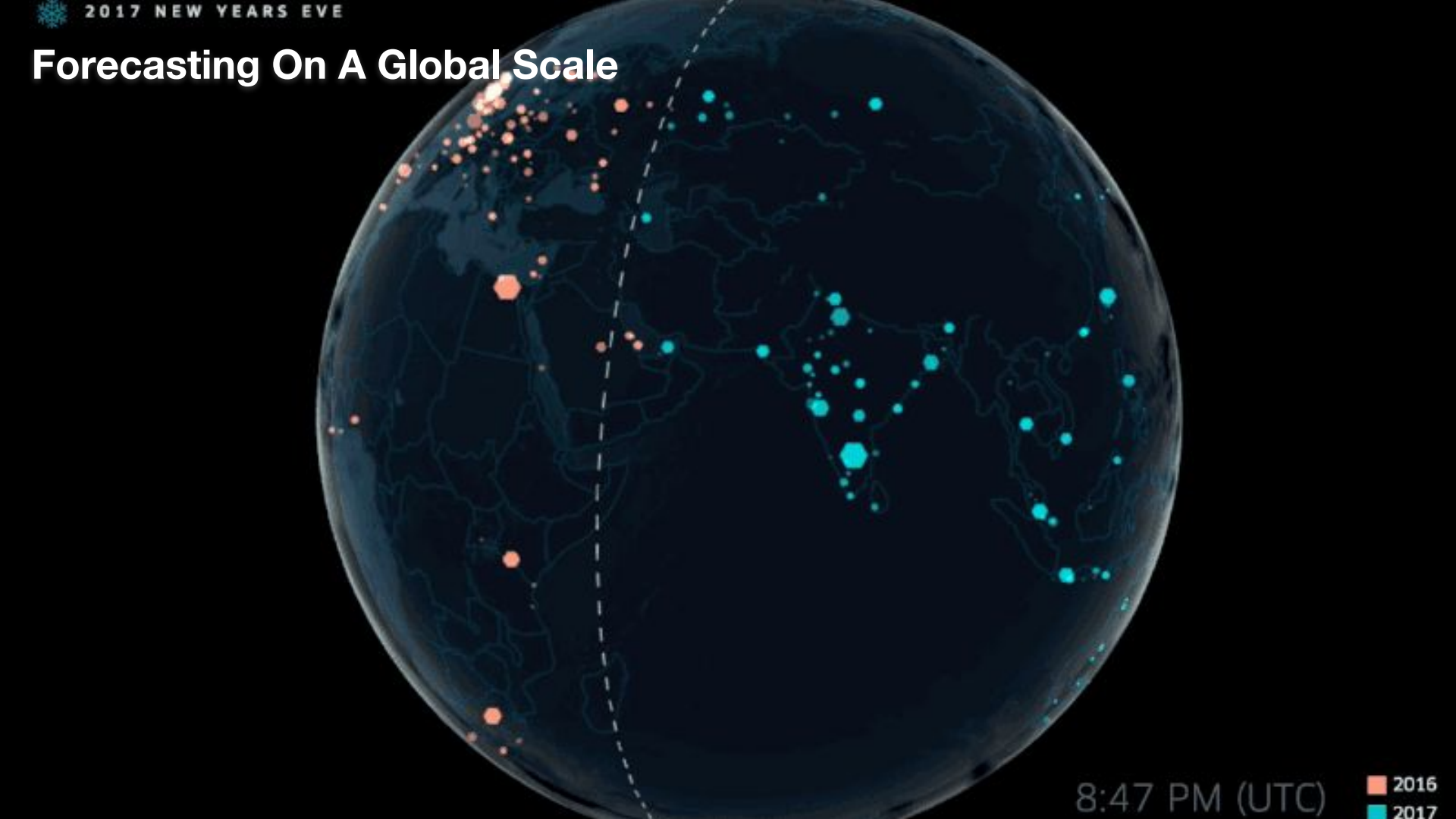
UBER

Machine Learning Pipeline for Real-time Forecasting @Uber Marketplace

Chong Sun, Danny Yuan

Forecasting On A Global Scale

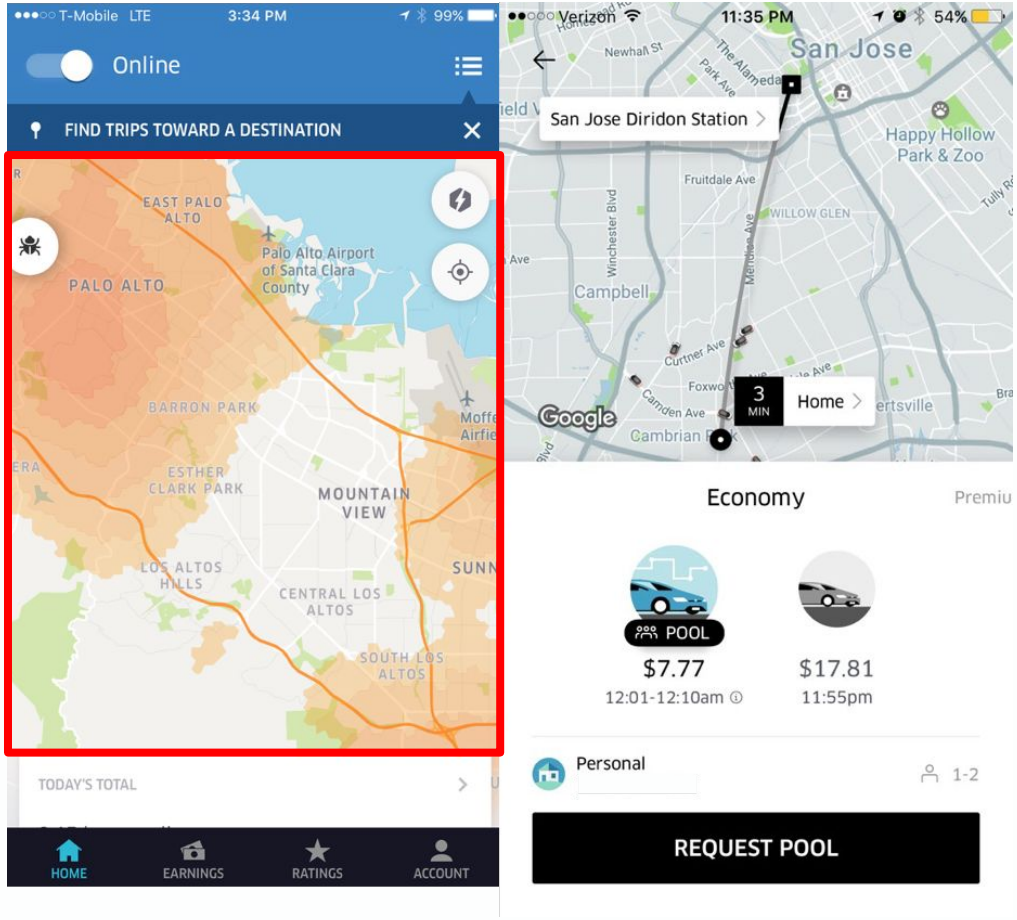


What We Do

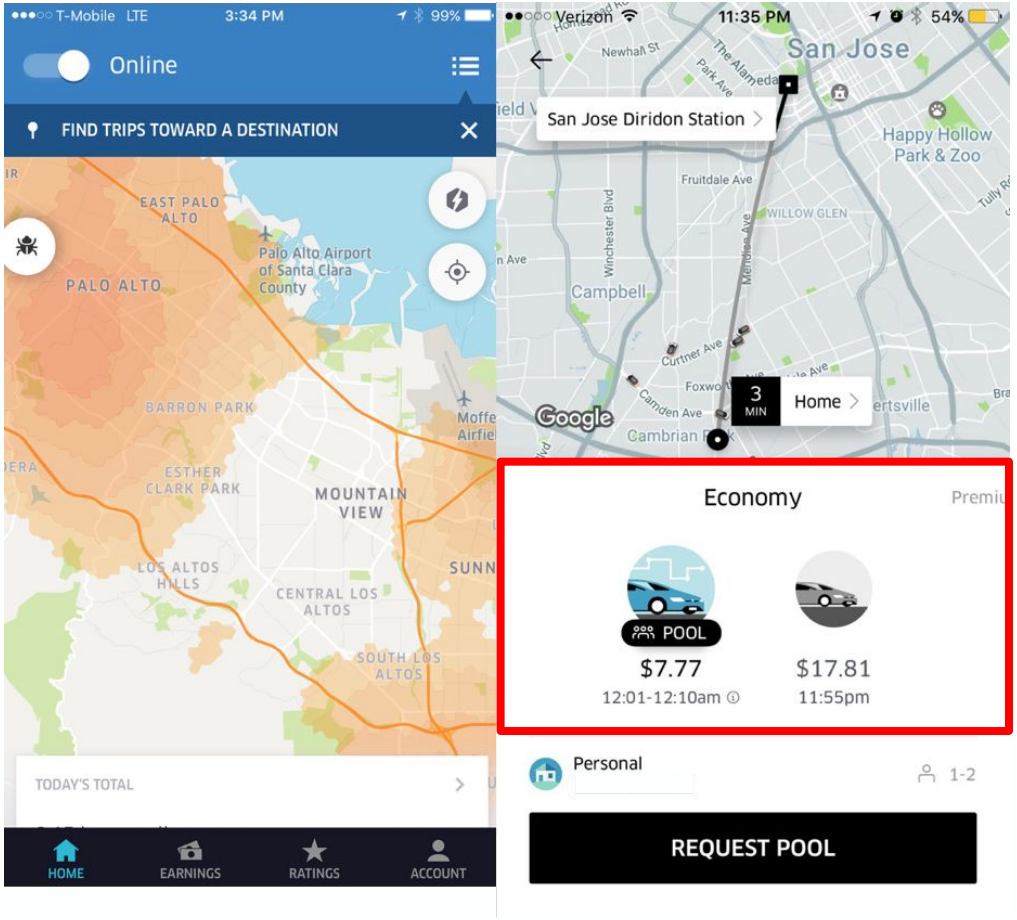
Predict the possible future states of the Uber **marketplace**...

Cases For Real-Time Forecasting

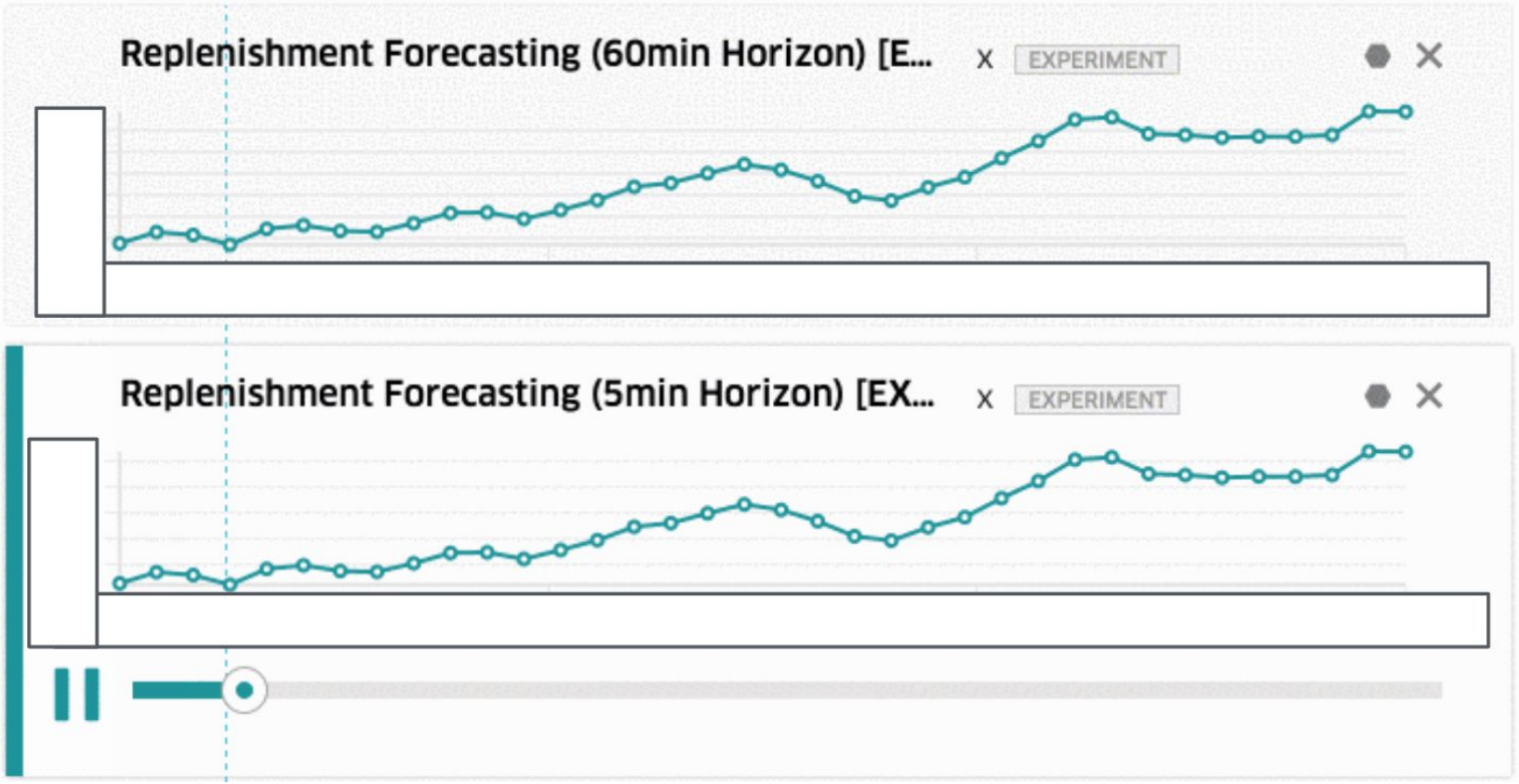
Dynamic Pricing: Every Minute, Every Where



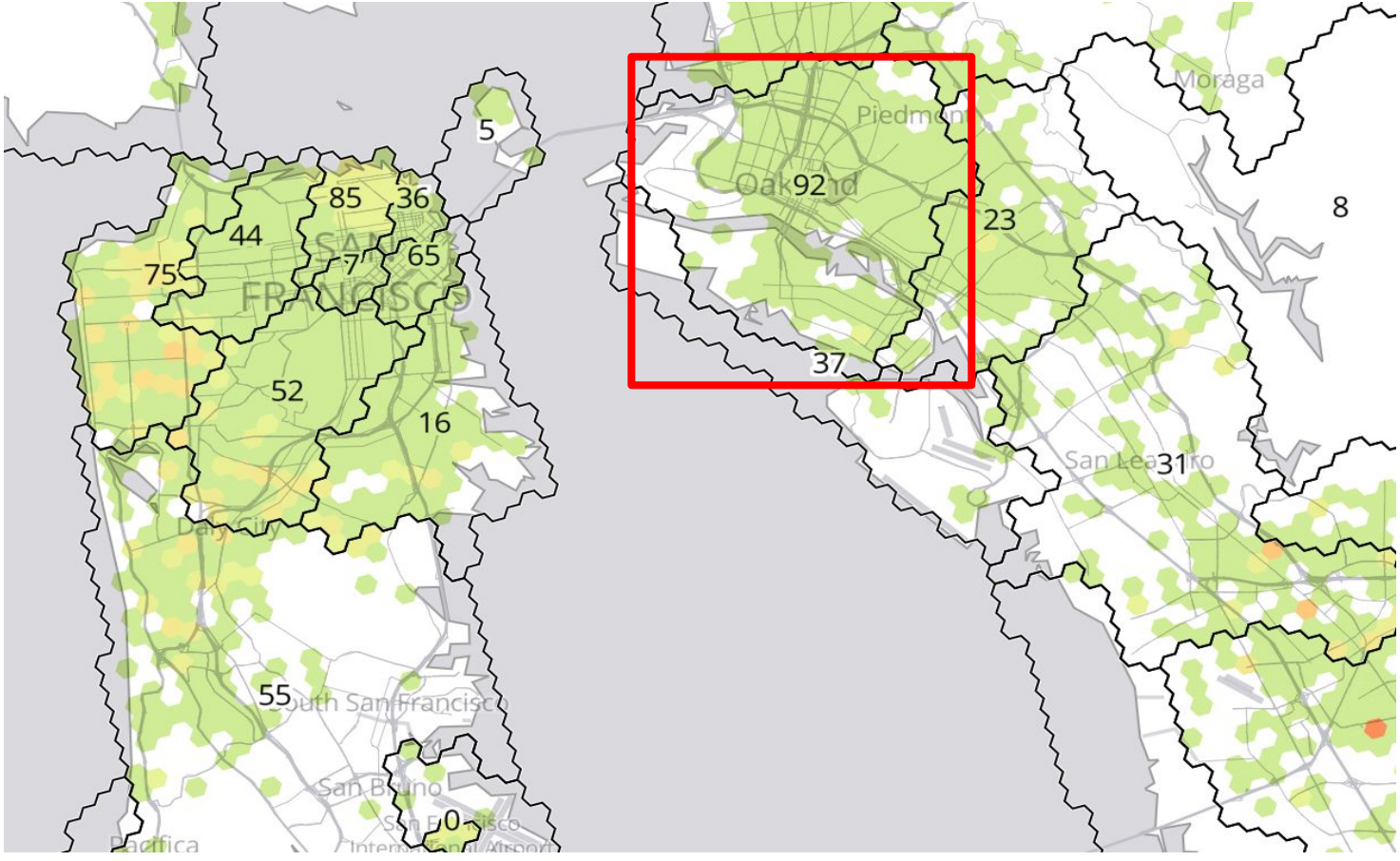
Dynamic Pricing: Every Minute, Every Where, Every Trip



