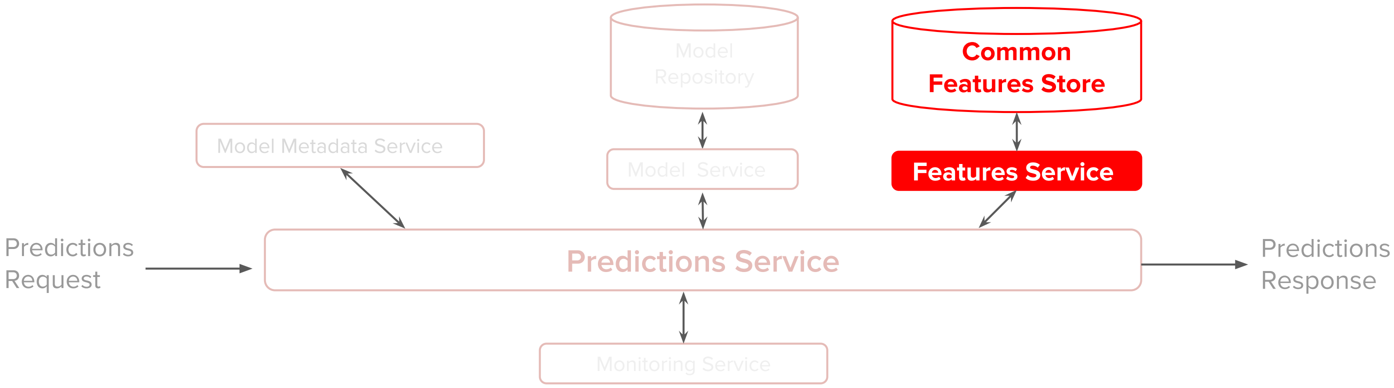
- Fetch models
- Caching layer
- Rollout Control
    - Shadowing
    - Experimentation

Predictions Pipeline



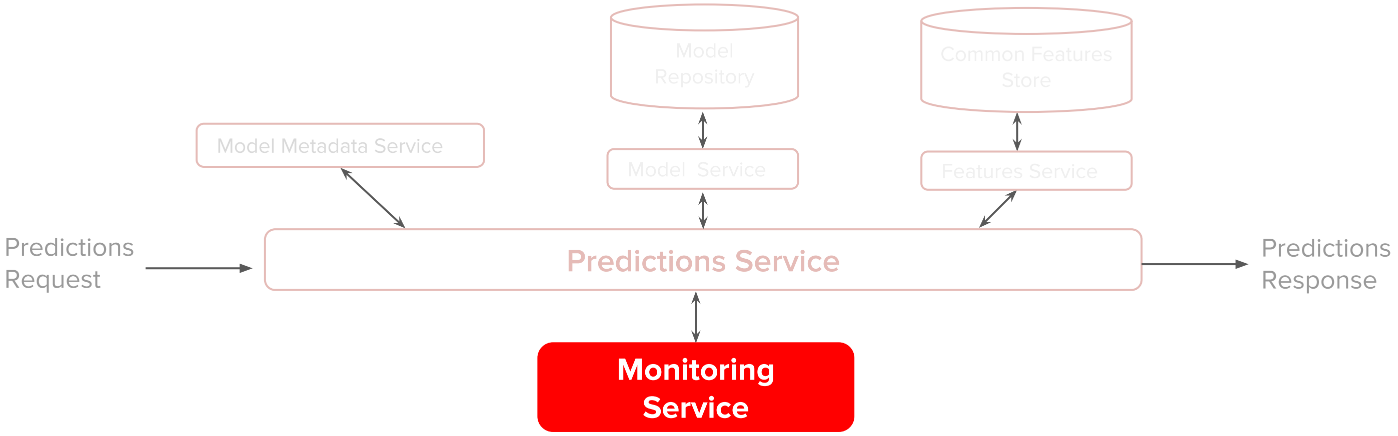
- Fetch models
- Caching layer
- Rollout Control
  - Shadowing
  - Experimentation
- Multiple model types
- Ensembling
- Segmentation

Predictions Pipeline



- Features Retrieval
- Source of truth
- Training / Prediction Consistency
- Batch aggregates
- Embeddings

