Scala @ Uber Data Science

Nick Jones, Uber Data Science
@nick.jones

UBER

Introduction

- Originally from Seattle, in the US
- Worked at Uber for 3 years in San Francisco and Amsterdam
- Before that worked at a firm that did expert witness testimony in legal cases involving statistics and machine learning

AGENDA

- Motivating Example
- Scala in Experimentation
- Scala in Machine Learning
- sparkmagic

AGENDA

- **Motivating Example**
- Scala in Experimentation
- Scala in Machine Learning
- sparkmagic

Motivating Example: Letting riders pay for Uber with cash

- **Why cash?**

Motivating Example

- **Why cash?**
- Only supporting credit cards made sense in early markets.

That all changed when we decided to invest heavily in India and Latin America

Motivating Example

- **...so why not cash from the beginning?**
- Not “magical”
  - Can’t just get out
  - Change
  - Split a fare?
  - Service fee?

Motivating Example

- **Given costs, cash needs to have some series benefits**
- So how do we see how big the benefits are?

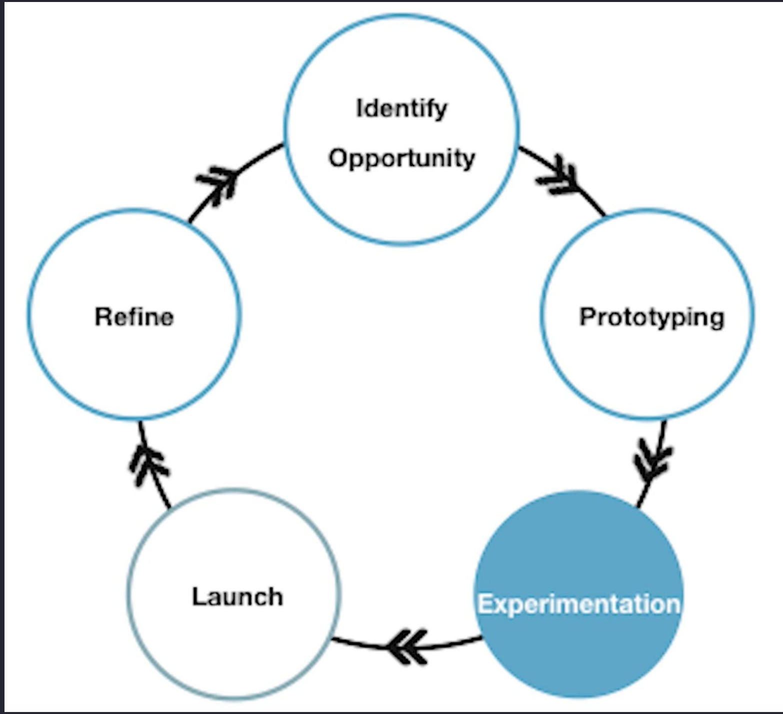
Experimentation

AGENDA

- Motivating Example
- **Scala in Experimentation**
- Scala in Machine Learning
- sparkmagic

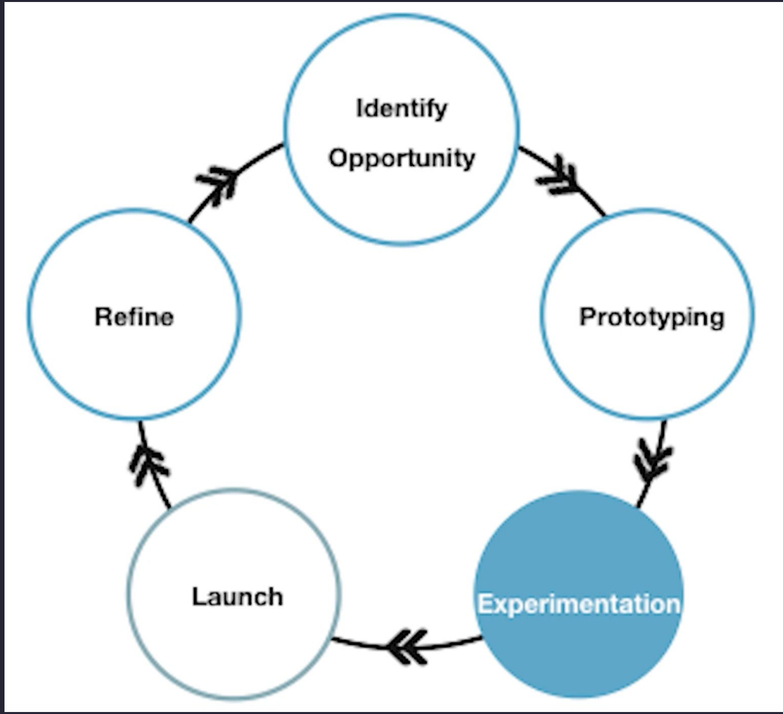
Uber Experimentation Platform

- **Why experimentation?**
- Outside world often has a larger impact on user behavior/metrics than the things that you ship...
- ...so we need to have a control group



Uber Experimentation Platform

- **Why experimentation?**
- Keeping a control group also allows us to easily see if what we shipped is causing regressions in metrics (aka causing the app to crash).



Uber Experimentation Platform

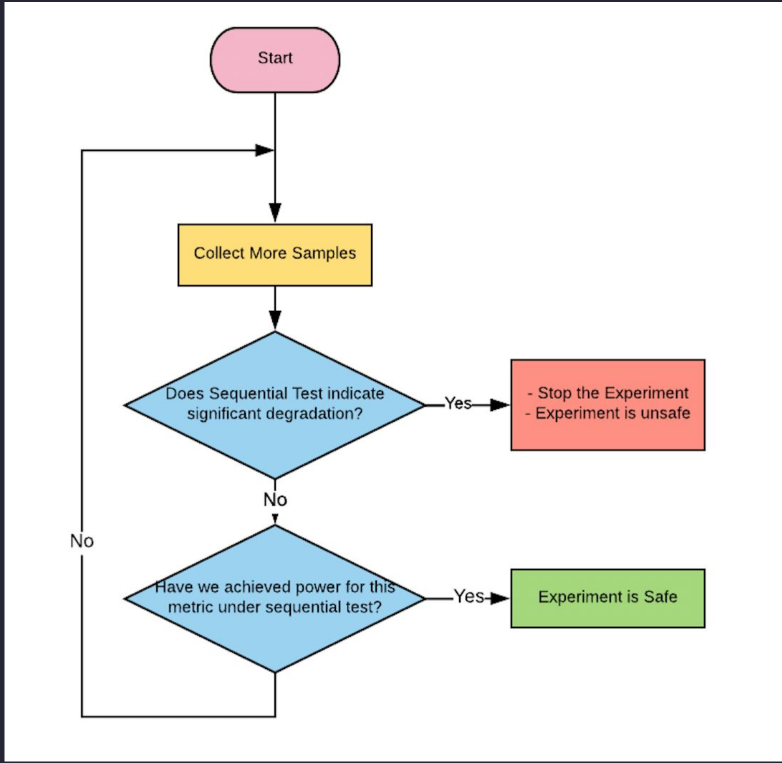
- **Staged rollout**
- Roll feature out to a small group of users first, then to more and more, performing tests along the way to ensure we don't have regressions.