We Forecast Time Series



We Forecast Time Series For Given Geo Locations



Cities

600+

Hexagons / city

10K

Products

5+

Inferences / min

60+

Quantities

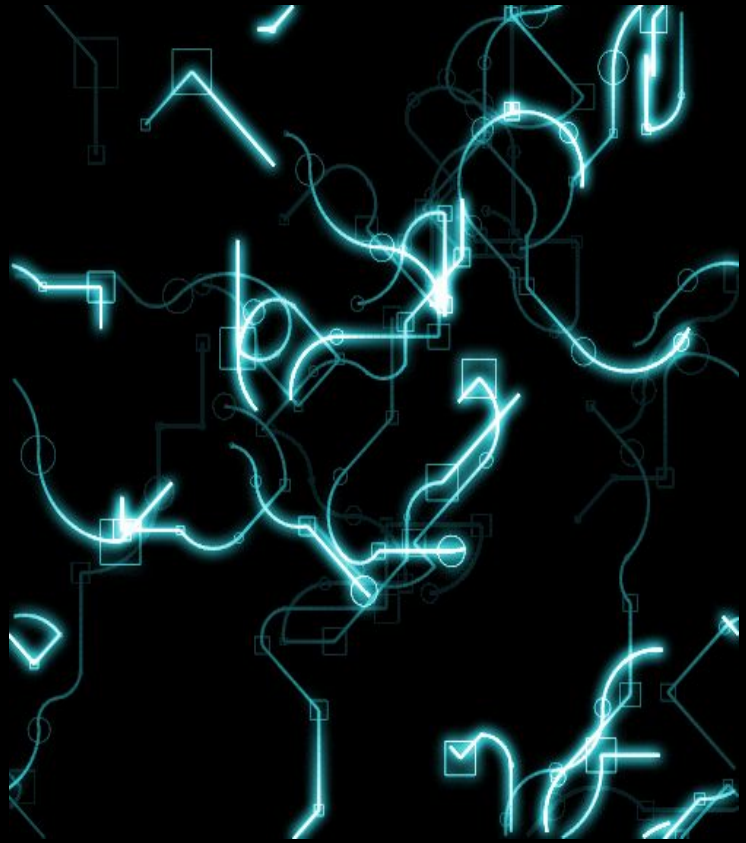
10+

Models

15+

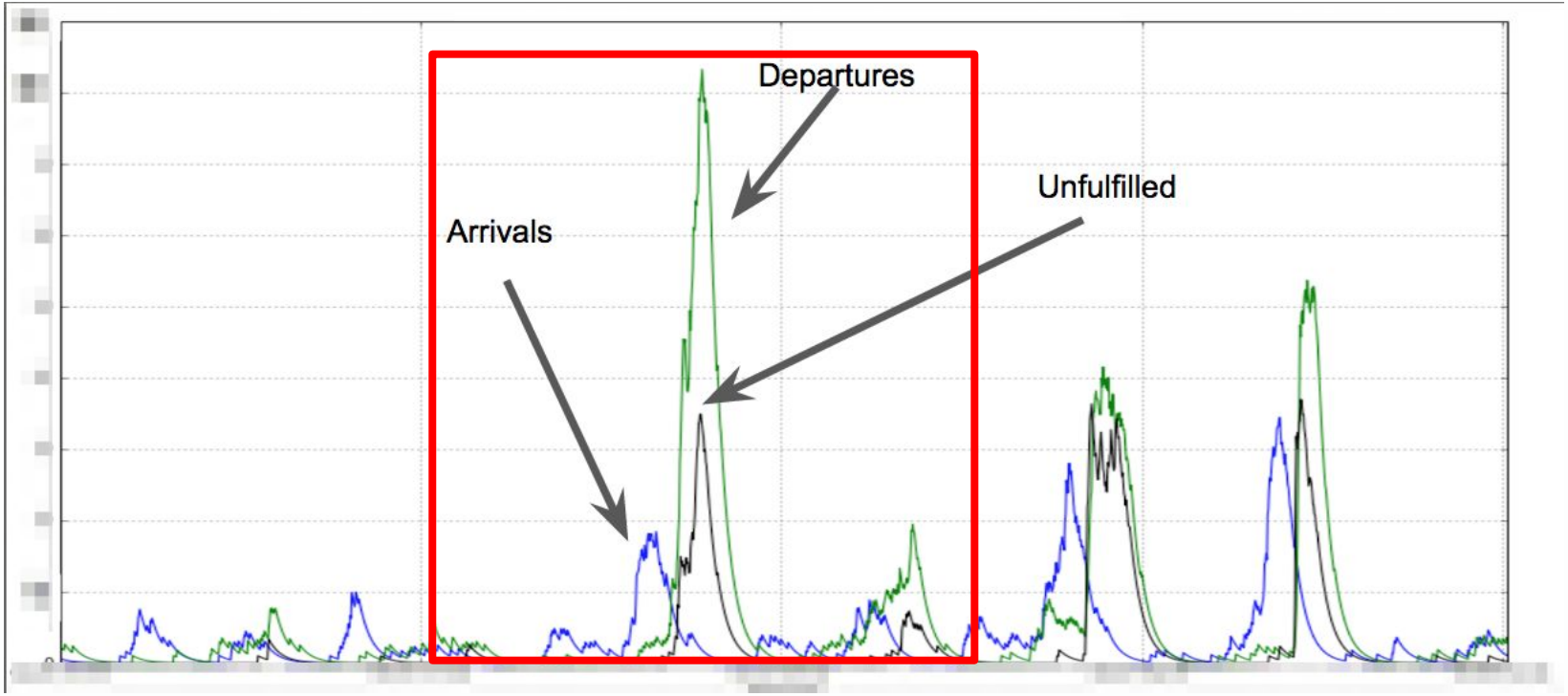
~5B+

Forecasts / min



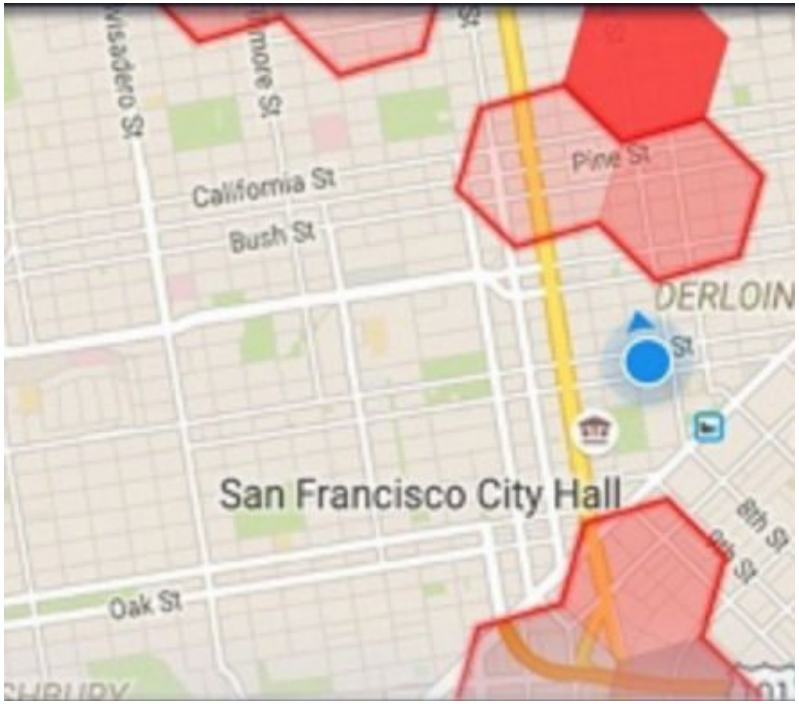
A Few Constraints

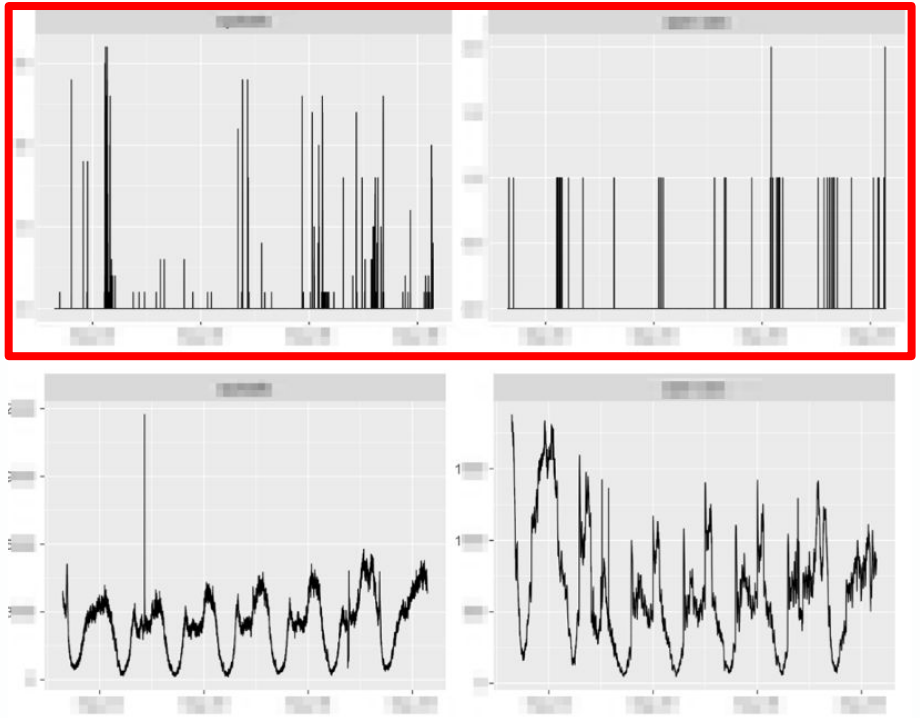
- More recent data has more signals



A Few Constraints

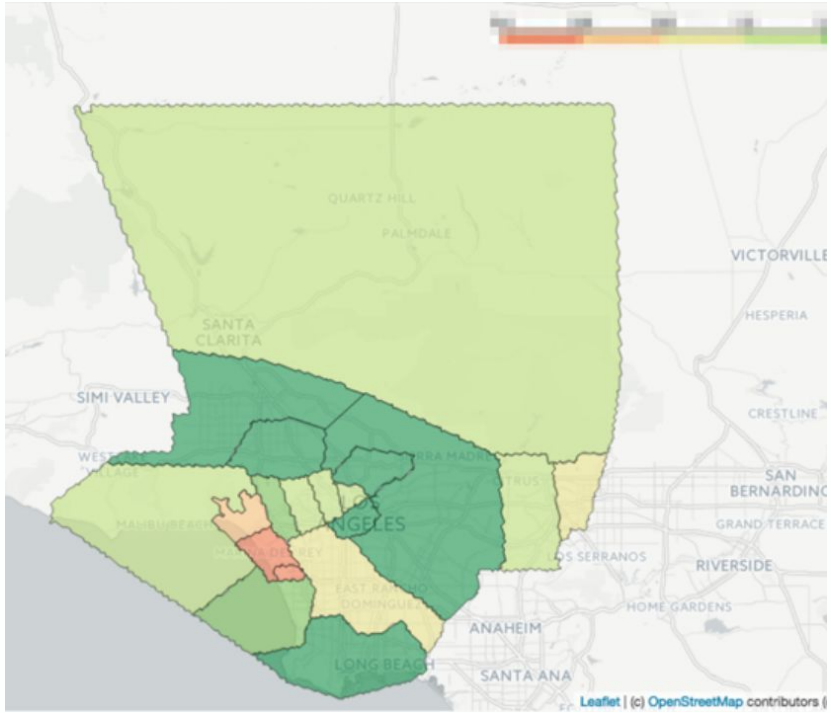
- Smaller areas have more noise

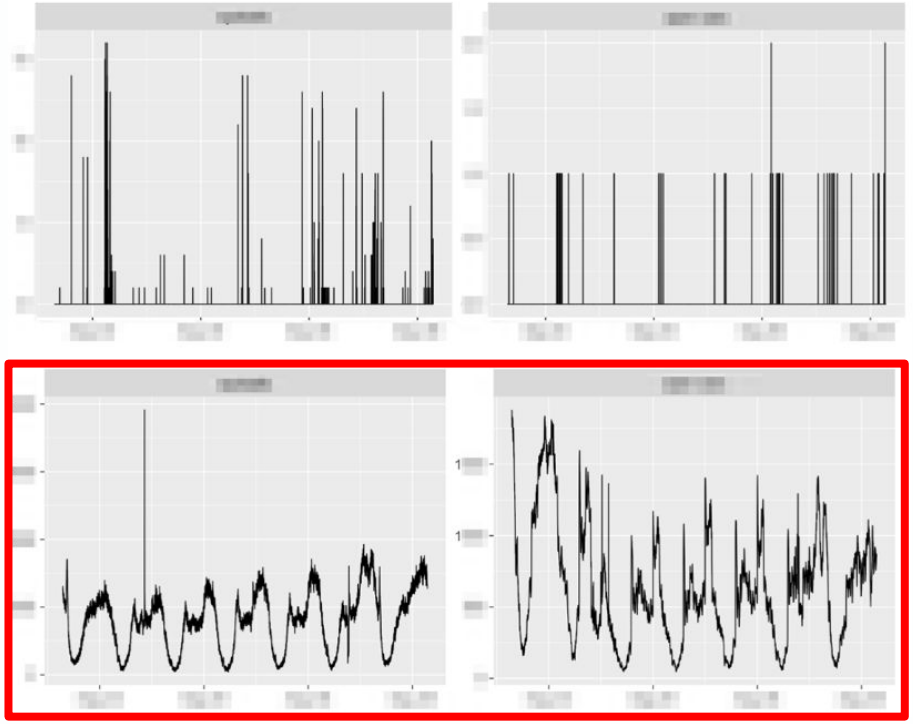




A Few Constraints

- Smaller areas have more noise



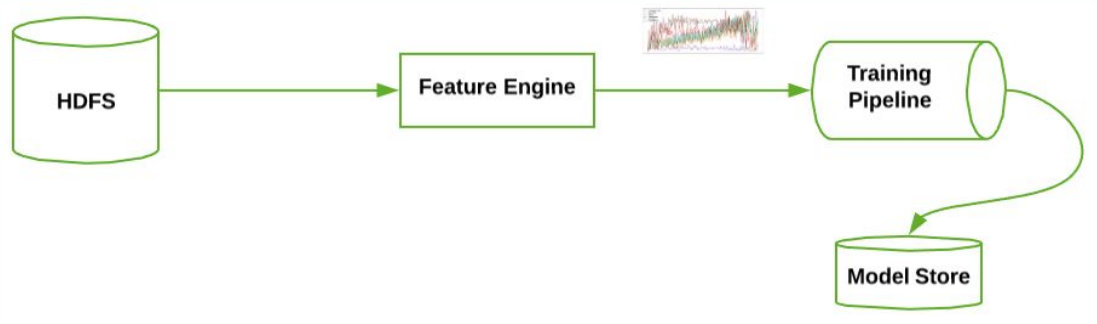


A Few Constraints

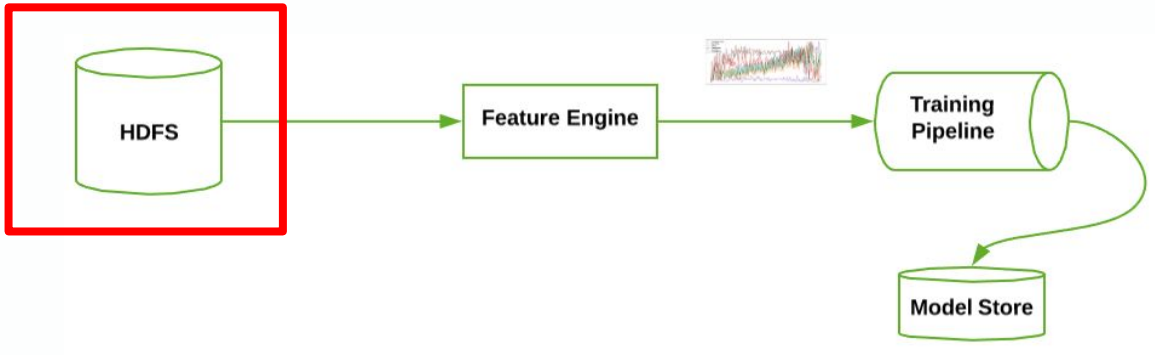
- More recent data has more signals
- Smaller areas have more noise
- We were rolling out business city by city with competing models
    - FFT
    - Kalman Filter
    - Regressions
    - LSTM