Predictions Pipeline



- Predictions Monitoring
- Features Distribution
- Logging
- Statsd
- Segment

Systems in action

Features integrity

- Consistency across environments
- Consistency over time

Features integrity

- **Consistency across environments**

  No differences between training and prediction

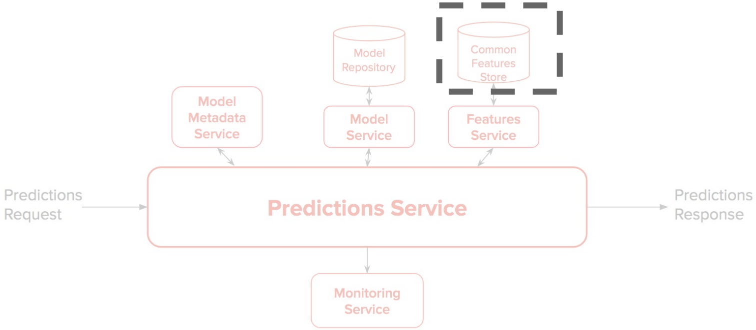
Features integrity

Consistency across environments

No differences between training and prediction

Common Features Store

- Used for both training and prediction
- Exact code
- Works for most features



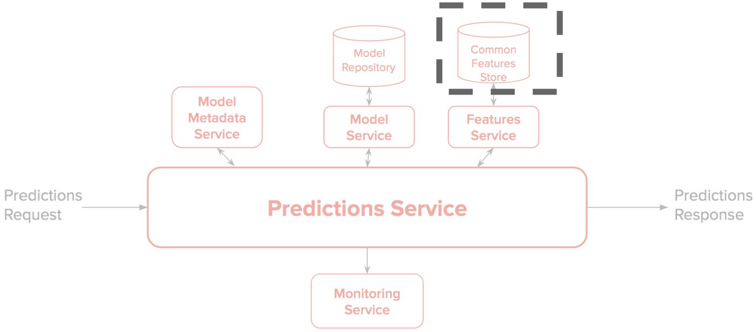
Features integrity

Consistency across environments

No differences between training and prediction

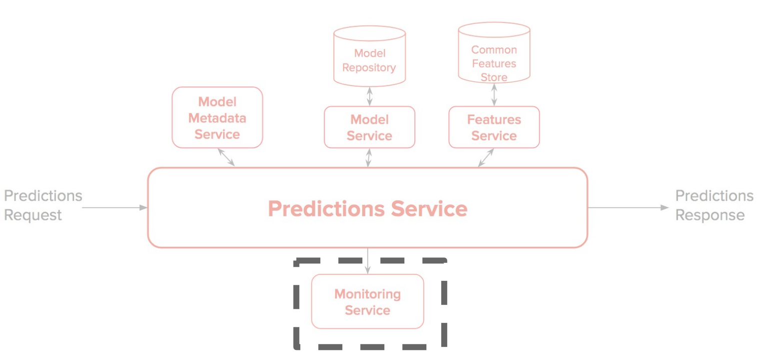
Common Features Store

- Used for both training and prediction
- Exact code
- Works for most features



Feature Logging on Prediction

- Log X%
- Re-train on that data
- Works for non new features



Features integrity

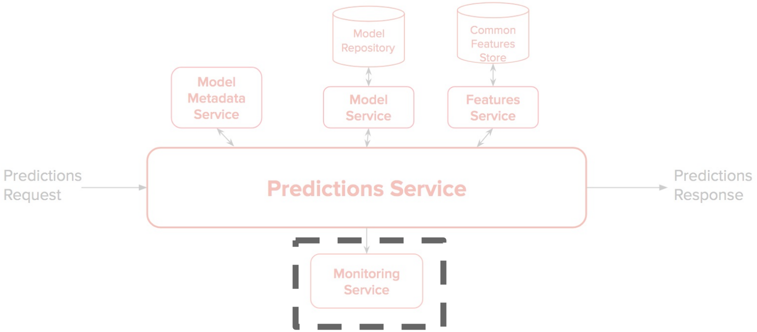
- Consistency across environments
- **Consistency over time**
  - No data degradations