Differences Between a Staged Rollout and Standard A/B Testing

	Staged Rollout	Standard A/B Testing
Type of Process	Reliable feature release	Feature evaluation
Goal	Whether the feature is causing a regression	Whether the feature is successful
Experiment Design and Configuration	Multiple adaptive stages	One fixed stage
Metrics	Core app health and business metrics	Complete set of app metrics
Statistical Test	Sequential test	Fixed horizon test (e.g. t-test, chi-squared test, and bootstrapping)

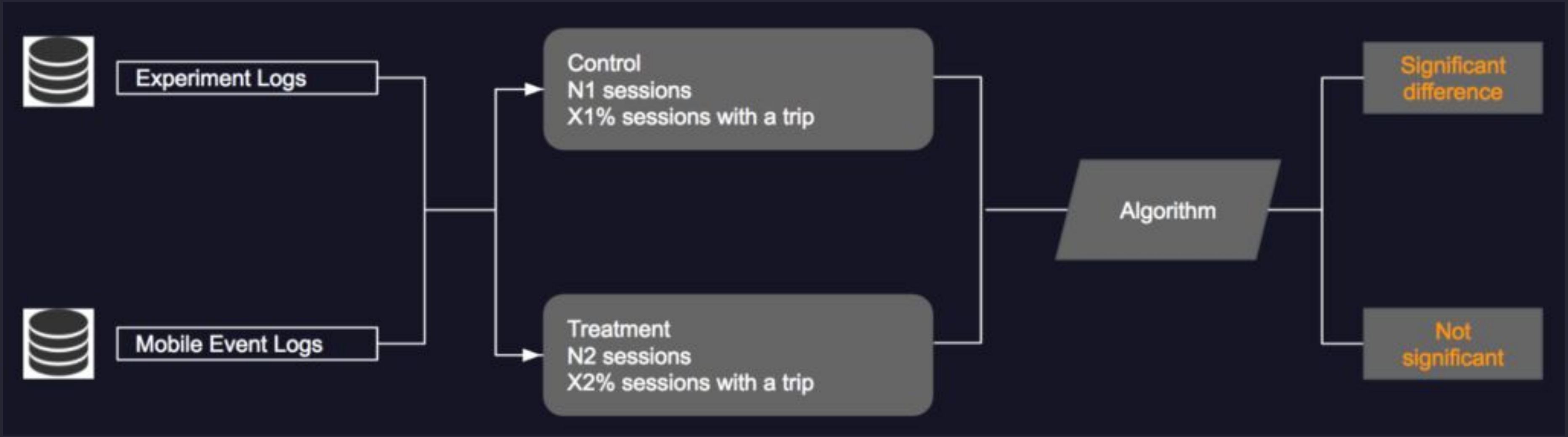
Uber Experimentation Platform

- **Staged rollout: How it works**
- Continue collecting samples until we have high confidence that it is not causing a regression, then roll it out to more users.



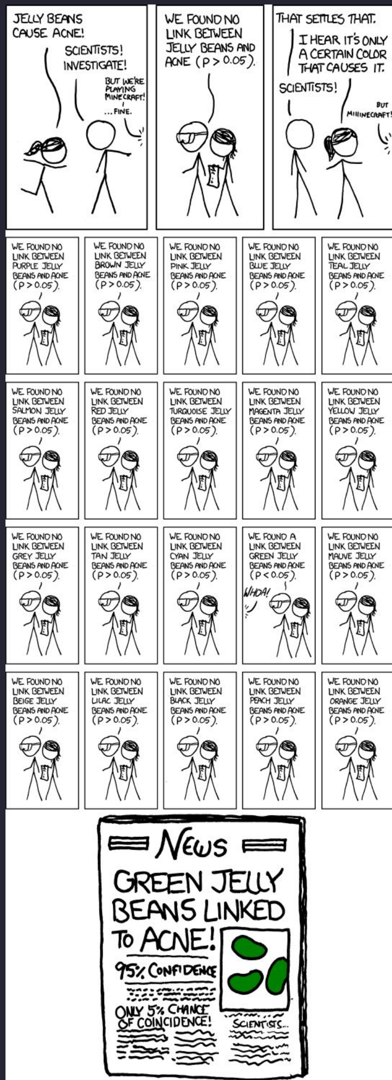
Uber Experimentation Platform

- Staged rollout architecture, example



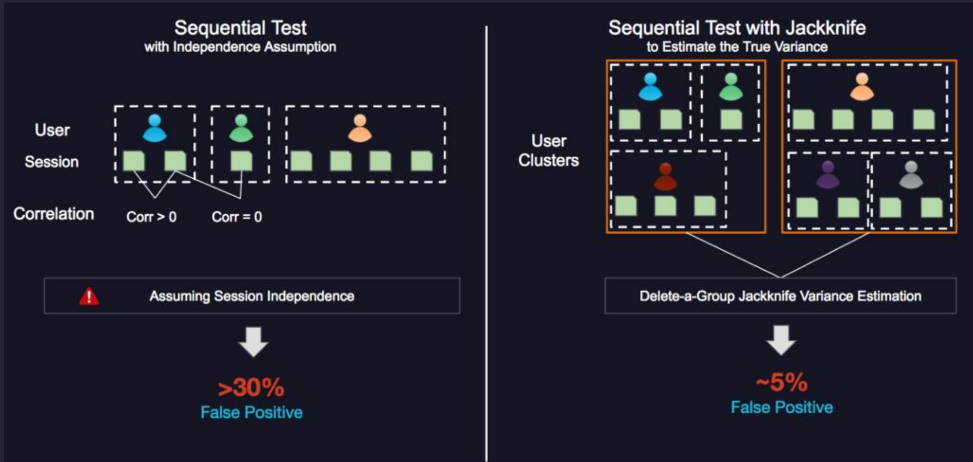
Uber Experimentation Platform

- **Staged rollout: How do we test?**
- Initially, classical statistical tests had large false positive rate.
- Then we tried sequential likelihood ratio tests (SLRT), assuming that users' sessions were independent. We still had a high false positive rate



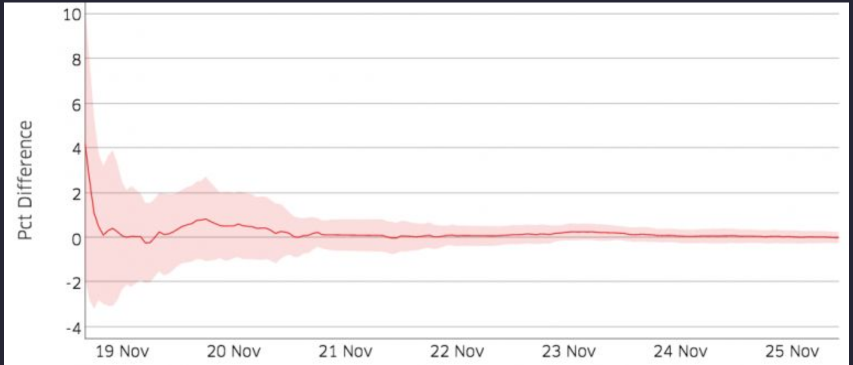
Uber Experimentation Platform

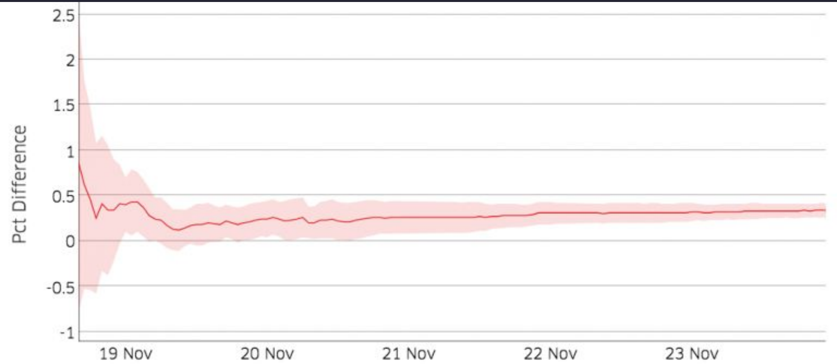
- **Staged rollout: How do we test?**
- We then kept the SLRT, but used delete-a-group jackknife variance estimation, which worked great…
- ...but now we have to do simulation-based estimation for every experiment we launch, all the time
- Scala handles this well