First Pipeline

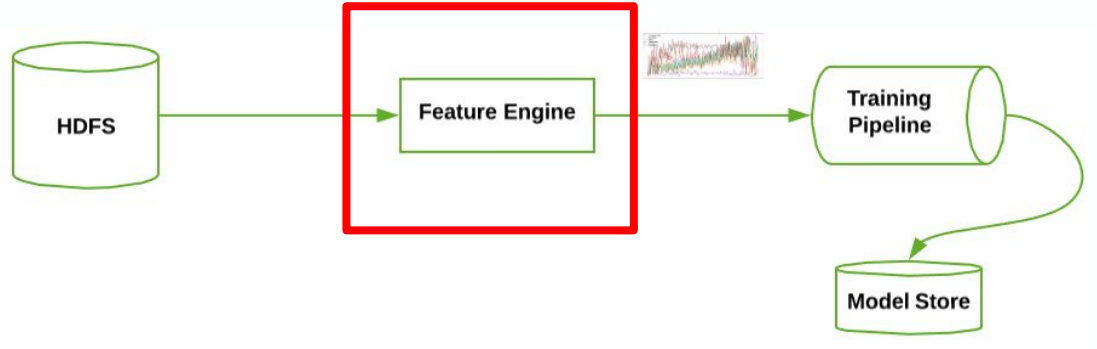
The Training Pipeline



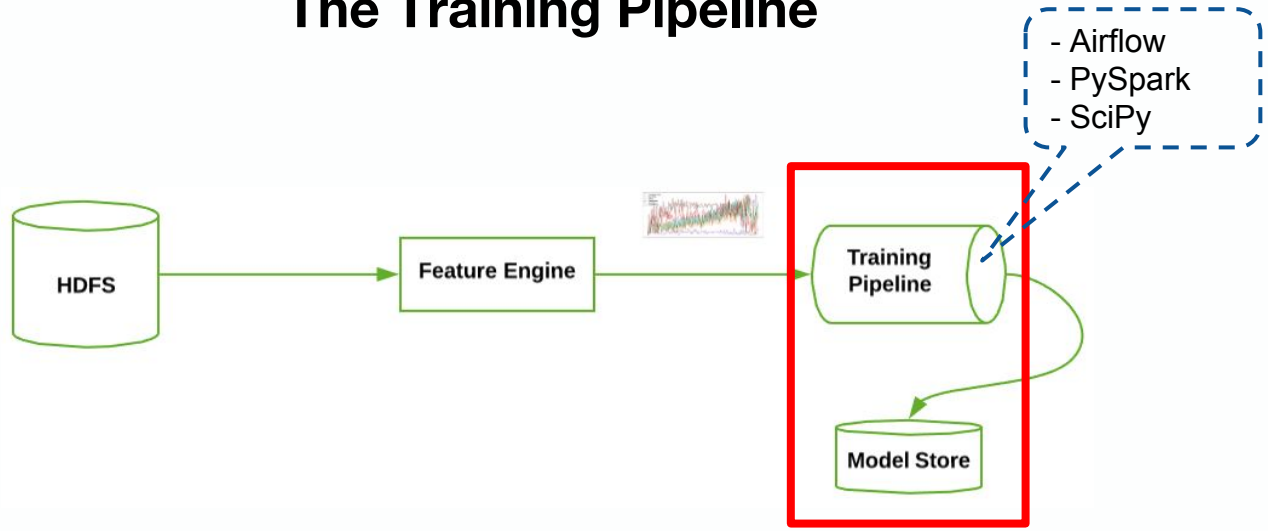
The Training Pipeline



The Training Pipeline

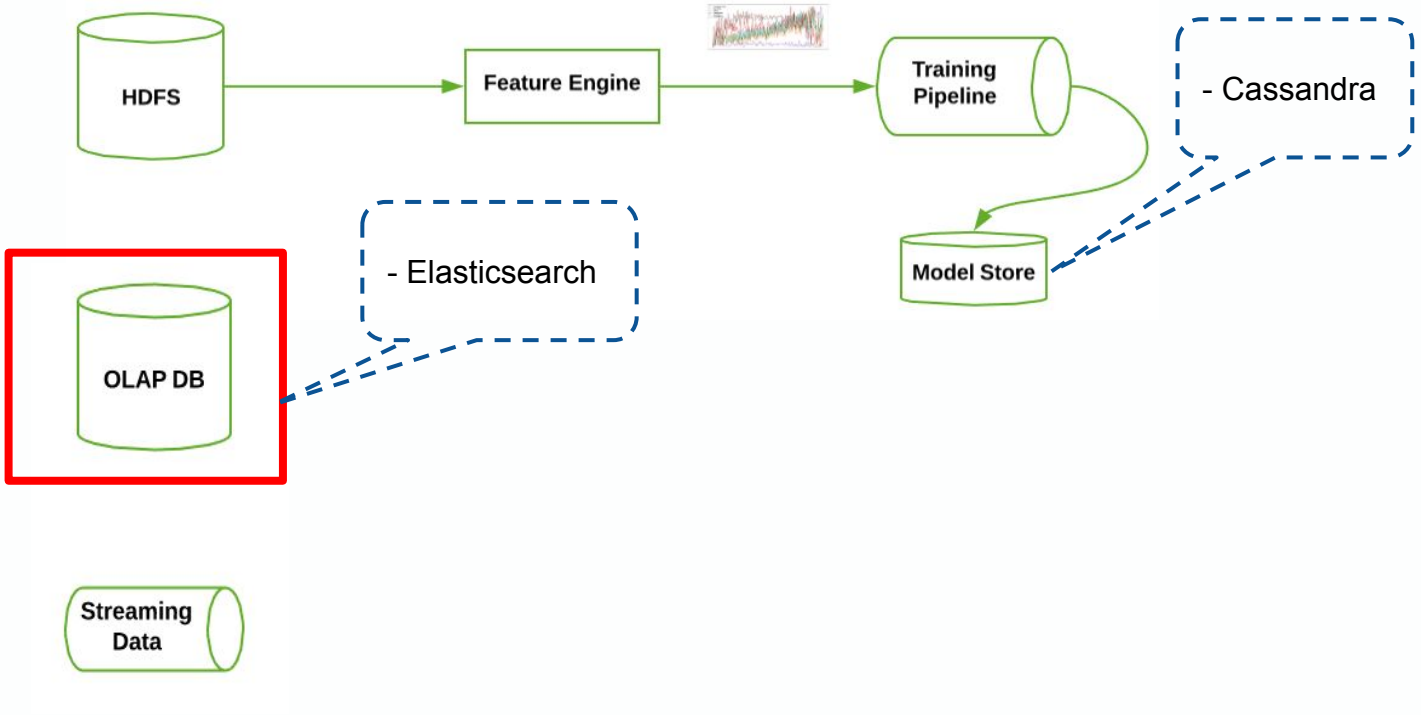


The Training Pipeline

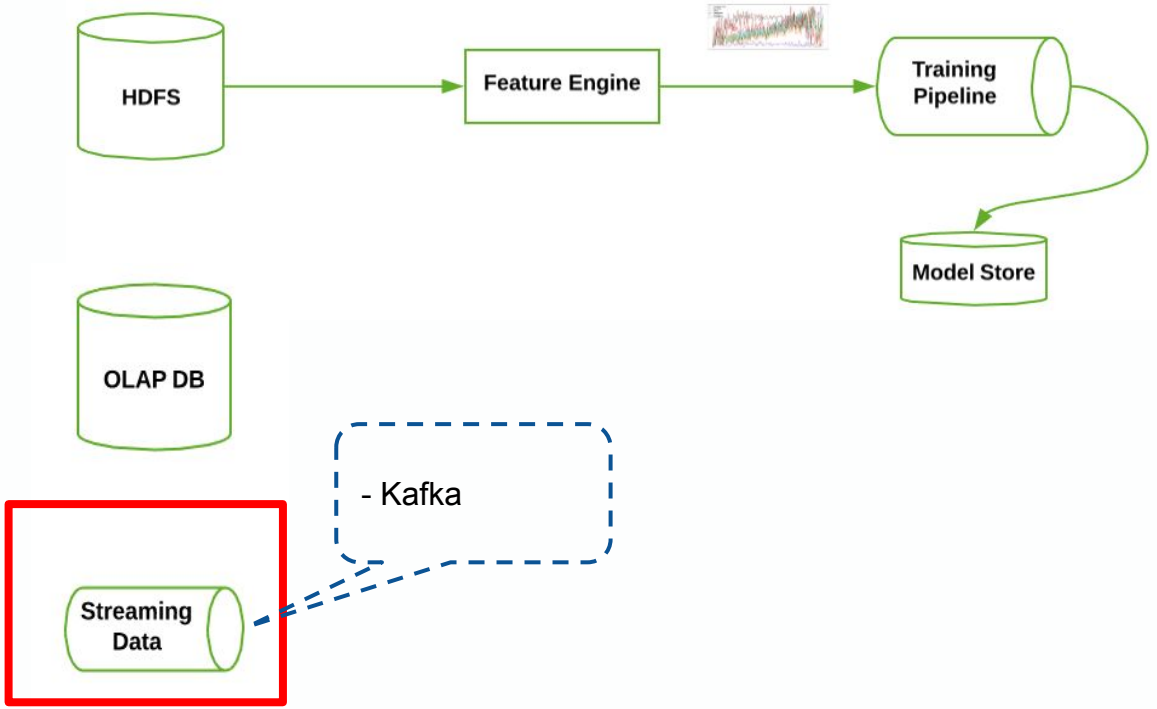


The Training Pipeline

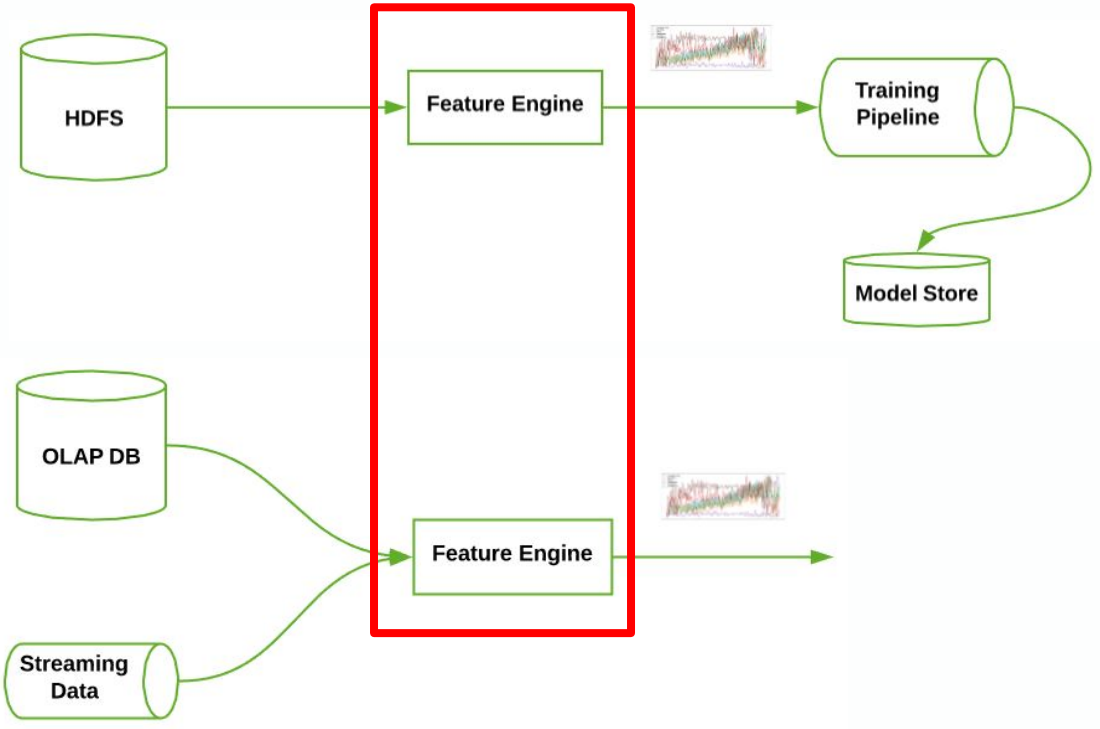
A Need for Fast Time Series DB



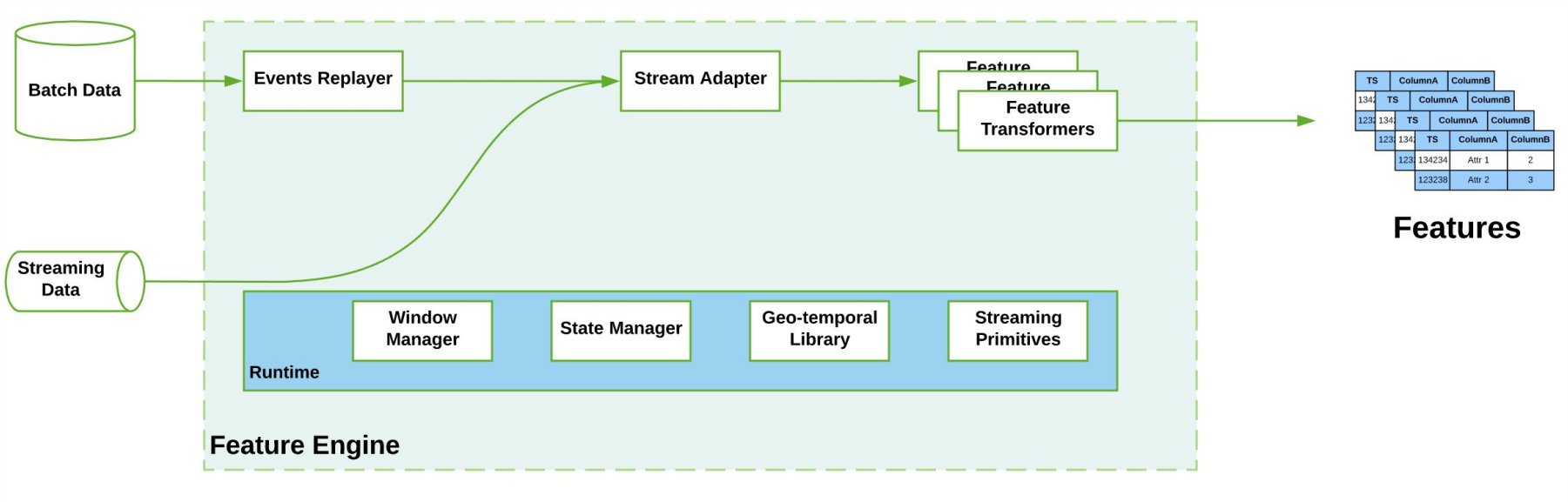
A Need For Streaming Data



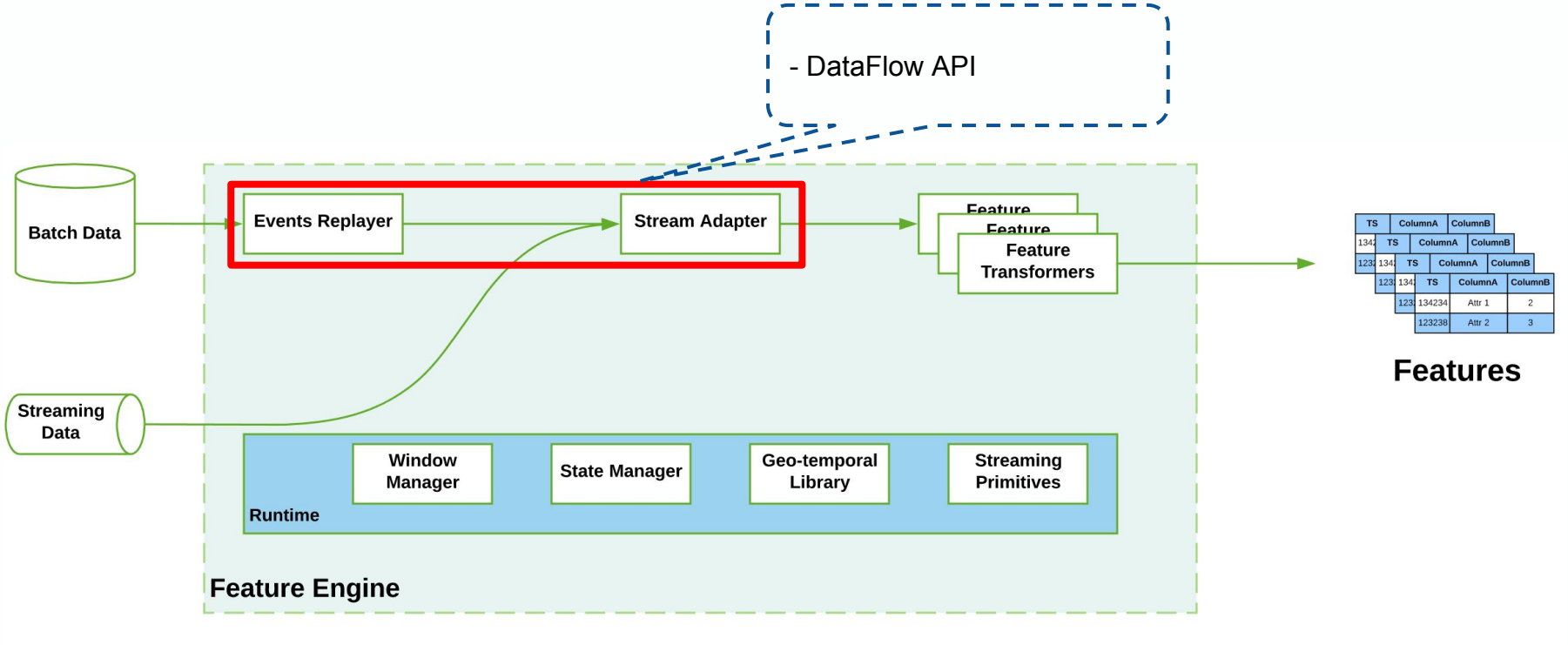
A Need For Unified Feature Engine



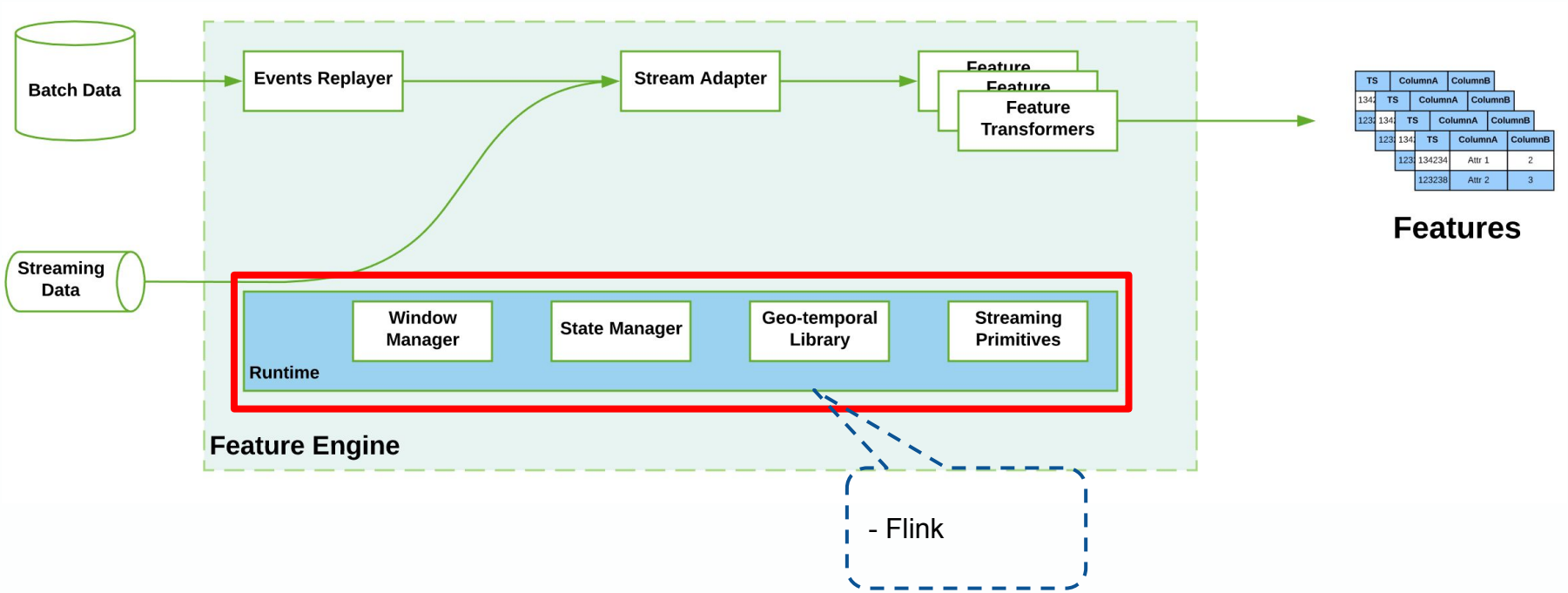
A Digression To Feature Engine



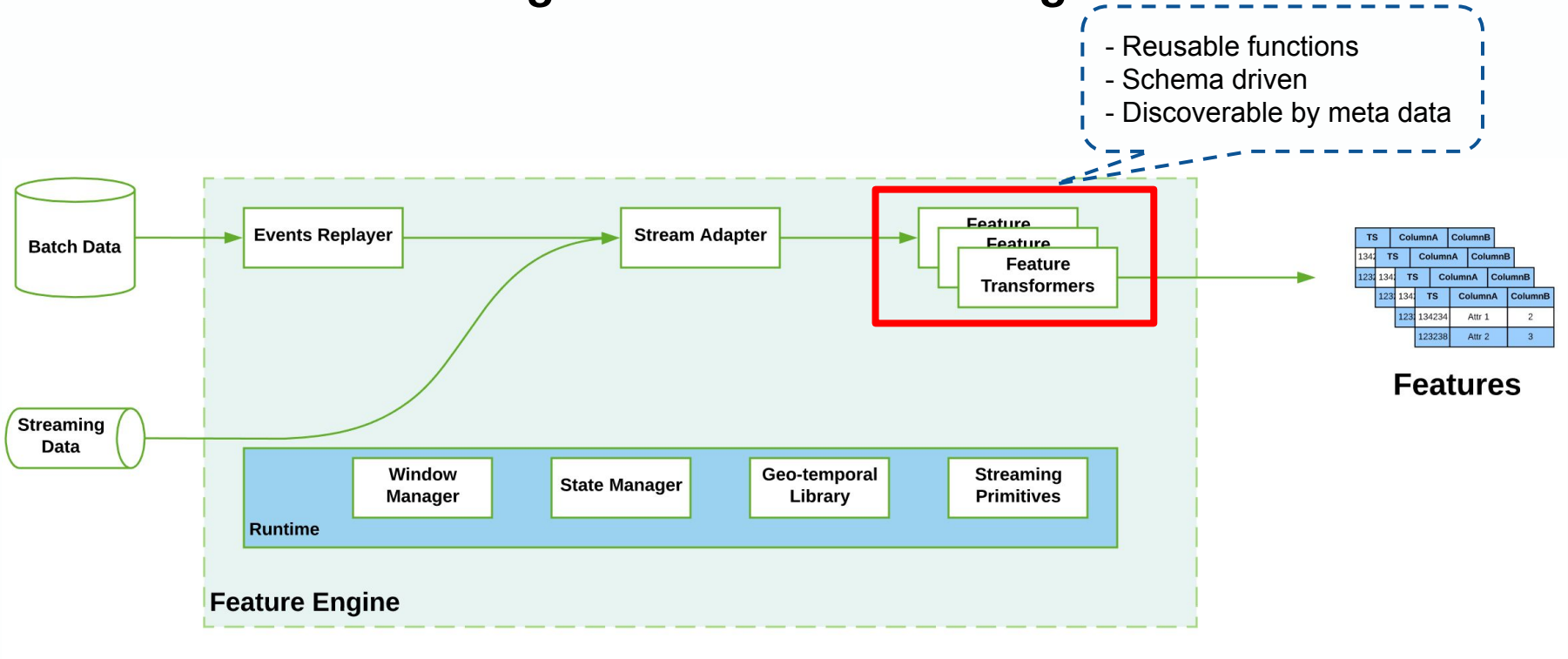