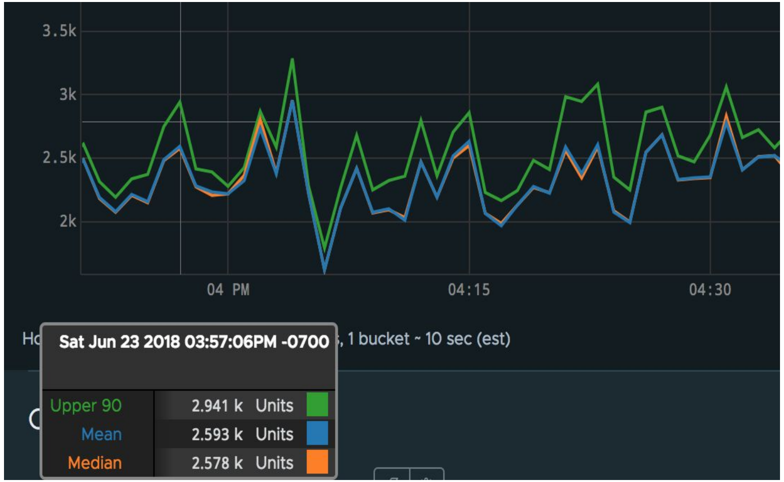
Features integrity

- Consistency across environments
- **Consistency over time**
  No data degradations

Features Monitoring

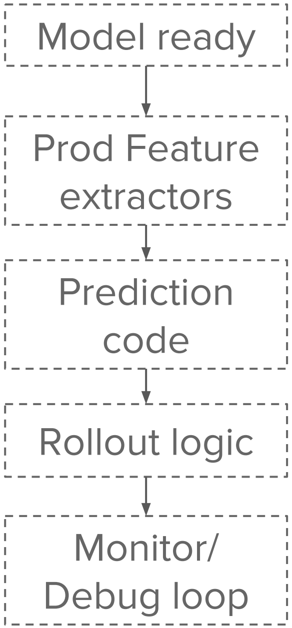
- Plot Distributions
- Alerting





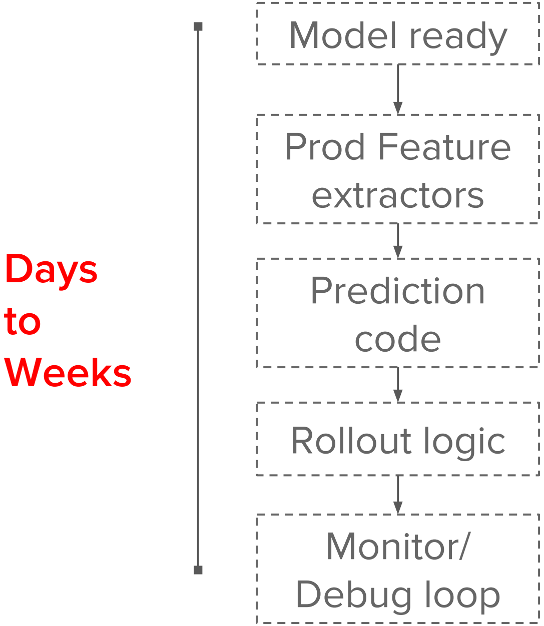
Launching new models

Previously



Launching new models

Previously



Launching new models

Previously

Days to Weeks

Model ready → Prod Feature extractors → Prediction code → Rollout logic → Monitor/ Debug loop

Now

Model ready → Register Model Metadata → Enable shadow / experiment → Monitor metrics

Launching new models

Previously

Days to Weeks

Model ready

↓

Prod Feature extractors

↓

Prediction code

↓

Rollout logic

↓

Monitor/ Debug loop

Now

Minutes to Hours

Model ready

↓

Register Model Metadata

↓

Enable shadow / experiment

↓

Monitor metrics

Takeaways

Robust tools are critical to deploying ML in production

Templatizing ML best practices lets us focus on algorithms

Particularly useful tools are

- **Common Features Store**
- **Experimentation and shadow systems**
- **Predictions and Features monitoring**

Thank you!

Interested? We are hiring!
www.doordash.com/careers

DOORDASH

Raghav Ramesh

raghav at doordash.com

@raghavramesh

LinkedIn