Uber Experimentation Platform

- Staged rollout testing, example



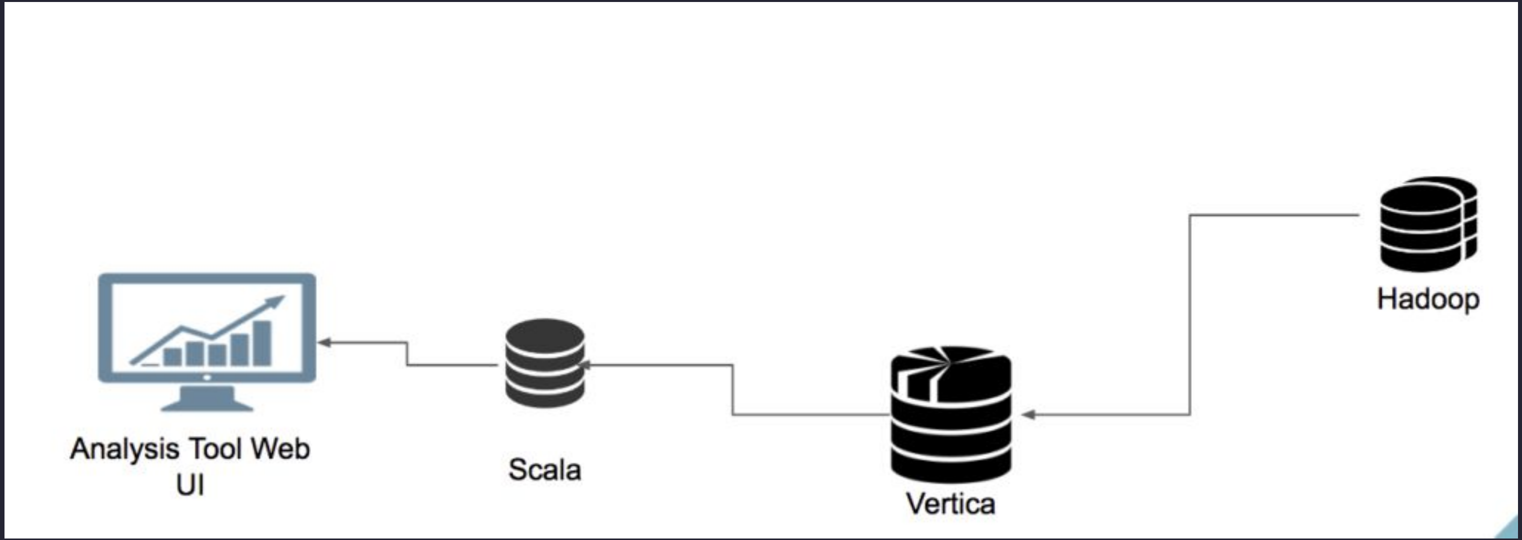


Uber Experimentation Platform

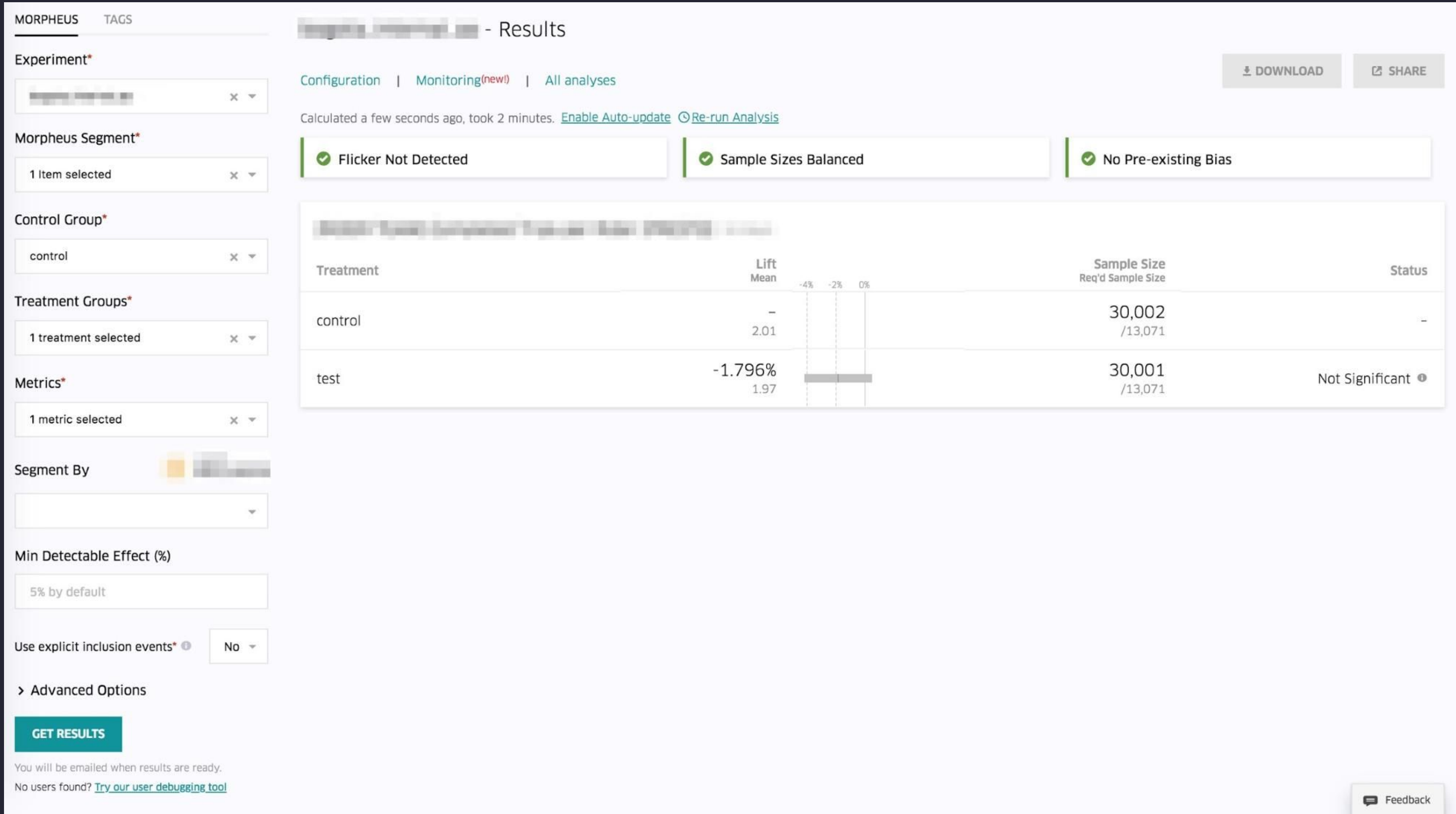
- **So, how do we know if what we shipped was successful?**
- Success metrics -> Statistical tests
- Used to be “by hand”
- Took lots of time, teams blocked on data science resources,