A Digression To Feature Engine



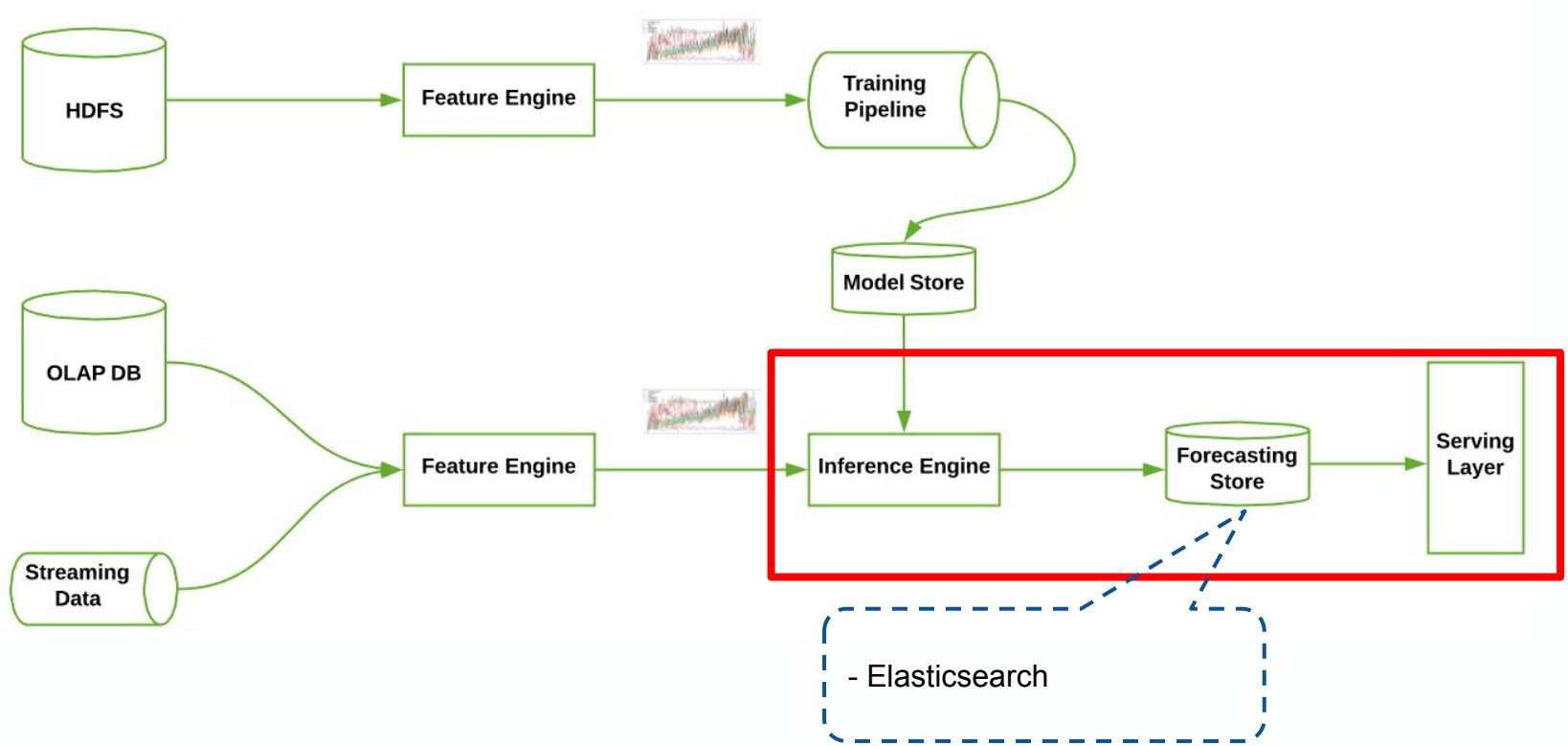
A Digression To Feature Engine



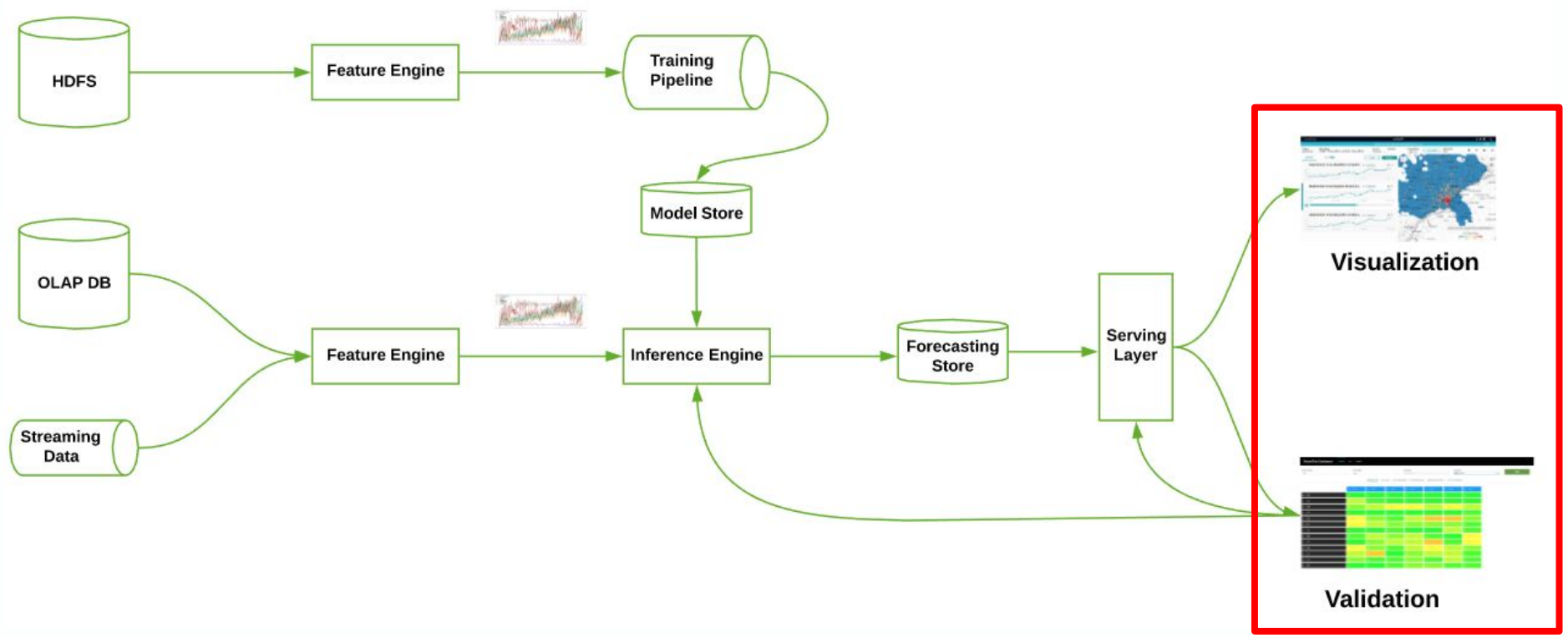
A Digression To Feature Engine



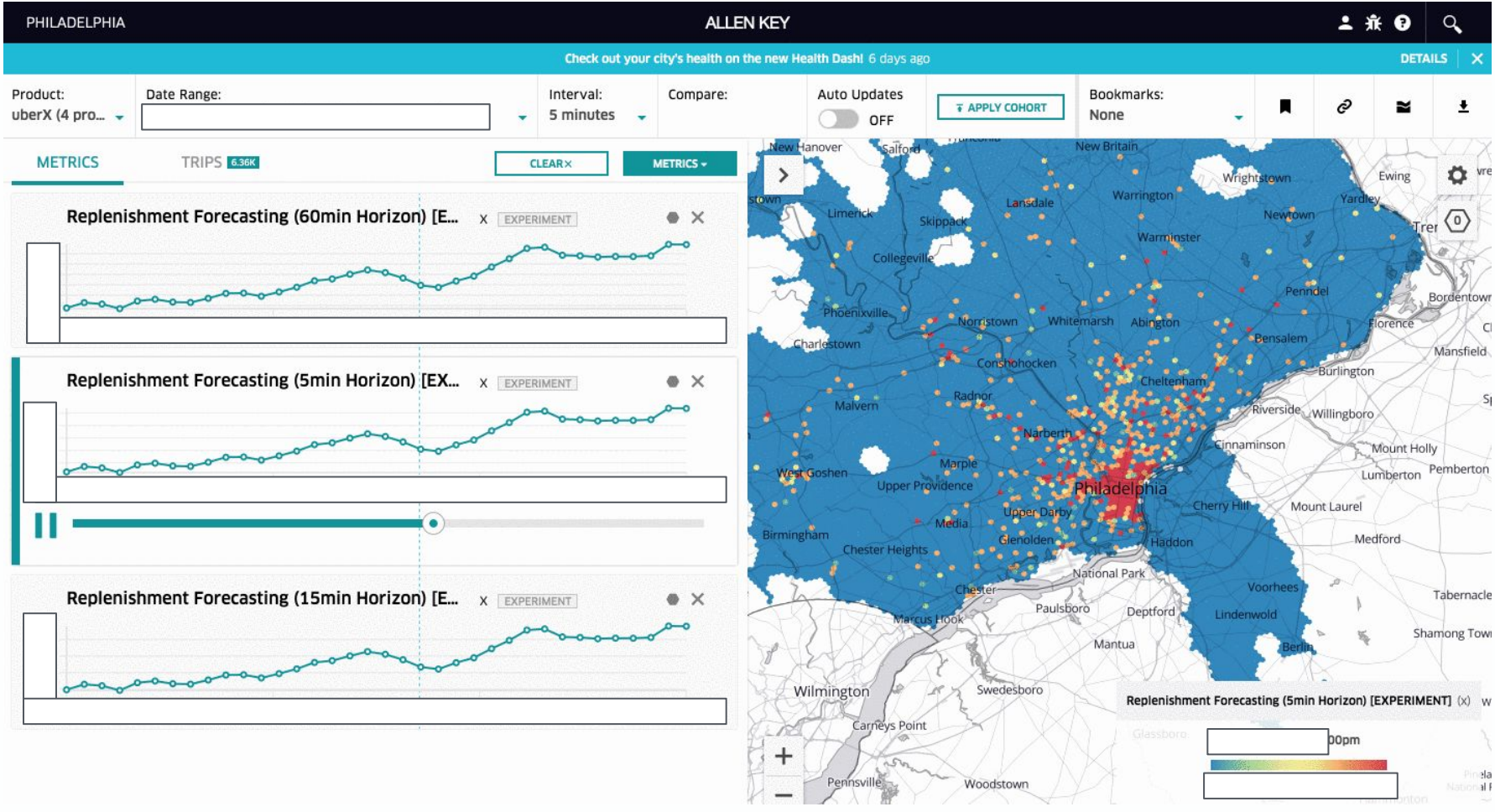
Inferencing Pipeline



Inferencing Pipeline

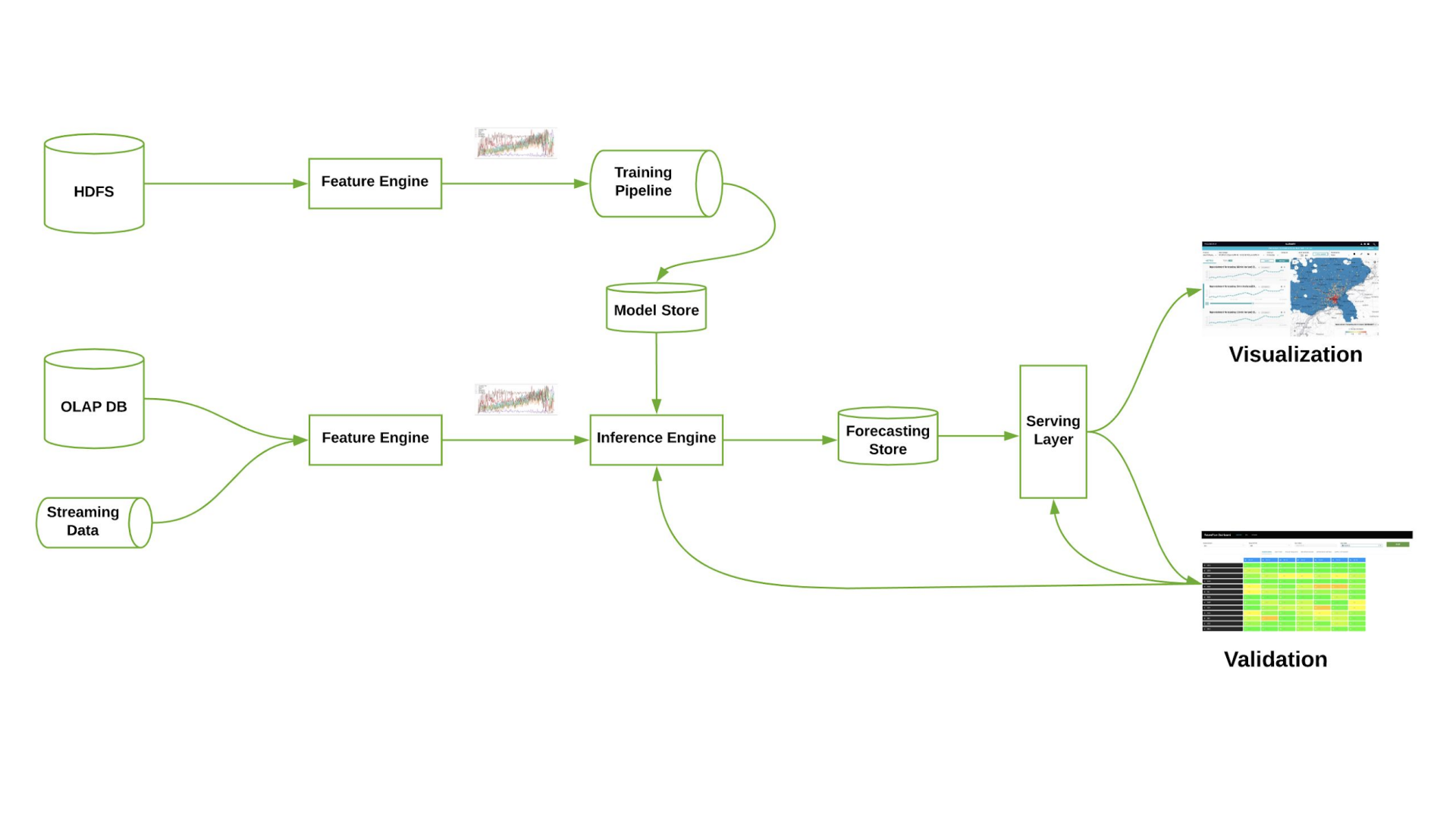


Real-time Visualization

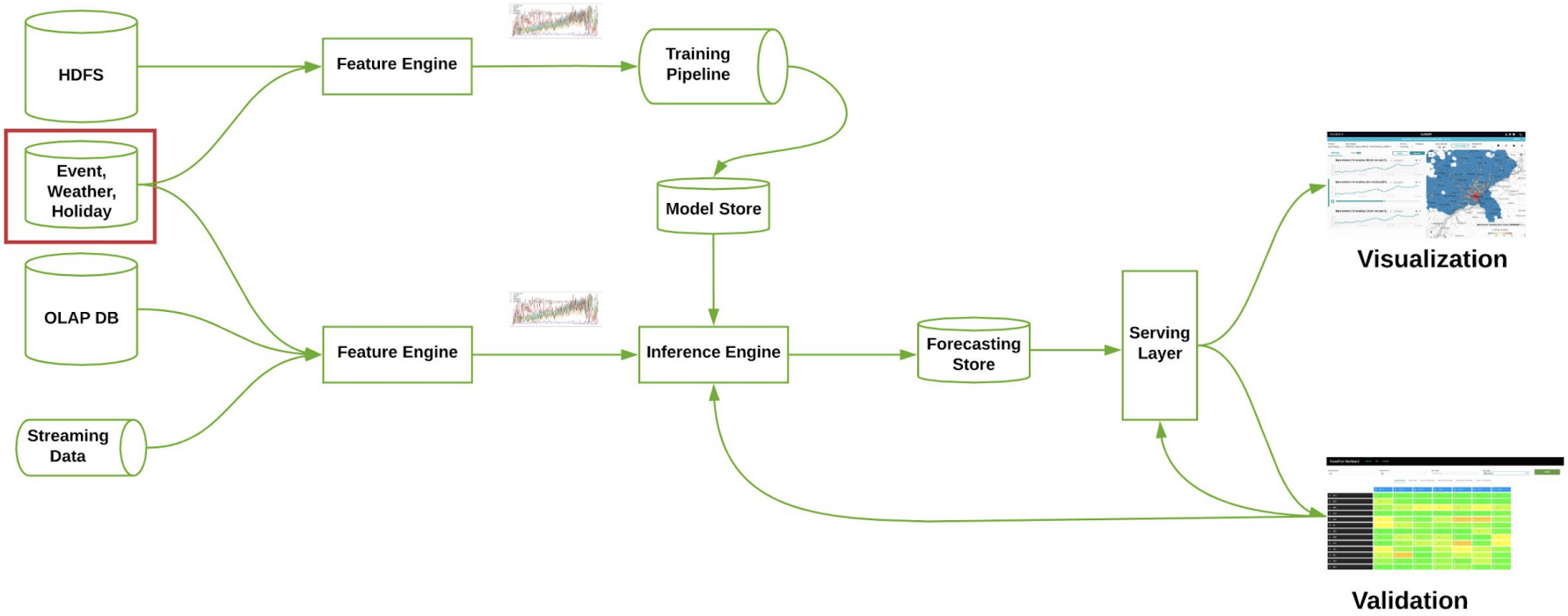


Real-time Validation

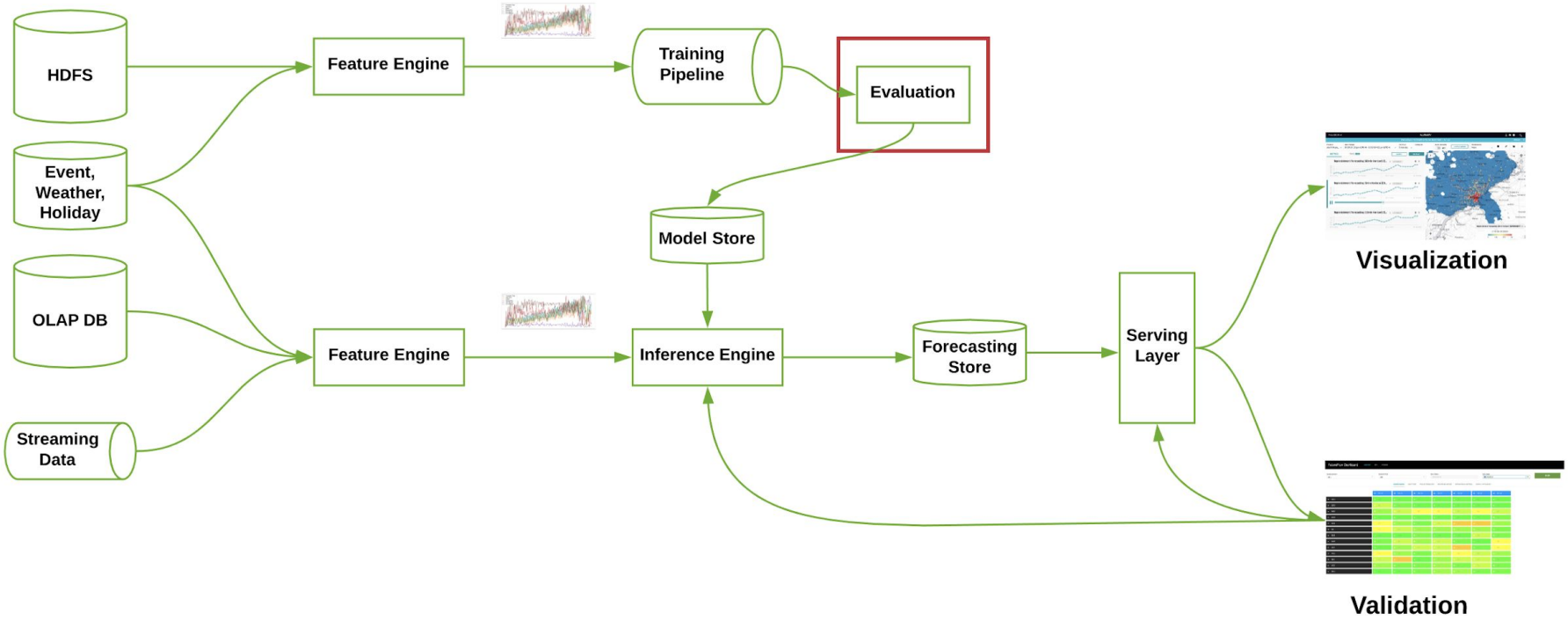
A New Challenge: Model Management


HDFS
Feature Engine
Training Pipeline
Model Store
OLAP DB
Feature Engine
Inference Engine
Forecasting Store
Serving Layer
Streaming Data
Visualization
Validation