Uber Experimentation Platform

- **So, how do we know if what we shipped was successful?**
- With the statistics engine and web UI, it takes minutes, not days

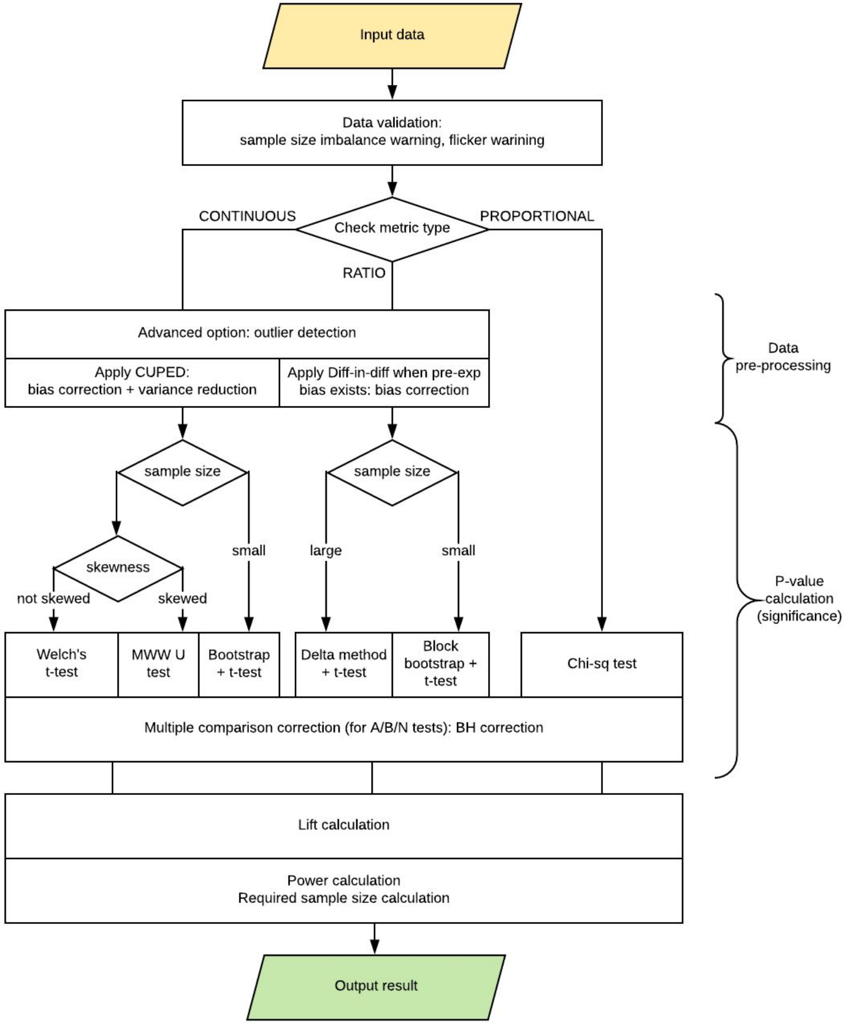


Uber Experimentation Platform



Uber Experimentation Platform

- **So, what is the statistics engine doing?**
- Quite a lot actually, and very quickly, and on-demand, which is why Scala is a great choice



Motivating Example

- So, was cash successful?

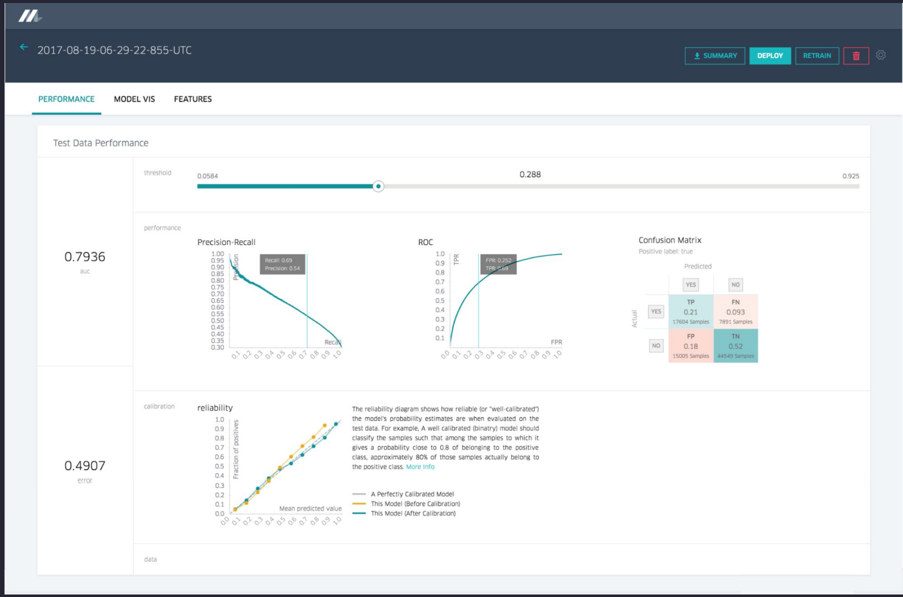
Motivating Example

- **So, was cash successful?**
- Yes, in fact so successful that we defaulted everyone to cash in some markets, so it made sign up more frictionless.

Motivating Example

- But, if a rider has a credit/debit card we would still want them to use it.
- So how can we intelligently decide who we should default to cash, and who we should ask for credit card info?

Michelangelo

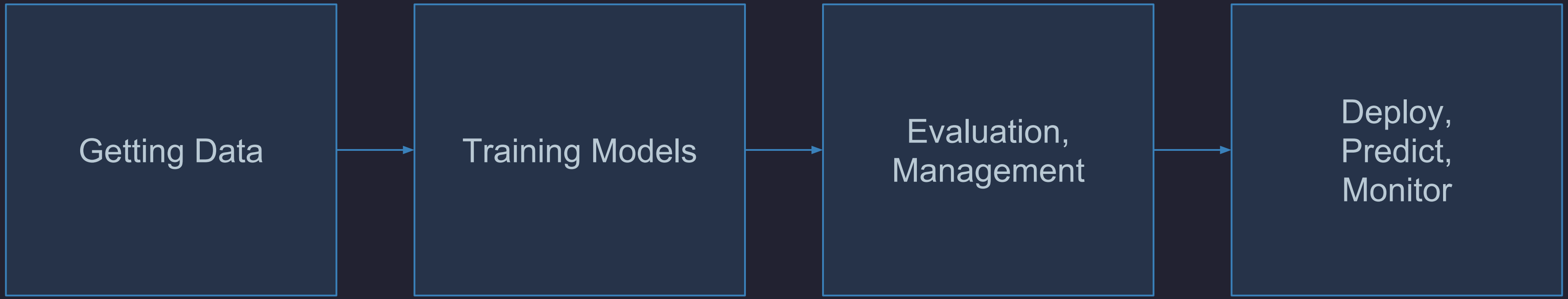


AGENDA

- Motivating Example
- Scala in Experimentation
- **Scala in Machine Learning**
- sparkmagic

Rewind to 2015

Traditional ML Lifecycle



Uber Machine Learning Platform: Early Challenges

Fragmentation

- Multiple systems for each part of life cycle
  - Built for team-specific use cases
- Non-overlapping feature sets
  - Hard-to-maintain systems
  - Increased cognitive burden

Uber Machine Learning Platform: Early Challenges

Limited Scale

- Data modelling used to be done with mostly Python, R
- Great for prototypes, but these models did not scale
- Not-so-great integration with existing big-data ecosystem at the time

Uber Machine Learning Platform: Early Challenges

Non-reproducible models

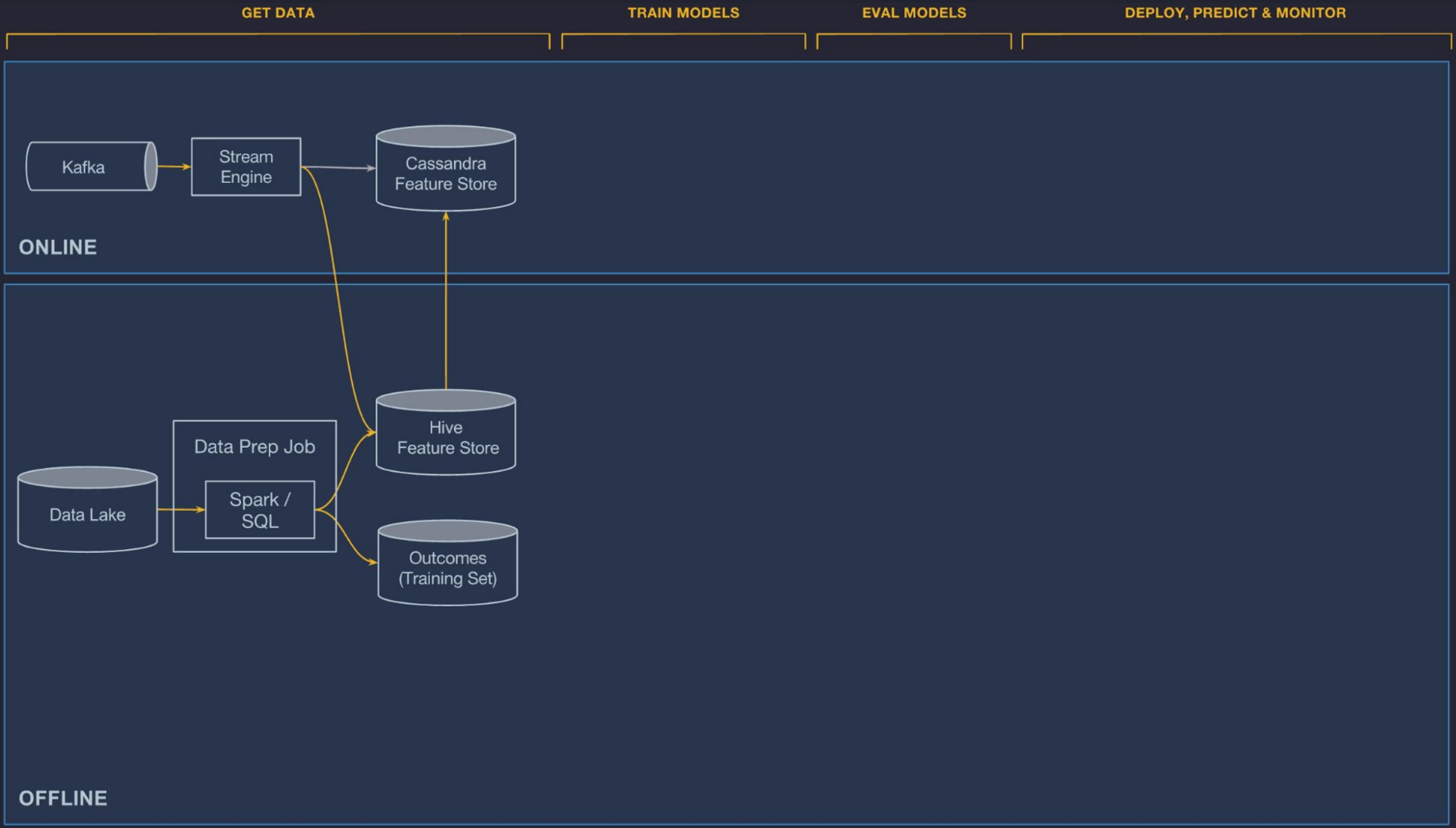
- Models would be trained without any way to retrain
- Comparisons were hard on new data

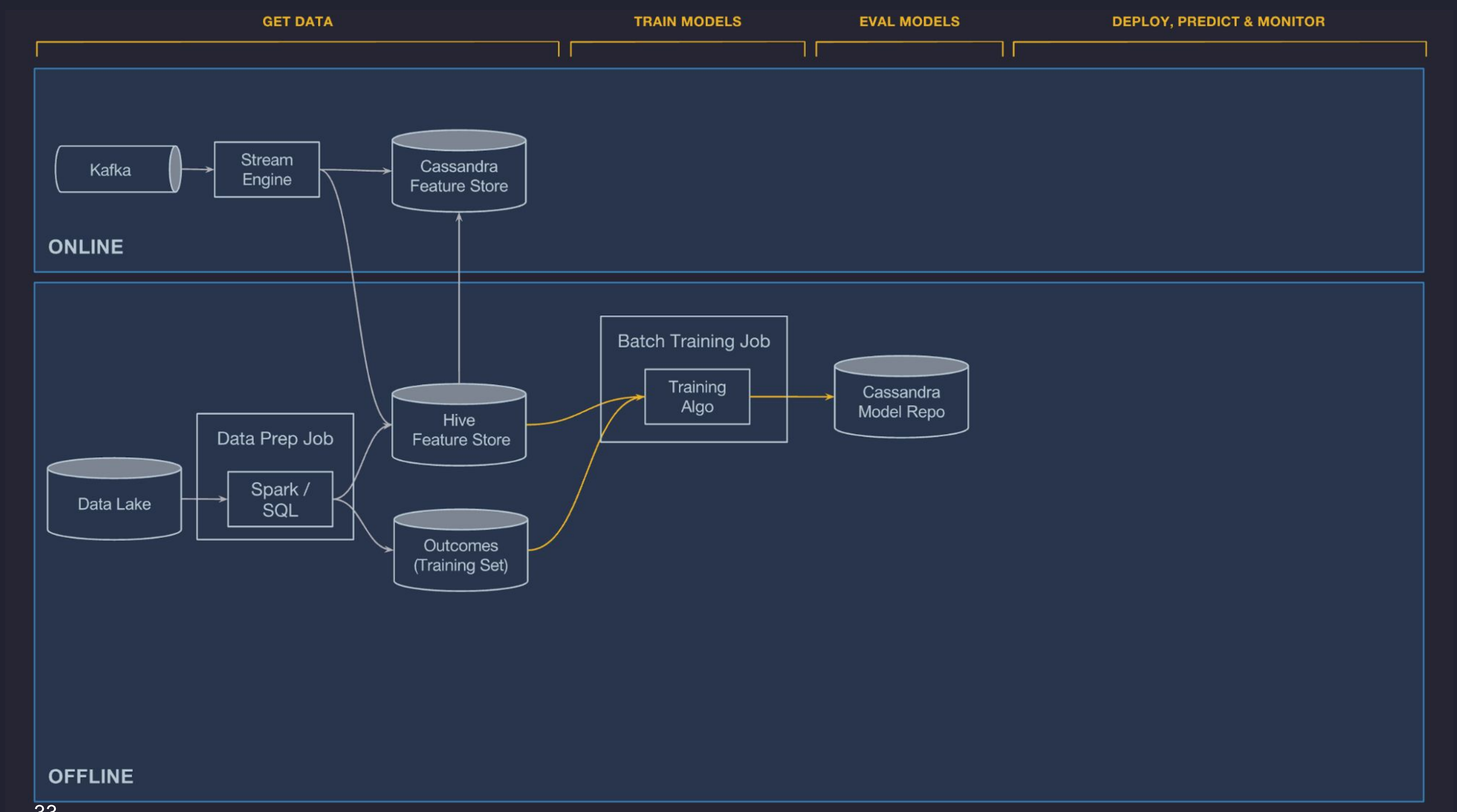
Uber Machine Learning Platform: Michelangelo