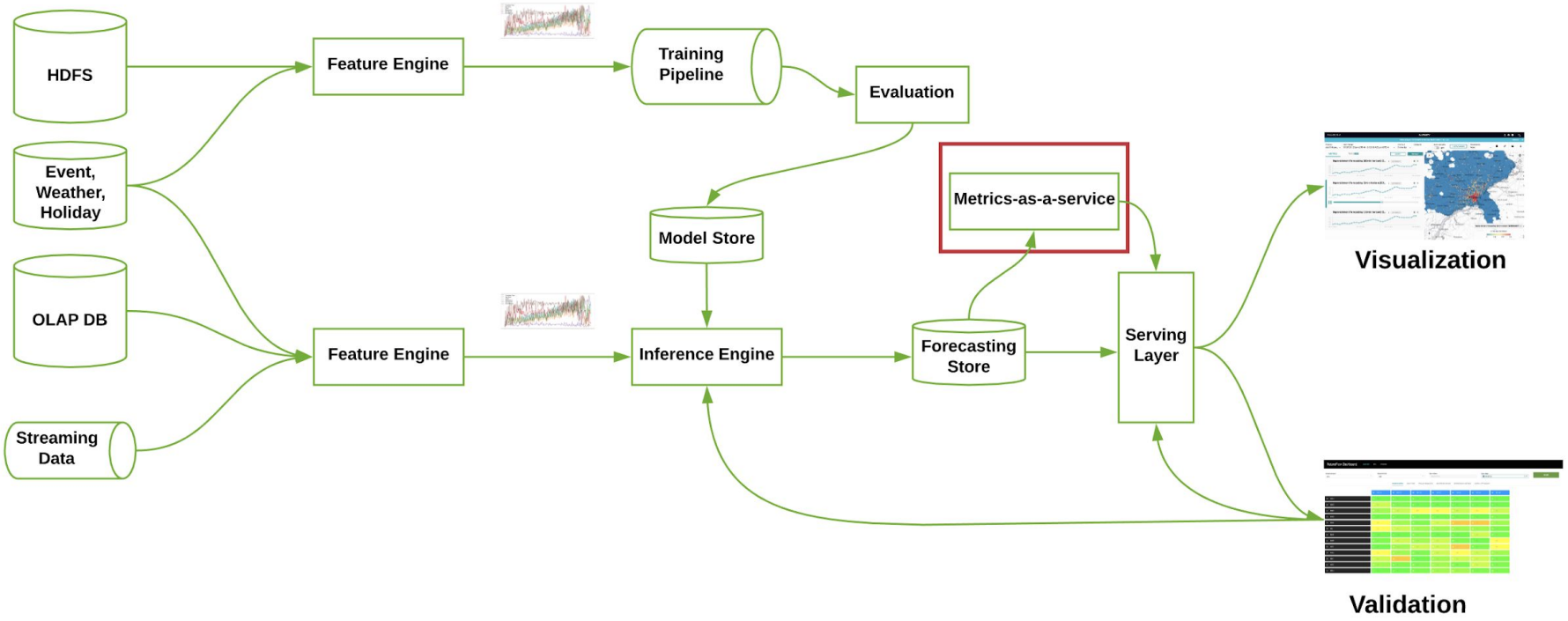
More Signals



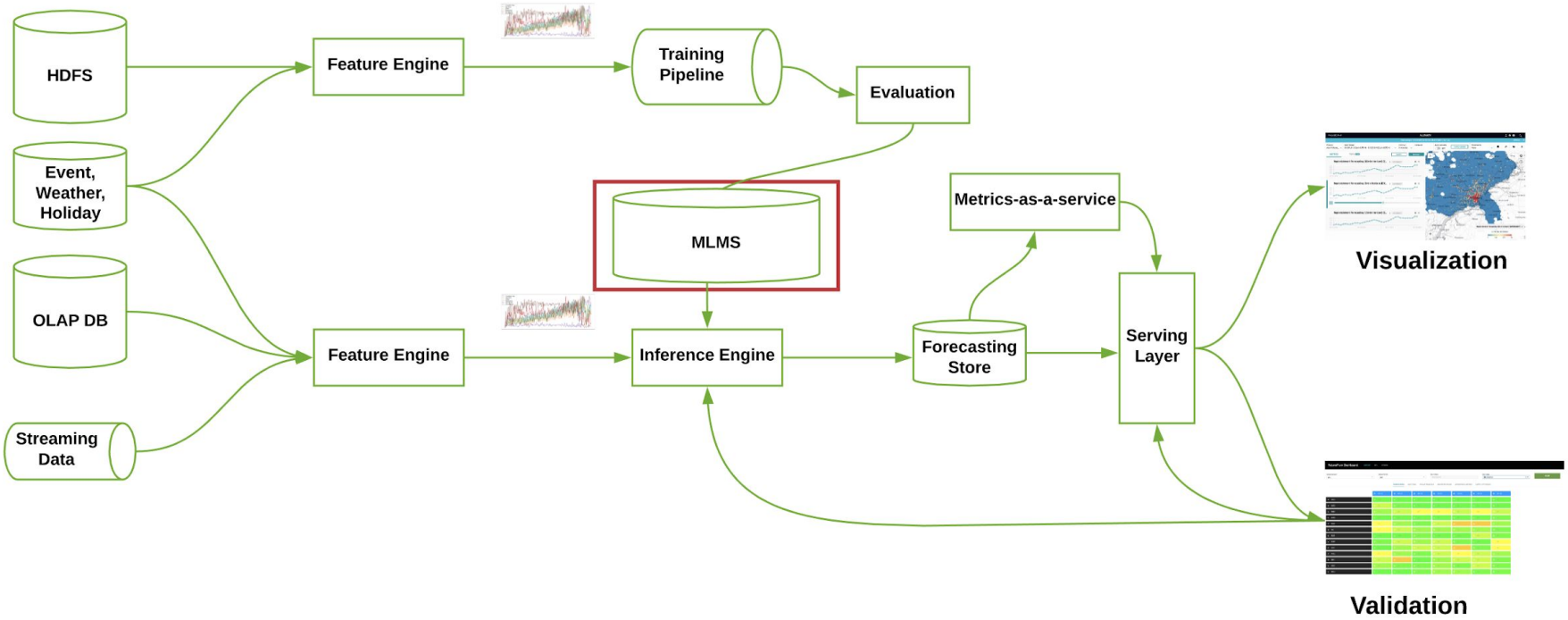
Scalable Model Evaluation



Metrics-as-a-Service



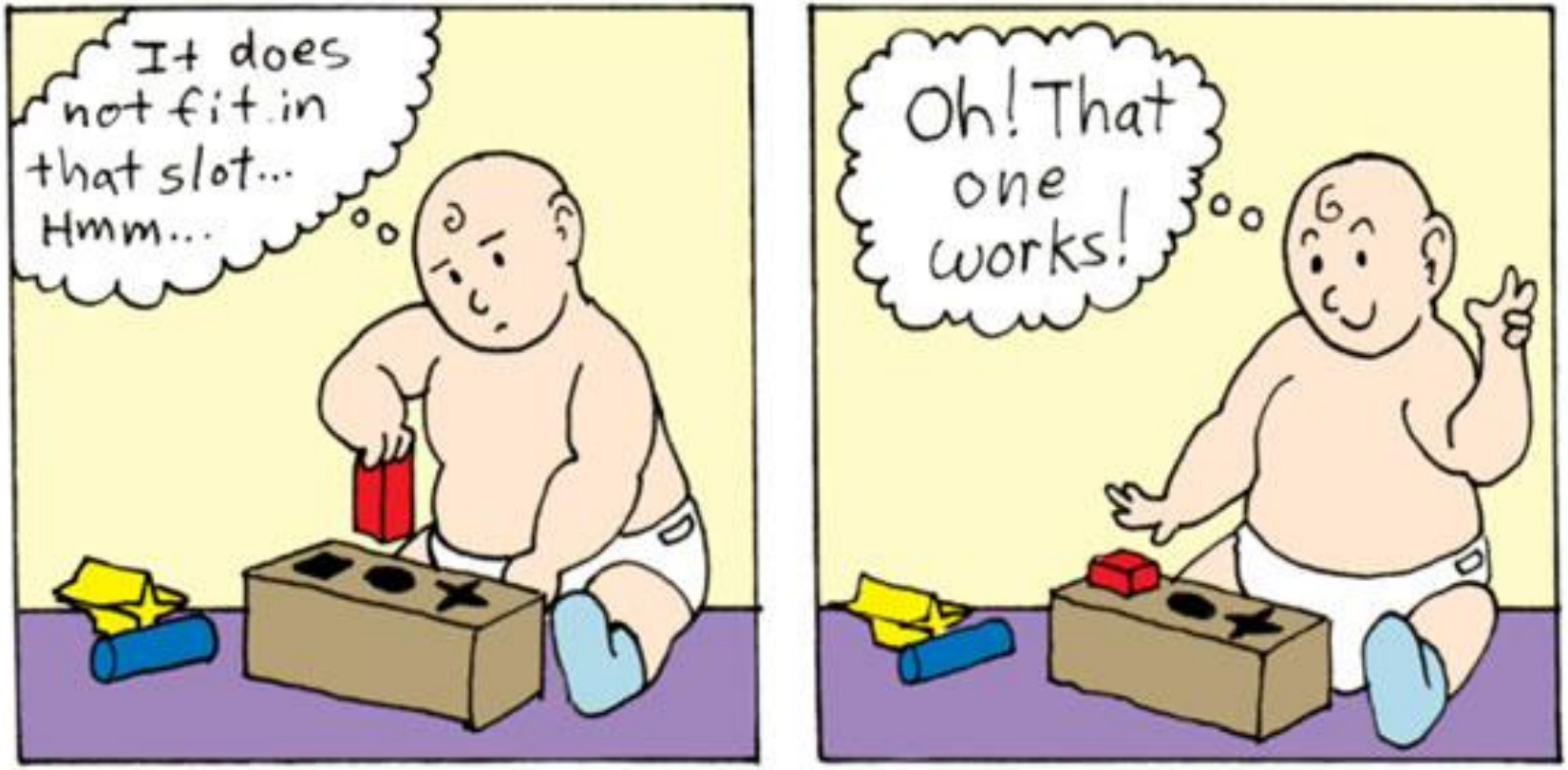
Model Lifecycle Management System (MLMS)



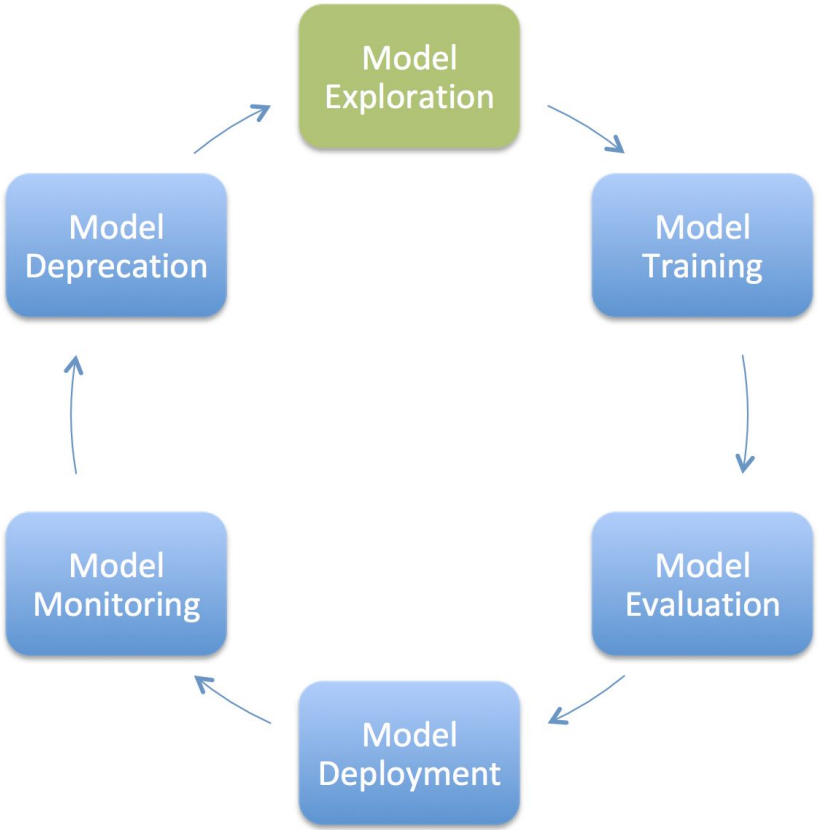
What if you're supporting 5+ teams, 10+ products with 4000+ model instances in production


START

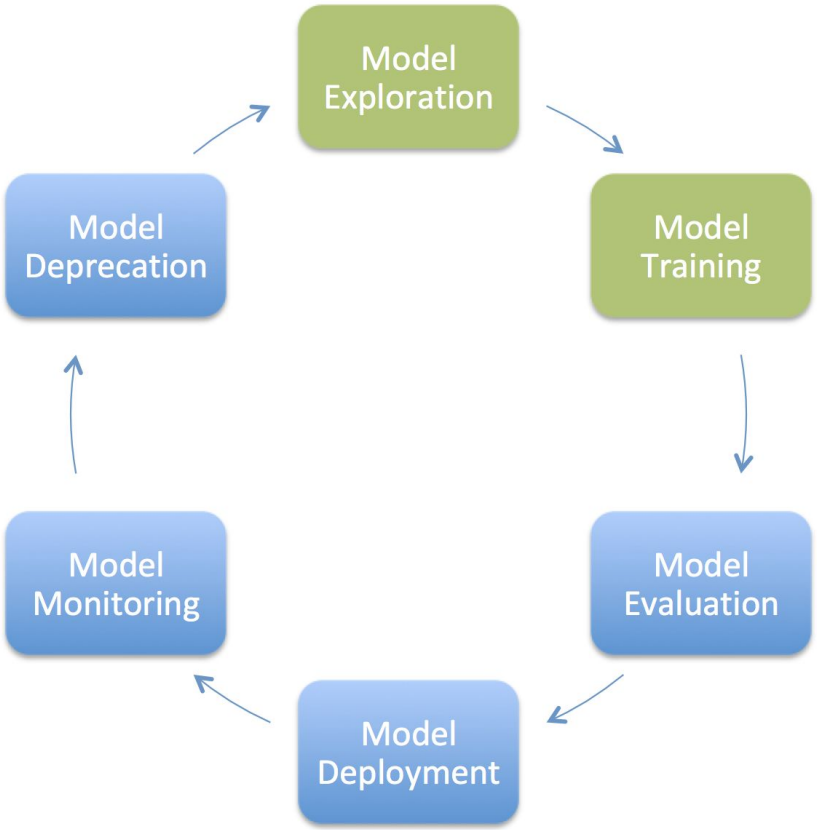

Damian Mor


It does not fit in that slot... Hmm...
Oh! That one works!

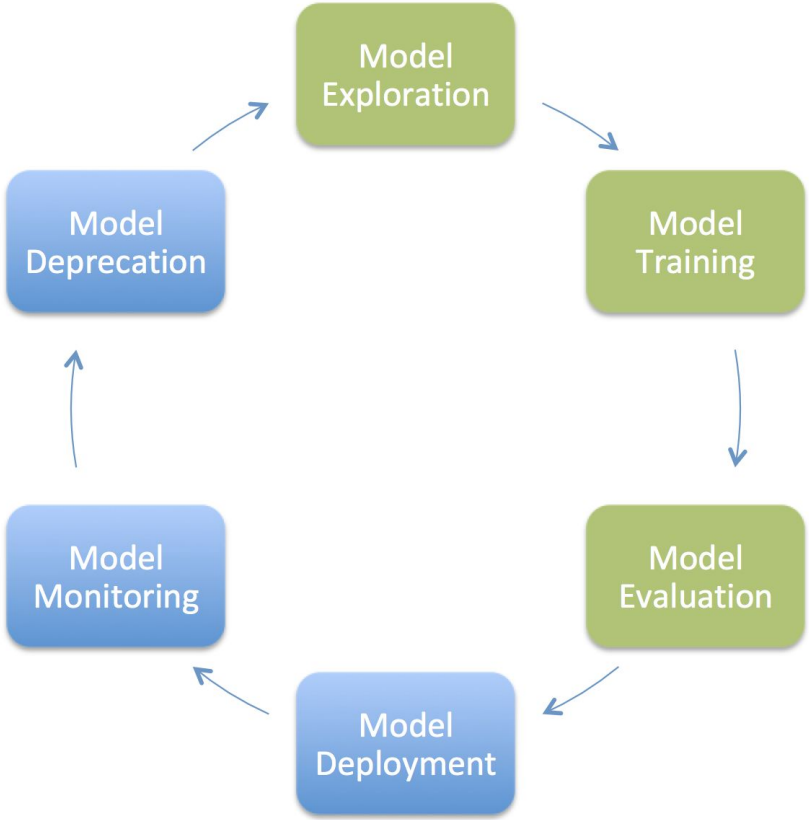
Machine Learning Model Lifecycle



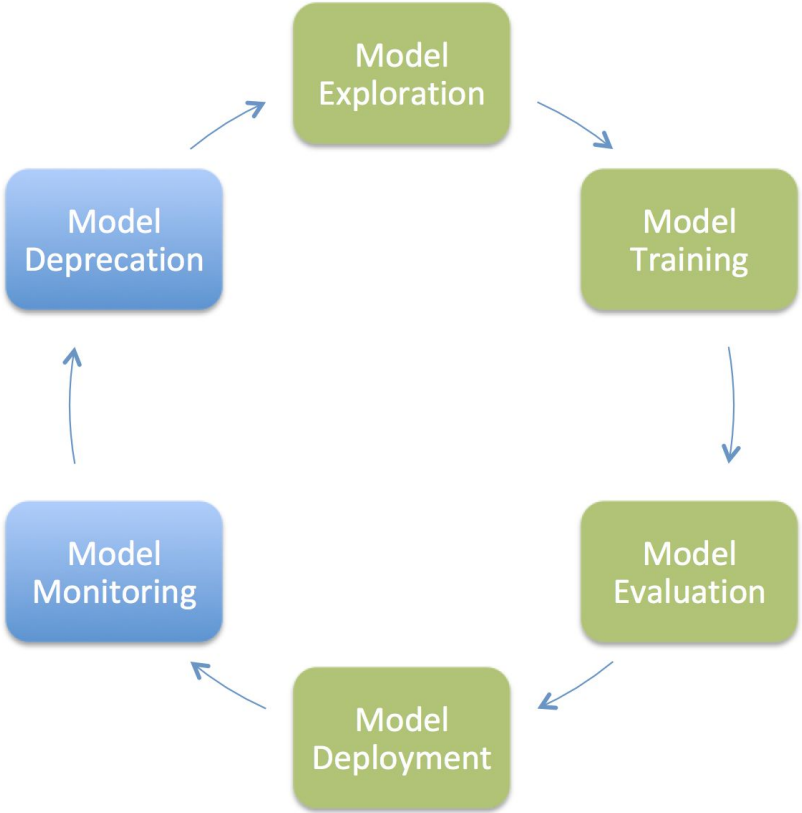
Machine Learning Model Lifecycle



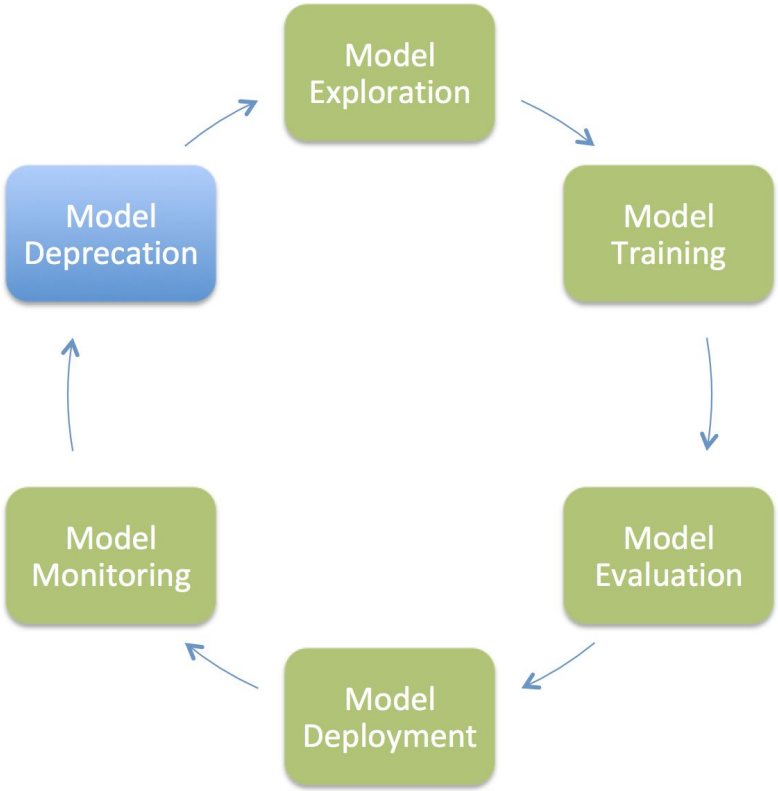
Machine Learning Model Lifecycle



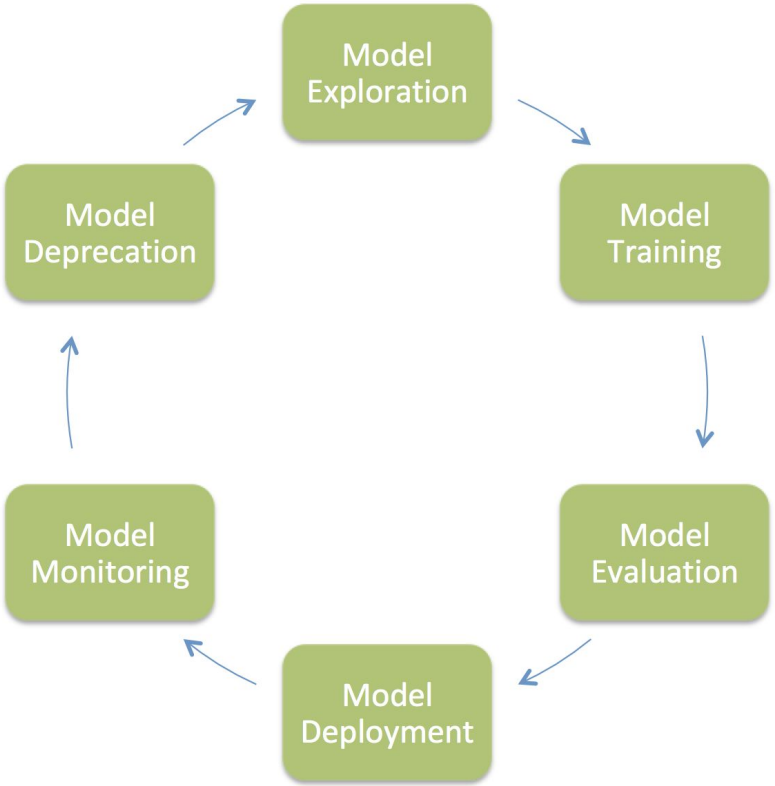
Machine Learning Model Lifecycle



Machine Learning Model Lifecycle



Machine Learning Model Lifecycle



Common Questions in the process ...

- *Where am I going to **save** and **serve** my models?*
- *How do I keep **track** of the model **metadata**, e.g., training data used ?*
- *How can I easily **find** a previous model for testing and performance comparison?*
- *How can I **automatically** deploy a large scale number of models?*
- *When should I decide to **trigger** model re-training?*
- *How can I make sure I would **not override** any (production) models?*
- *How do we manage multiple **dependent** models?*
- *… ...*

Common Questions in the process ...

- *Where am I going to **save** and **serve** my models?*
- *How do I keep **track** of the model **metadata**, e.g., training data used ?*
- *How can I easily **find** a previous model for testing and performance comparison?*
- *How can I **automatically** deploy a large scale number of models?*
- *When should I decide to **trigger** model re-training?*
- *How can I make sure I would **not override** any (production) models?*
- *How do we manage multiple **dependent** models?*
- *... ...*

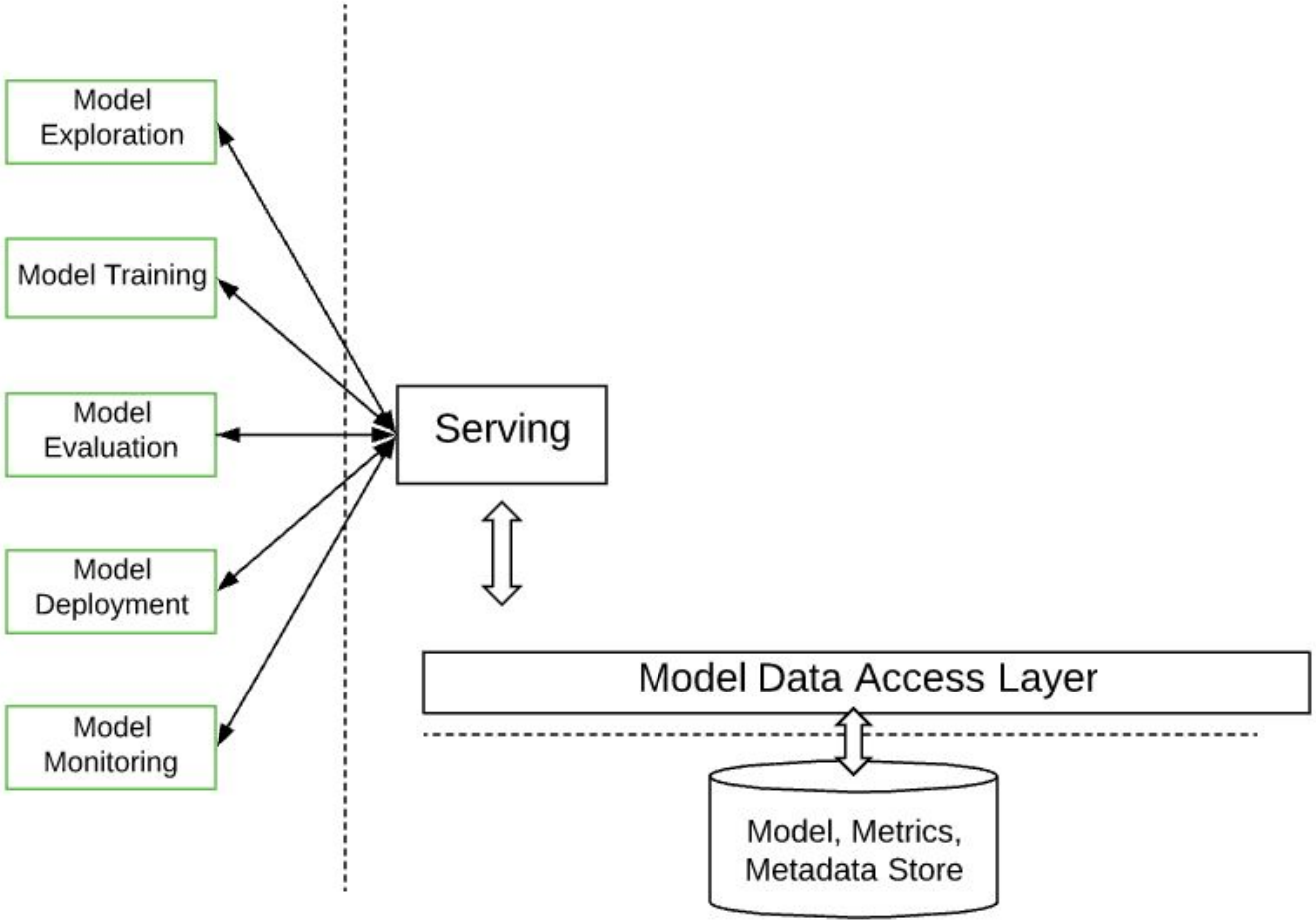
Model Lifecycle Management System (MLMS)

MLMS Design Principles

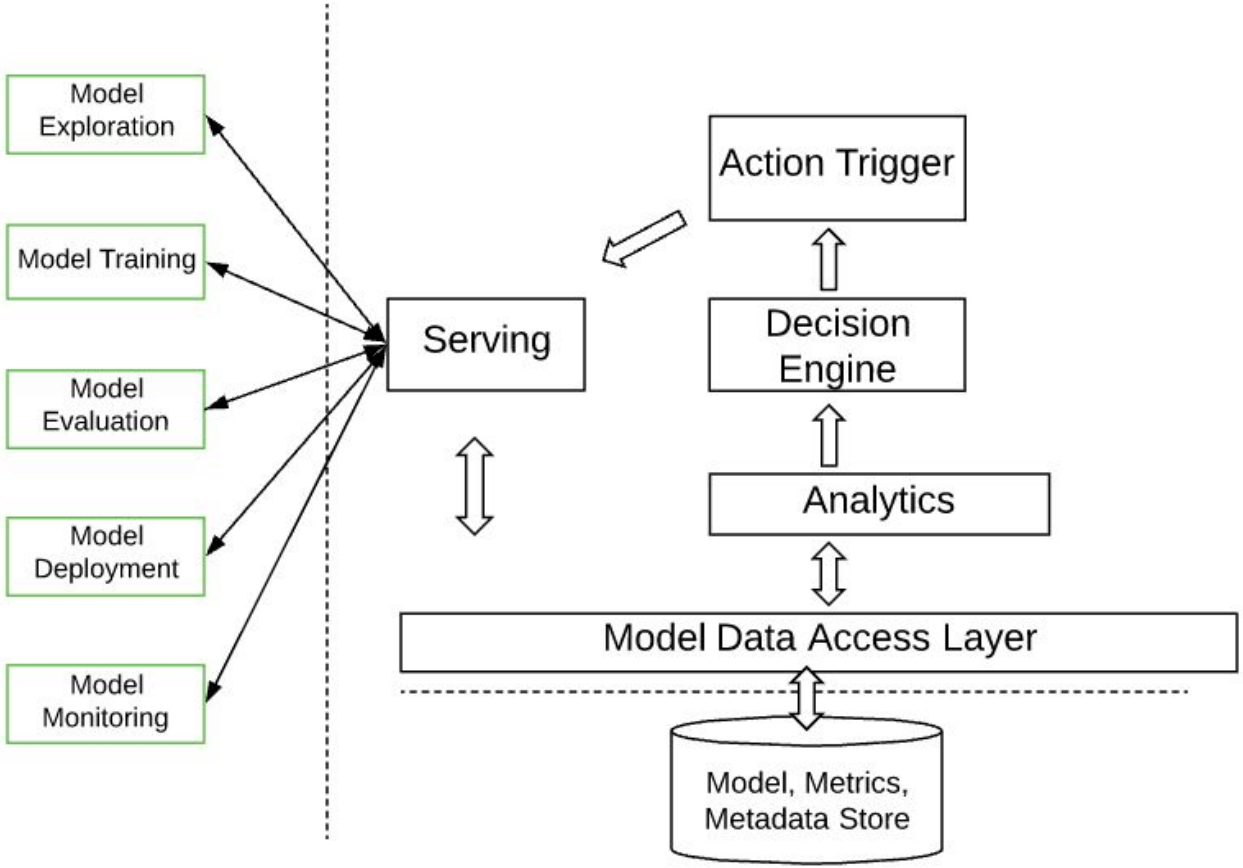
- Immutable Models
- Model Neutral
- Flexible
- Automated Dynamic Orchestration

MLMS Architecture

MLMS Architecture



MLMS Architecture



MLMS Architecture


```
Given: model_name == "linear model" and geo_id == "1"
When: model_status =="EVALUATED" and model_accuracy > 0.95
Then:
{
      "type": "HTTP",
      "endpoint": "http://localhost:12345/my-service/deploy",
      "http": {
            "method": "POST",
            "header": {
                  "Content-type": "application/json"
            }
            "body": {
                  "name": <model_name>,
                  "version": <model_version>,
                  "cityId": <geo_id>,
                  "accuracy": <model_accuracy>
            }
      }
}
```

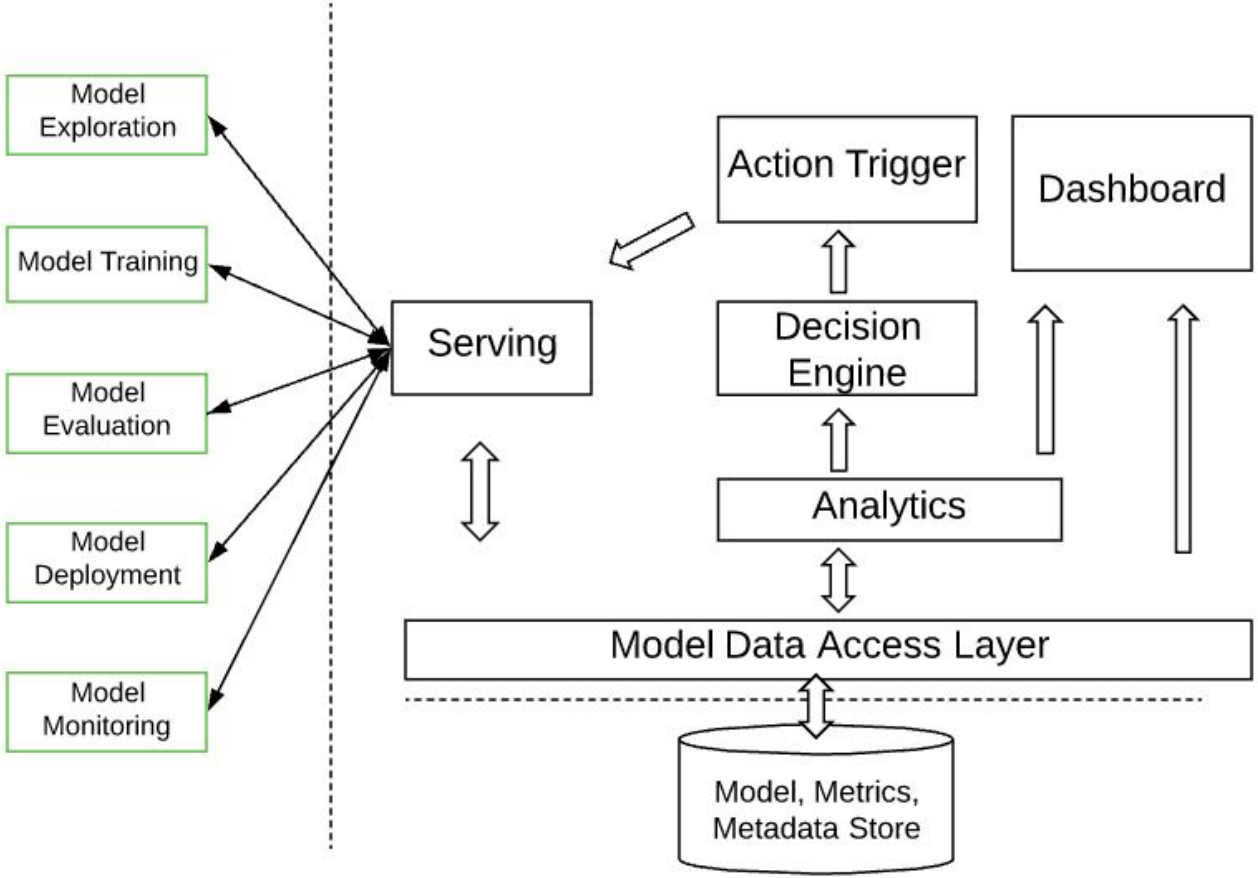
MLMS Architecture


```
Given: model_name == ”linear_model” and geo_id == “1”
When: model status ==”EVALUATED” and model accuracy > 0.95
Then:
{
      “type”: “HTTP”,
      “endpoint”: “http://localhost:12345/my-service/deploy”,
      “http”: {
            “method”: “POST”,
            “header”: {
                  “Content-type”: “application/json”
            }
            “body”: {
                  “name”: <model_name>,
                  “version”: <model_version>,
                  “cityId”: <geo_id>,
                  “accuracy”: <model_accuracy>
            }
      }
}
```