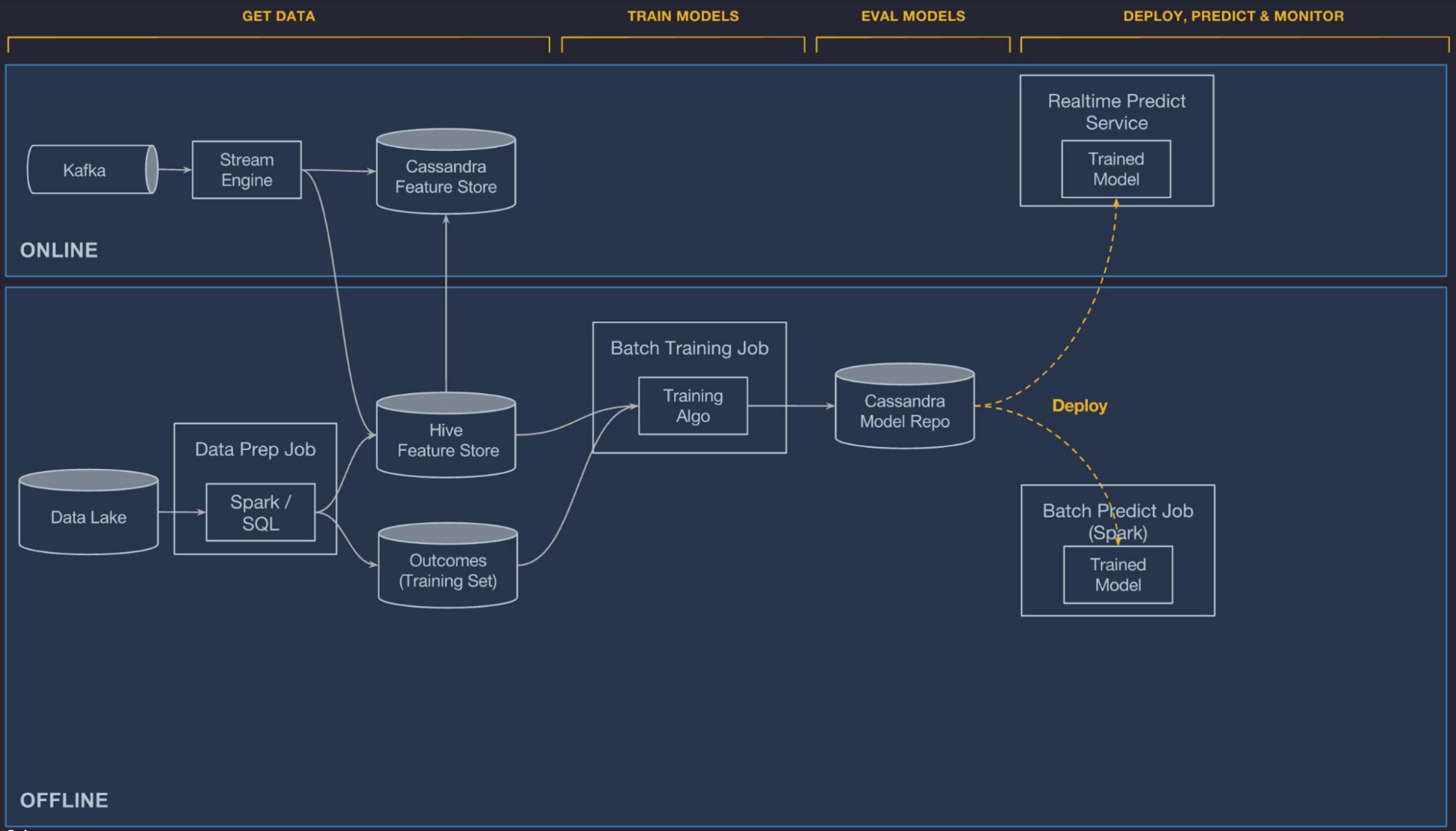
Michelangelo: self-service platform for Uber teams to build, train, and deploy ML models, based on Spark and MLlib

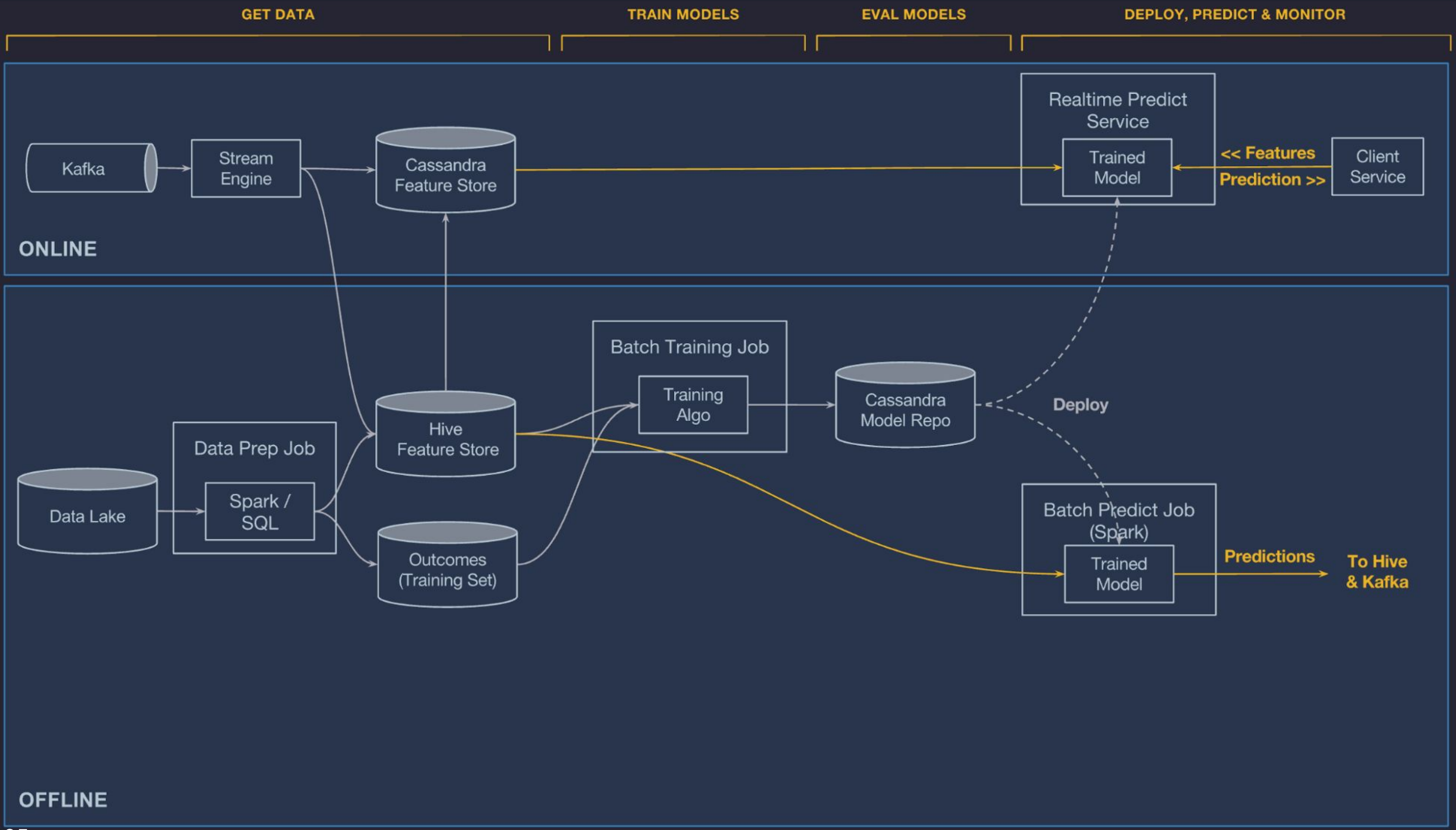
Michelangelo aims to provide:

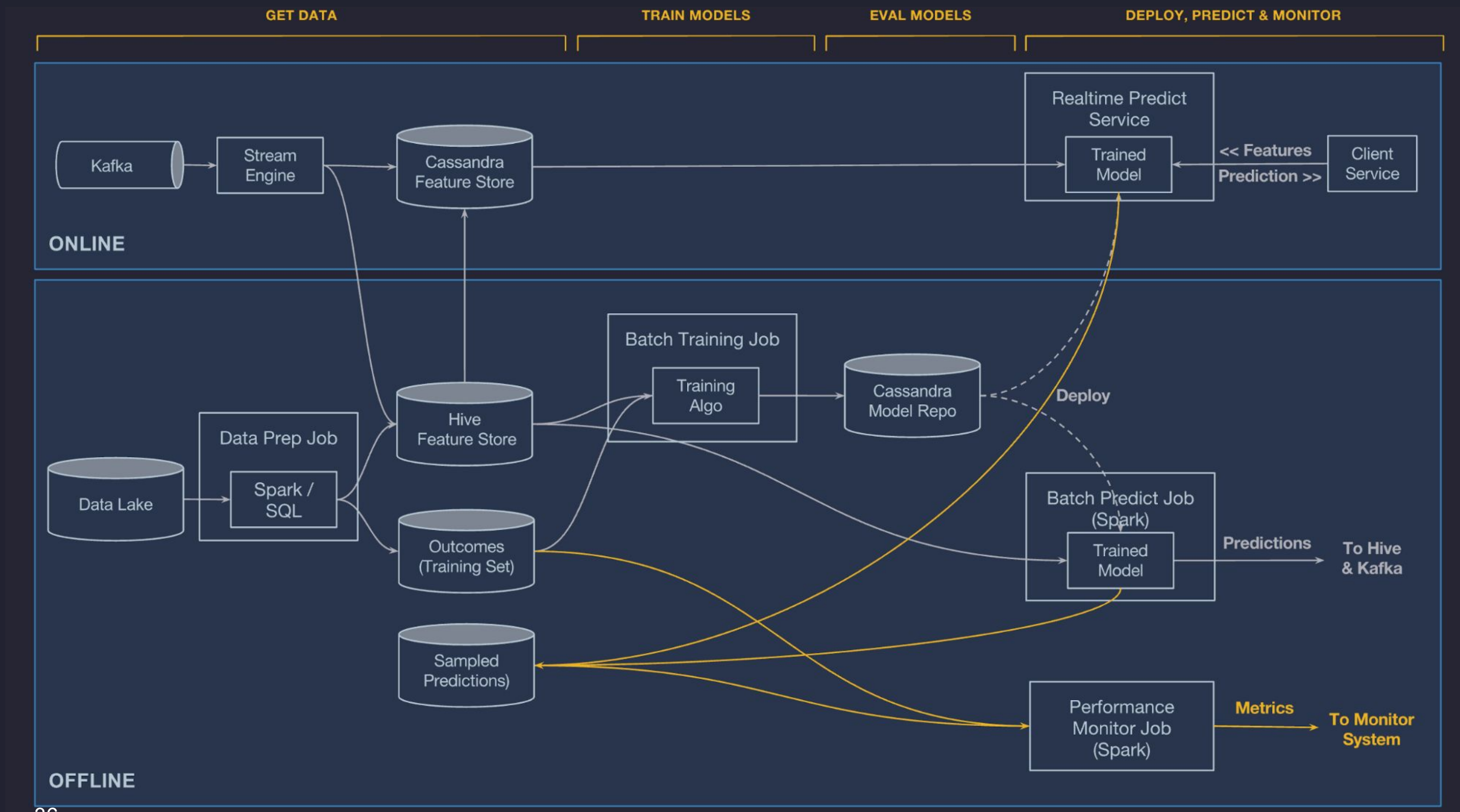
- Model training for very large datasets
- Centralized feature store
- Model analysis/comparison tooling
- Real-time predictions

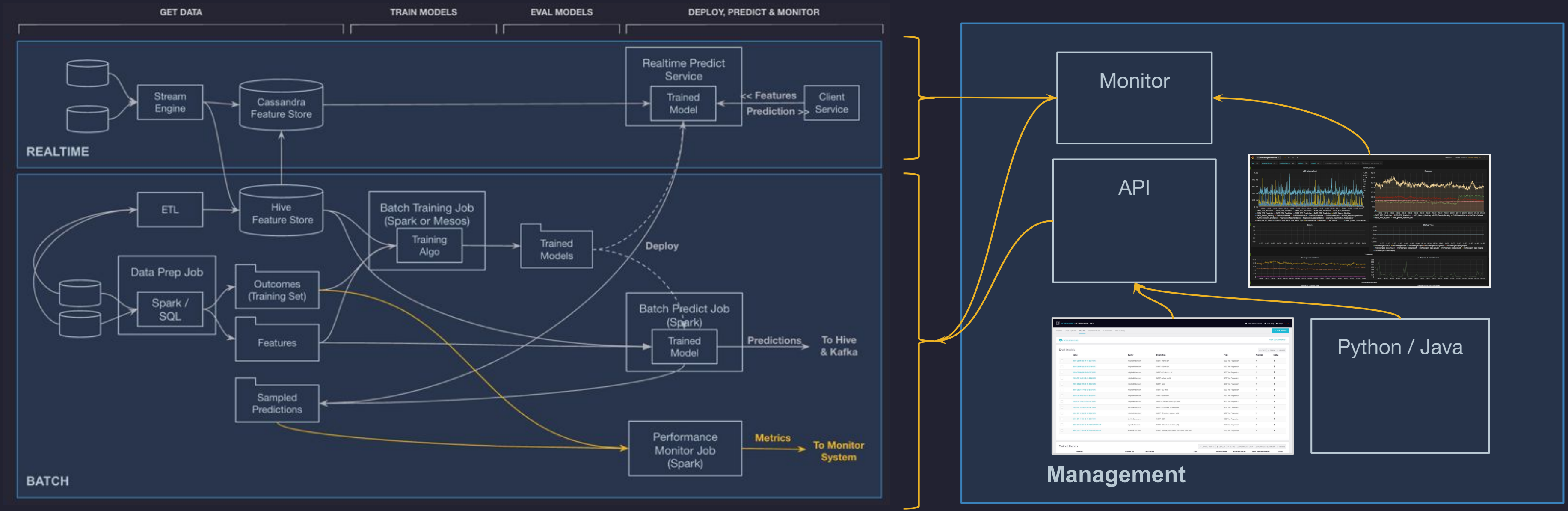

GET DATA
TRAIN MODELS
EVAL MODELS
DEPLOY, PREDICT & MONITOR
Kafka
Stream Engine
Cassandra Feature Store
ONLINE
Data Prep Job
Spark / SQL
Data Lake
Hive Feature Store
Outcomes (Training Set)
OFFLINE