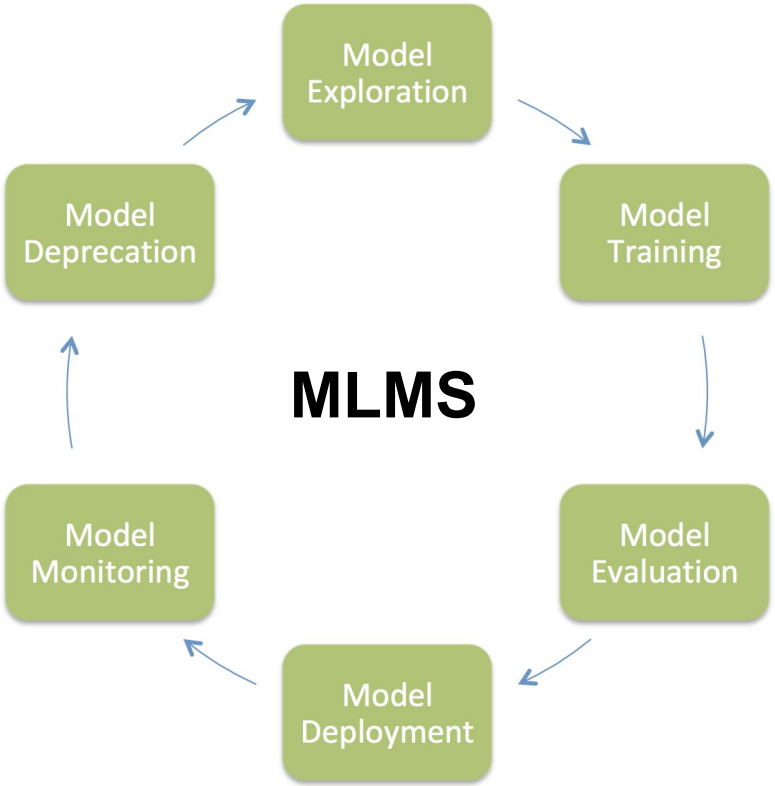
MLMS Architecture


```
Given: model_name == ”linear_model” and geo_id == “1”
When: model_status ==”EVALUATED” and model_accuracy > 0.95
Then:
{
      “type”: “HTTP”,
      “endpoint”: “http://localhost:12345/my-service/deploy”,
      “http”: {
            “method”: “POST”,
            “header”: {
                  “Content-type”: “application/json”
            }
            “body”: {
                  “name”: <model_name>,
                  “version”: <model_version>,
                  “cityId”: <geo_id>,
                  “accuracy”: <model_accuracy>
            }
      }
}
```

MLMS Architecture

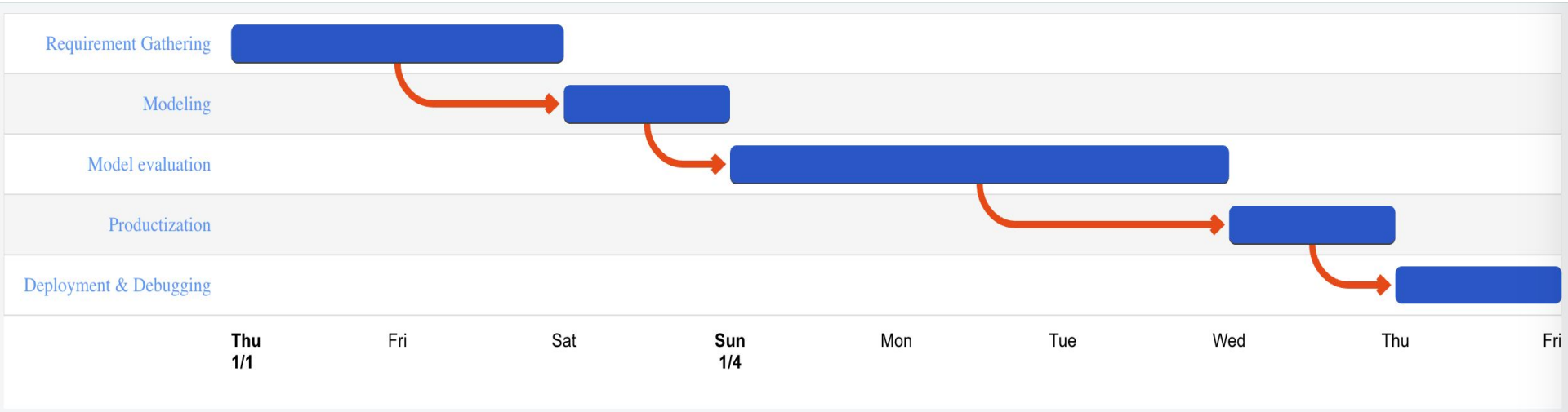


Machine Learning Model Lifecycle

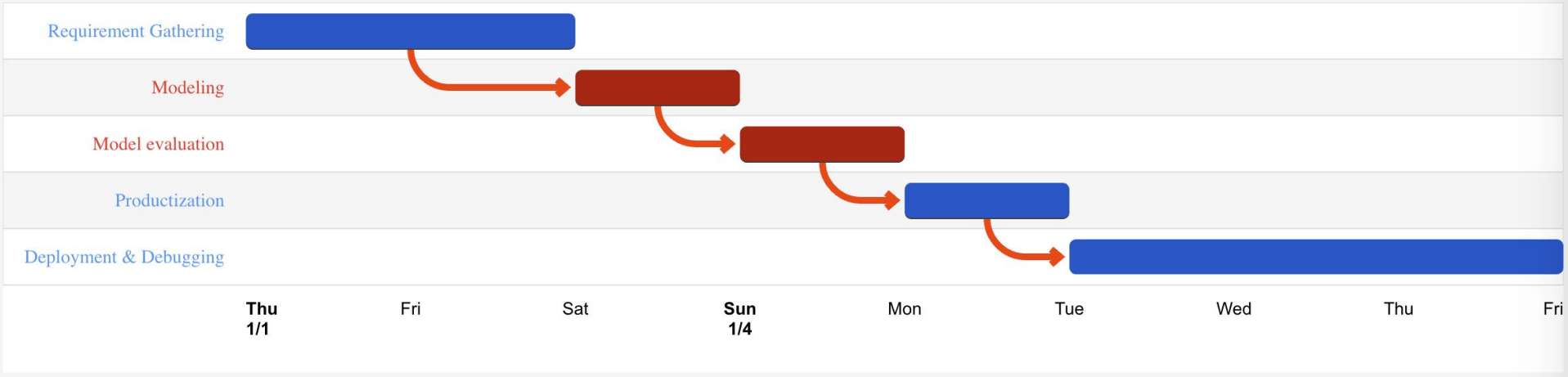


Data Science and Engineering Work Flow

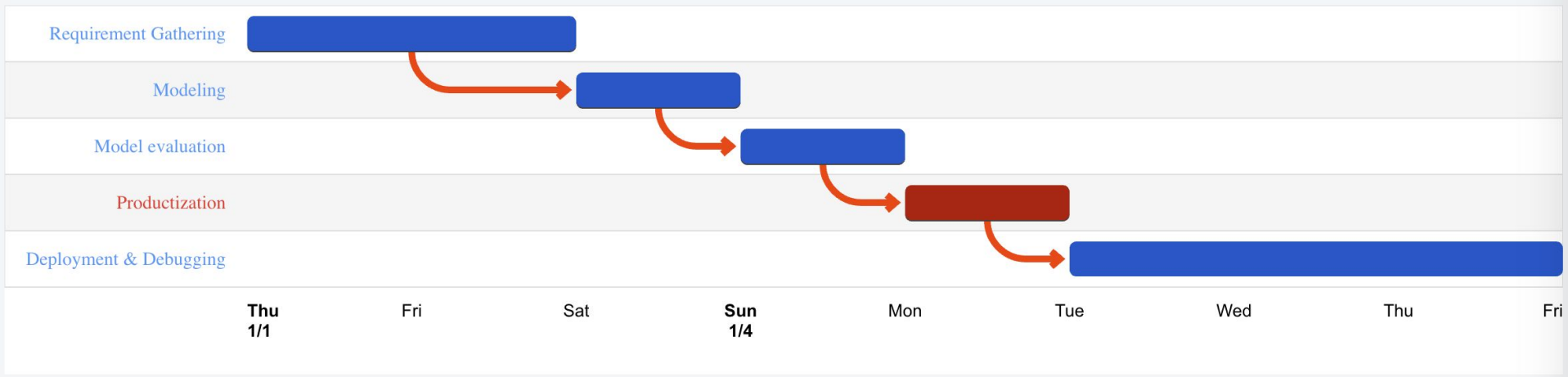
Data Scientists And Engineers Work In Lock Steps



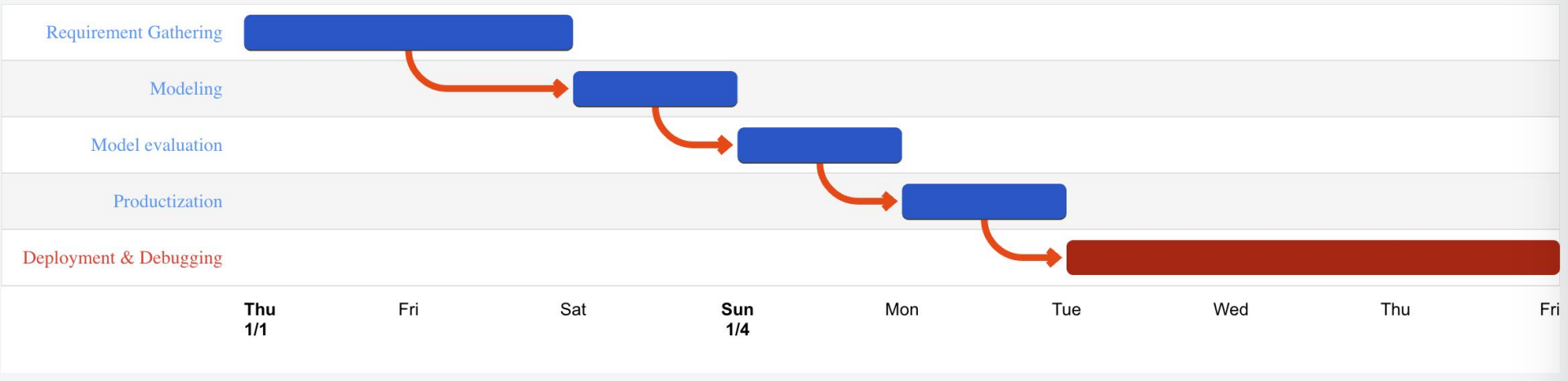
Engineers Are Blocked Before Modeling Is Done



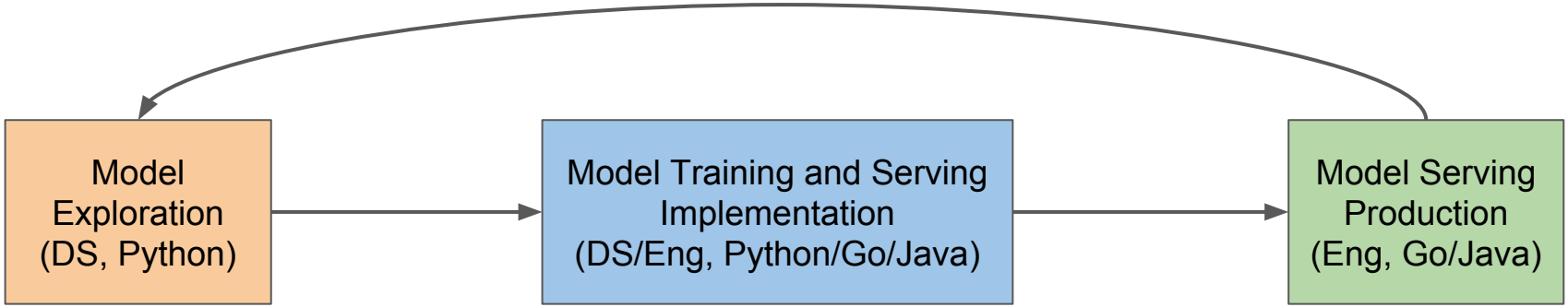
Time For Productization Is Often Squeezed



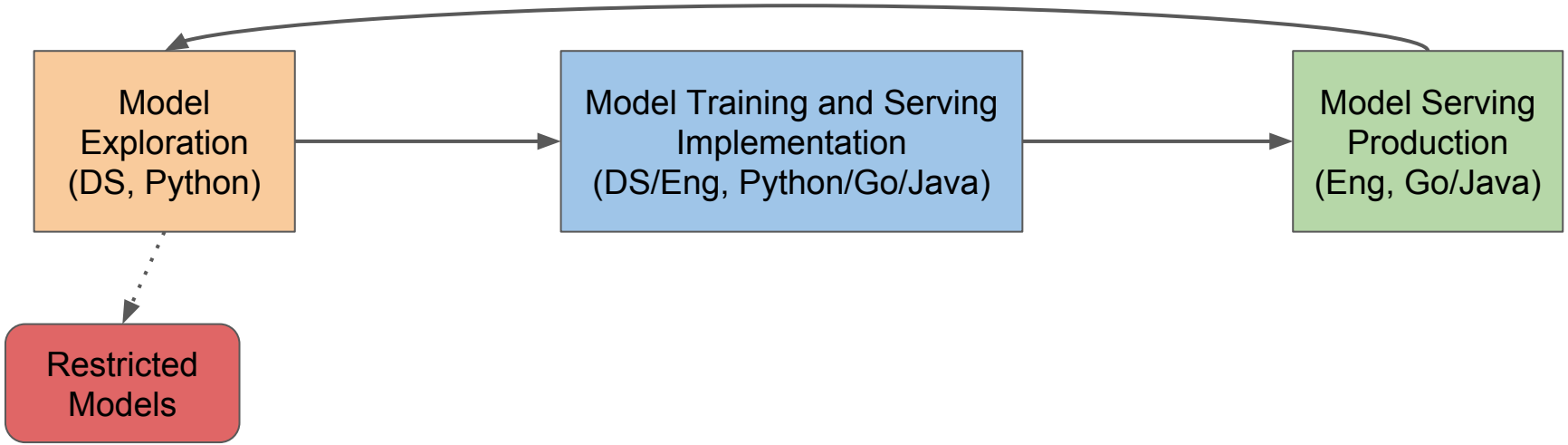
Rolling Out To All Cities Are Slow And Painful



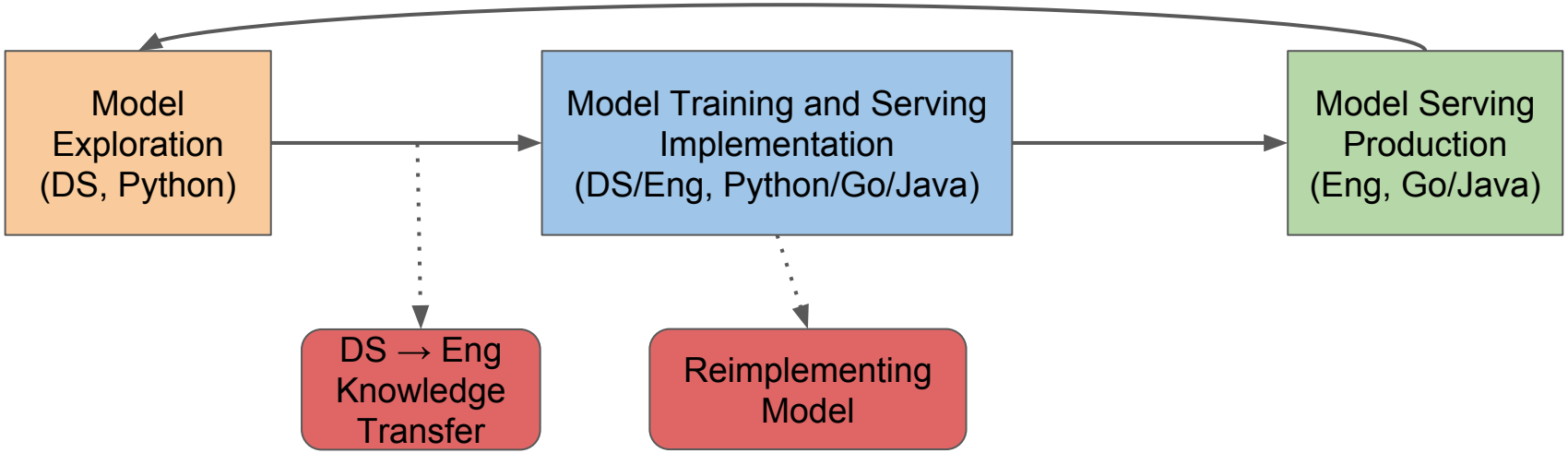
Analysis of Bottlenecks



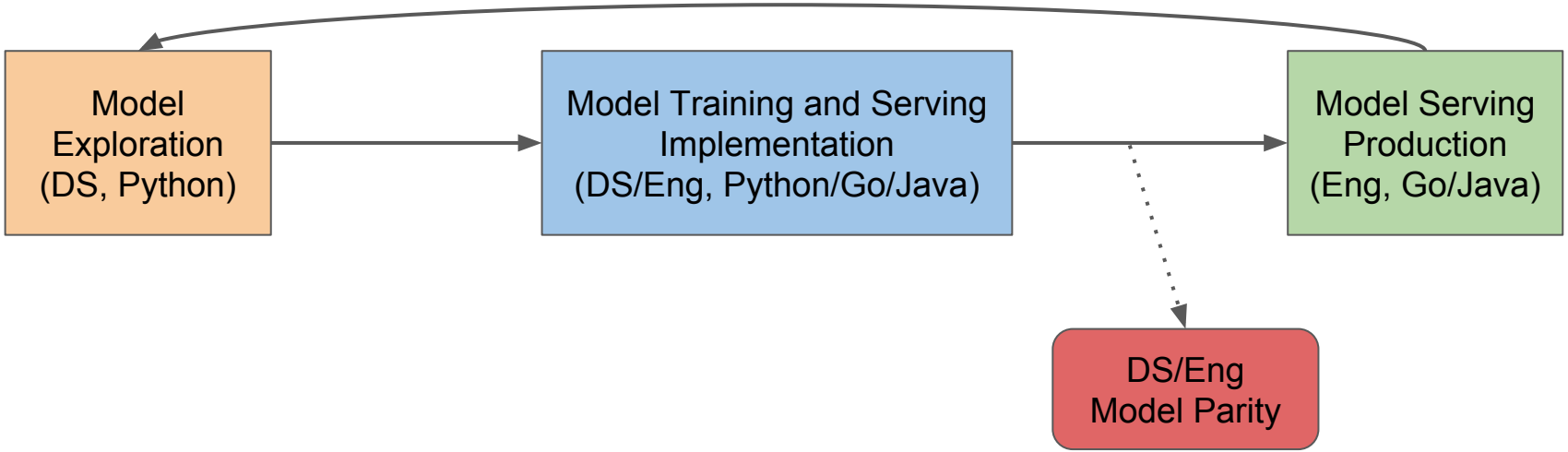
Analysis of Bottlenecks



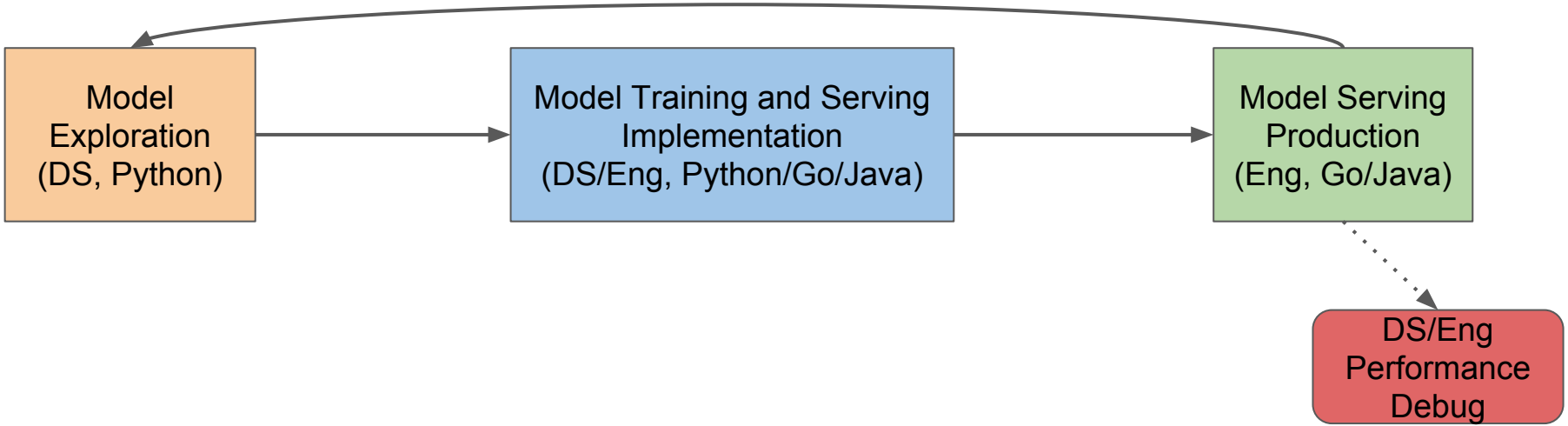
Analysis of Bottlenecks



Analysis of Bottlenecks



Analysis of Bottlenecks



Key Insight: Can We All Enjoy One ML Ecosystem?

Unified Framework → Many Benefits

- Standardized project structure
- Out-of-box support of local and remote deployment
- Reusable algorithms and framework
- Design review between engineer and DS
- Code review between engineer and DS
- Who codes, who debugs

A good framework for us

A flexible generic framework with no restriction on model building

A good framework for us

A flexible generic framework with no restriction on model building

A unified framework for model development and production

A good framework for us

A flexible generic framework with no restriction on model building

A unified framework for model development and production

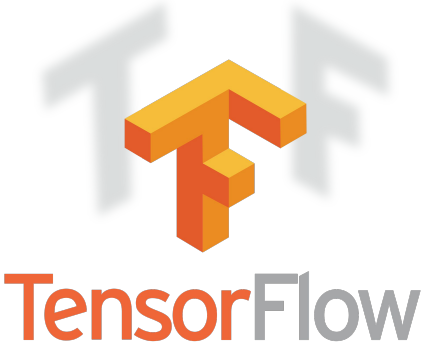
A production proven and well maintained framework easy to adopt

A good framework for us

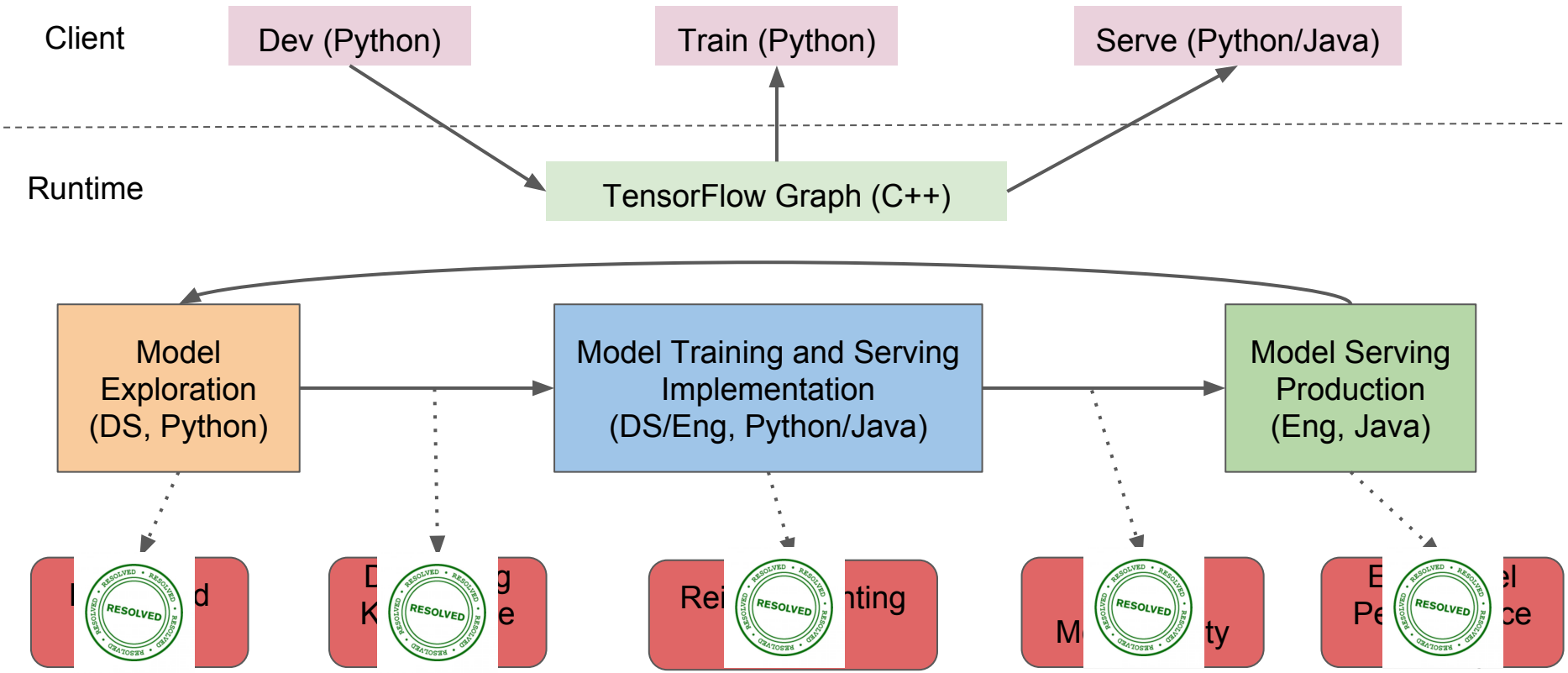
A flexible generic framework with no restriction on model building

A unified framework for model development and production

A production proven and well maintained framework easy to adopt



TensorFlow

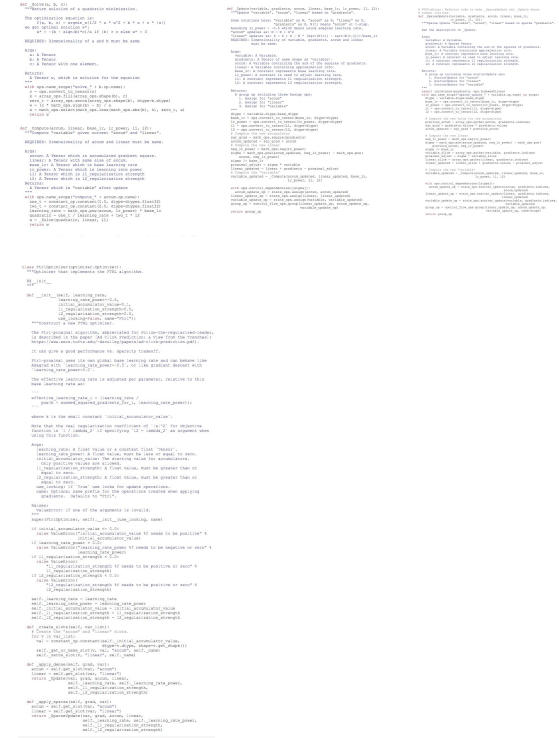

Client
Dev (Python)
Train (Python)
Serve (Python/Java)
Runtime
TensorFlow Graph (C++)
Model Exploration (DS, Python)
Model Training and Serving Implementation (DS/Eng, Python/Java)
Model Serving Production (Eng, Java)
RESOLVED

Enable DS to Write Production-Ready Code

- Tensorflow
  - Efficient core
  - DS-friendly API
- Engineers focusing on optimization and automation
  - Parallelization of algorithms
  - End-to-end automation
  - Visualization
  - Integration
  - Project scaffolding

Example

Build your own FTRL



Use a framework

```
# With TFLearn estimators.
ftrl = Ftrl(learning_rate=0.01, learning_rate_power=-0.1)
regression = regression(net, optimizer=ftrl)

# Without TFLearn estimators (returns tf.Optimizer).
ftrl = Ftrl(learning_rate=0.01).get_tensor()
```

Building Tools

- Model Lifecycle Management System
- Hyperparameter Tuning
- Horovod for Distributed TensorFlow Training

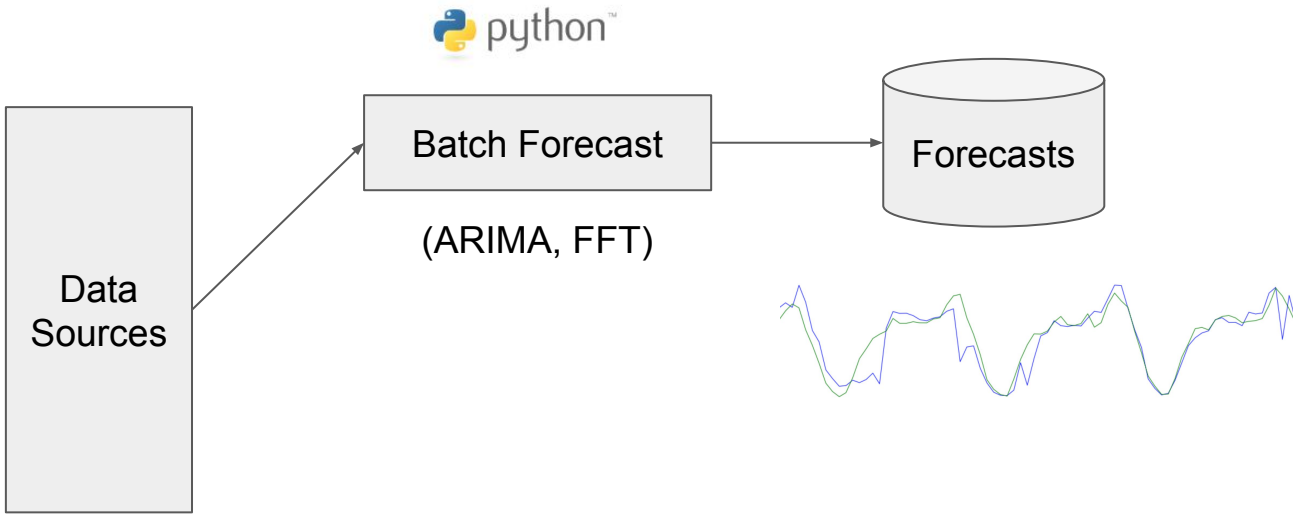
Conclusion

- A fully automated MLMS is key to the success of complex ML systems
- A single framework for DS and engineers boosts productivity
- Building great tools is crucial to ML projects

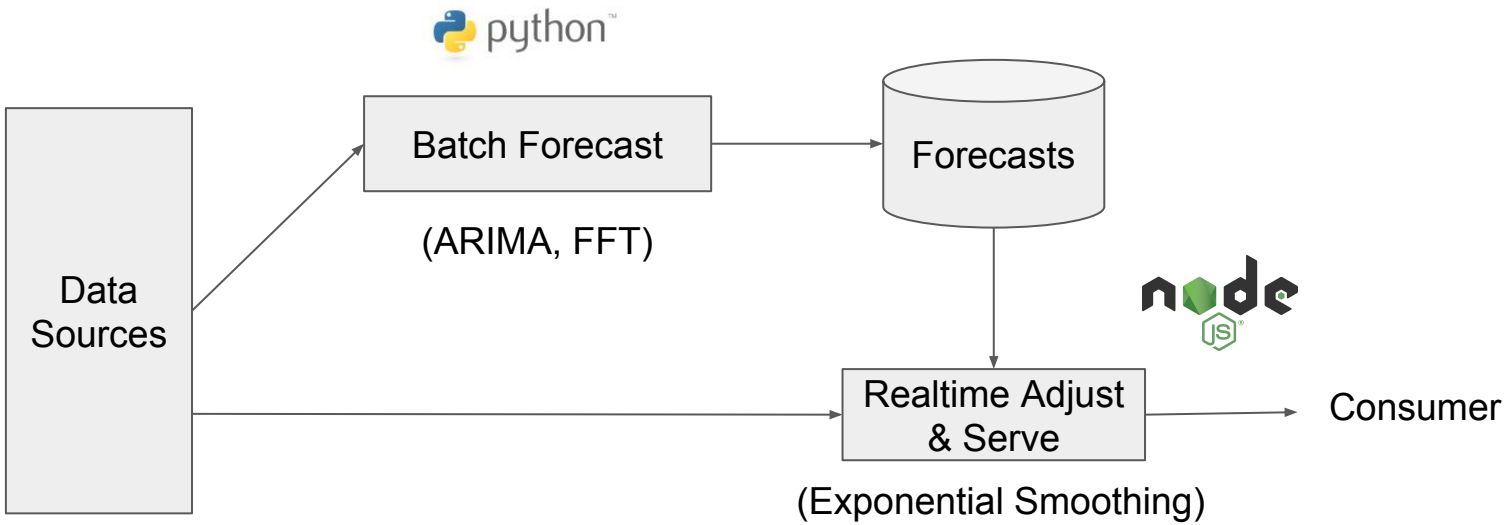
Q & A

How do we make the forecasts?

Batch forecasting (2015)



Batch forecasting + Real-time Adjustment



Issues Observed

Not many ML libraries for Node.js

Real-time component (Node.js) can not support CPU intensive computation

Can not handle large scale data features in real-time

Can not share code for batch and online processing

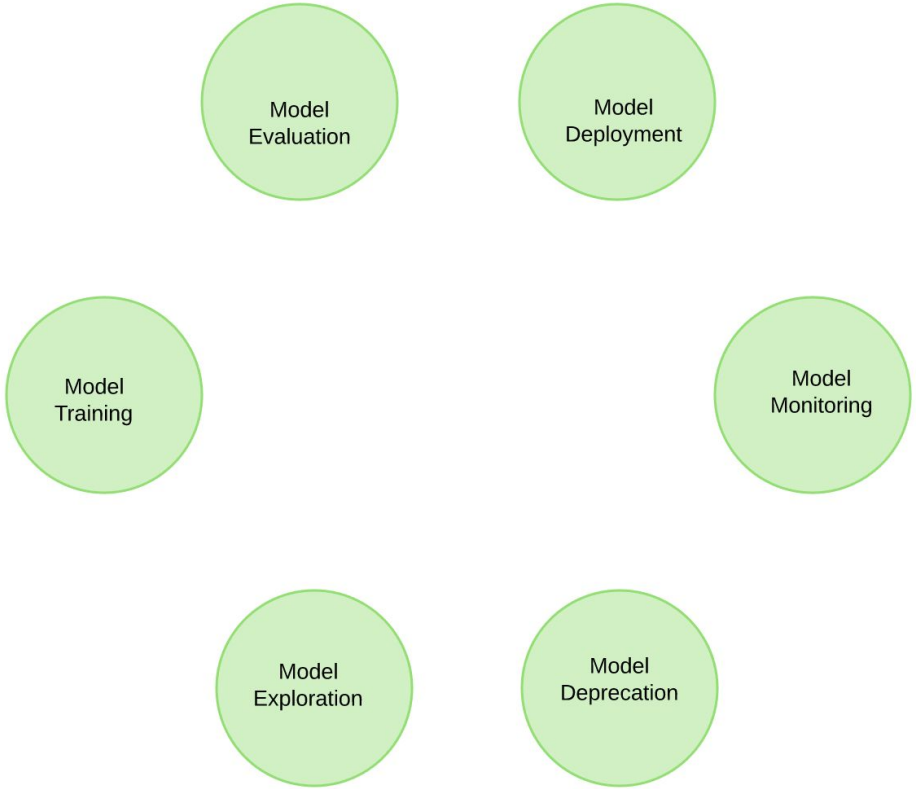
Second Generation of Forecasting Engine

(Inspired by DataFlow and TensorFlow)

Some interesting design principles:

Both realtime and batch prediction: prediction is minute level, backtesting/evaluation requires batch processing

Machine Learning Model Lifecycle

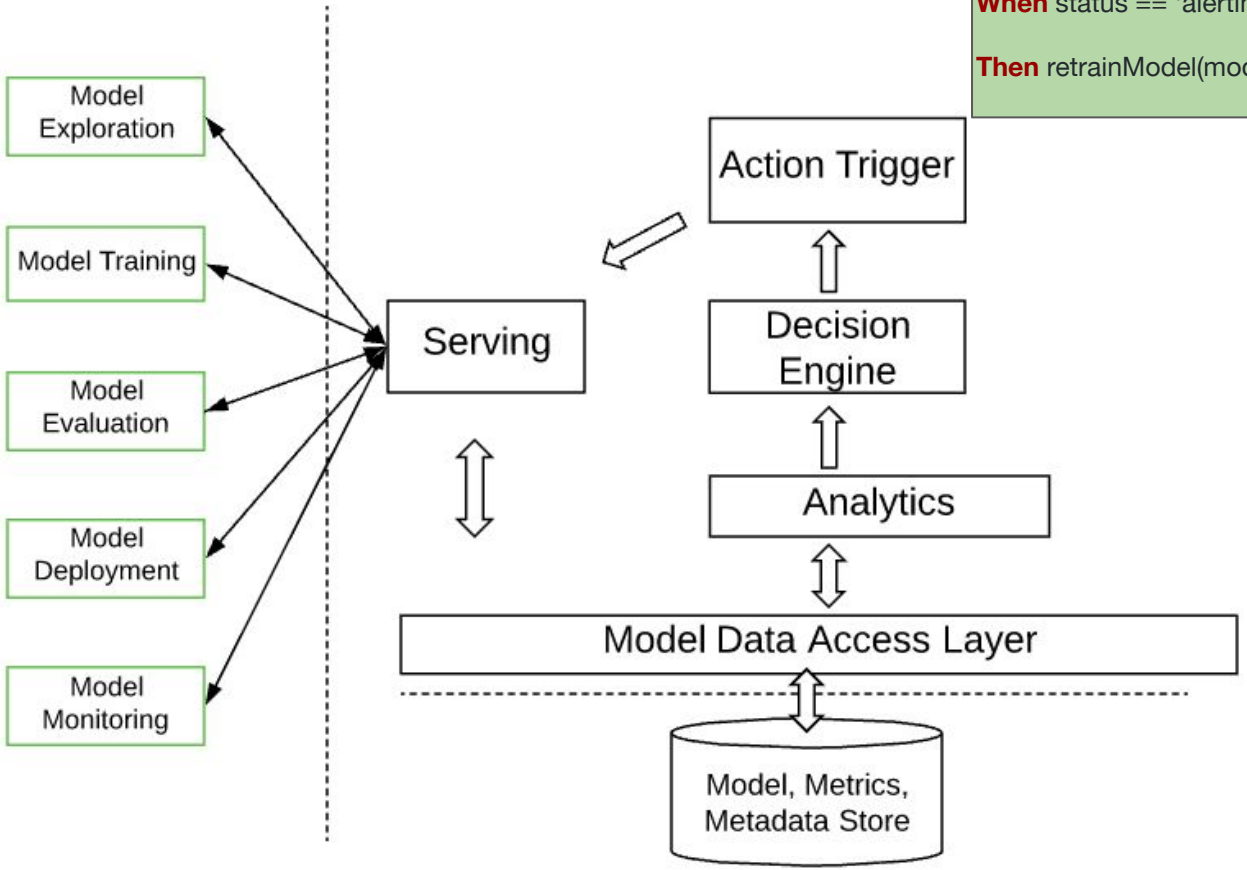


MLMS Architecture

Given model_name=linear_demand_model and city_id=1

When status == 'alerting' and time_sustained > 3 days

Then retrainModel(model_name, city_id, model_version)



Issues?

DS and engineers use different languages

Development and production serving use different tech stacks