GET DATA
TRAIN MODELS
EVAL MODELS
DEPLOY, PREDICT & MONITOR
Kafka
Stream Engine
Cassandra Feature Store
ONLINE
Batch Training Job
Training Algo
Cassandra Model Repo
Hive Feature Store
Data Prep Job
Spark / SQL
Data Lake
Outcomes (Training Set)
OFFLINE


GET DATA
TRAIN MODELS
EVAL MODELS
DEPLOY, PREDICT & MONITOR
ONLINE
Kafka
Stream Engine
Cassandra Feature Store
Realtime Predict Service
Trained Model
OFFLINE
Data Lake
Data Prep Job
Spark / SQL
Hive Feature Store
Outcomes (Training Set)
Batch Training Job
Training Algo
Cassandra Model Repo
Deploy
Batch Predict Job (Spark)
Trained Model


GET DATA
TRAIN MODELS
EVAL MODELS
DEPLOY, PREDICT & MONITOR
Kafka
Stream Engine
Cassandra Feature Store
Realtime Predict Service
Trained Model
<< Features
Prediction >>
Client Service
ONLINE
Batch Training Job
Training Algo
Cassandra Model Repo
Deploy
Data Prep Job
Spark / SQL
Hive Feature Store
Data Lake
Outcomes (Training Set)
Batch Predict Job (Spark)
Trained Model
Predictions
To Hive & Kafka
OFFLINE


GET DATA
TRAIN MODELS
EVAL MODELS
DEPLOY, PREDICT & MONITOR
ONLINE
Kafka
Stream Engine
Cassandra Feature Store
Realtime Predict Service
Trained Model
<< Features
Prediction >>
Client Service
OFFLINE
Data Lake
Data Prep Job
Spark / SQL
Hive Feature Store
Outcomes (Training Set)
Sampled Predictions)
Batch Training Job
Training Algo
Cassandra Model Repo
Deploy
Batch Predict Job (Spark)
Trained Model
Predictions
To Hive & Kafka
Performance Monitor Job (Spark)
Metrics
To Monitor System


GET DATA
TRAIN MODELS
EVAL MODELS
DEPLOY, PREDICT & MONITOR
Stream Engine
Cassandra Feature Store
Realtime Predict Service
Trained Model
<< Features
Prediction >>
Client Service
REALTIME
ETL
Hive Feature Store
Batch Training Job (Spark or Mesos)
Training Algo
Trained Models
Deploy
Data Prep Job
Spark / SQL
Outcomes (Training Set)
Features
Batch Predict Job (Spark)
Trained Model
Predictions
To Hive & Kafka
Sampled Predictions
Performance Monitor Job (Spark)
Metrics
To Monitor System
BATCH
Monitor
API
Python / Java
Management

Uber Machine Learning Platform: Feature Store (aka Palette)

Problem

- Hardest part of ML is finding good features
- Same features are often used by different models built by different teams

Uber Machine Learning Platform: Feature Store (aka Palette)

Solution: Centralized feature store

- Curated by Platform team
- Features selected by “join” key
- Offline & online pipelines

Uber Machine Learning Platform: DSL for Feature Selection/ Engineering

Pure function expressions for

- Feature selection
- Feature transformations (for derived & composite features)

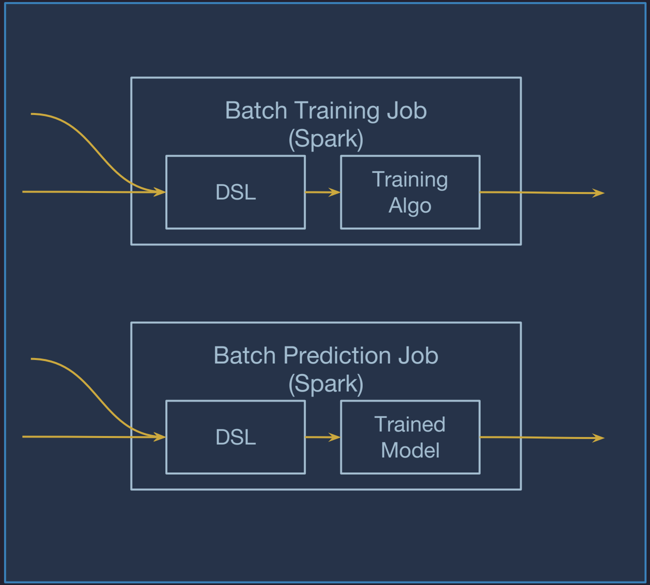
Standard set of accessor functions for

- Feature store
- Basis features
- Column stats (min, max, mean, std-dev, etc.)

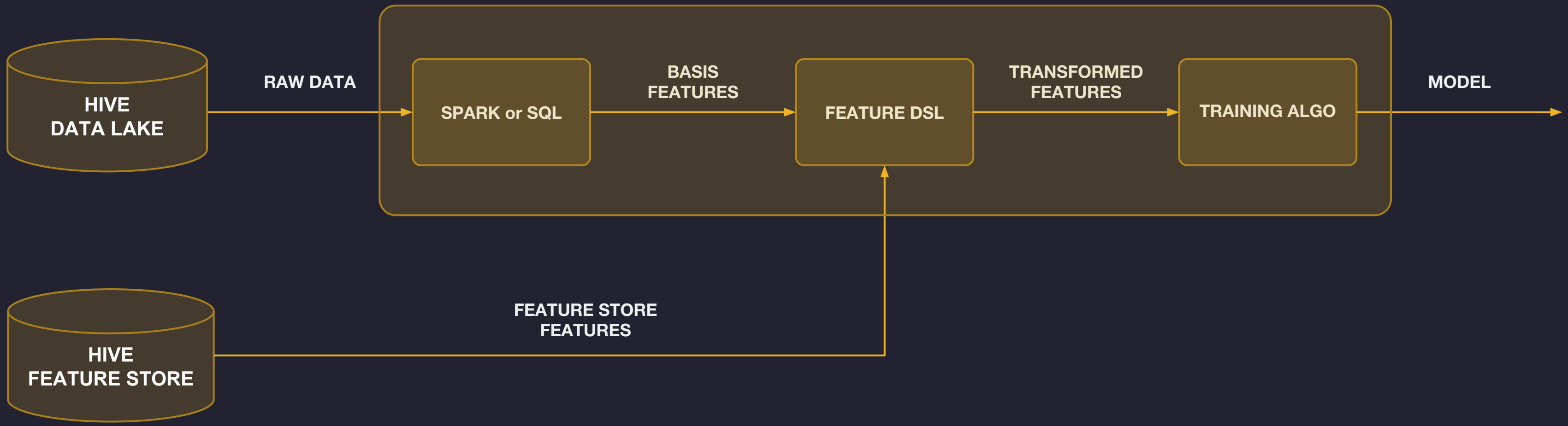
Standard transformation functions + UDFs

Examples

- `@palette:rider:signup_info:signup_channel:rs_uuid`
- `nFill(@basis:distance, mean(@basis:distance))`

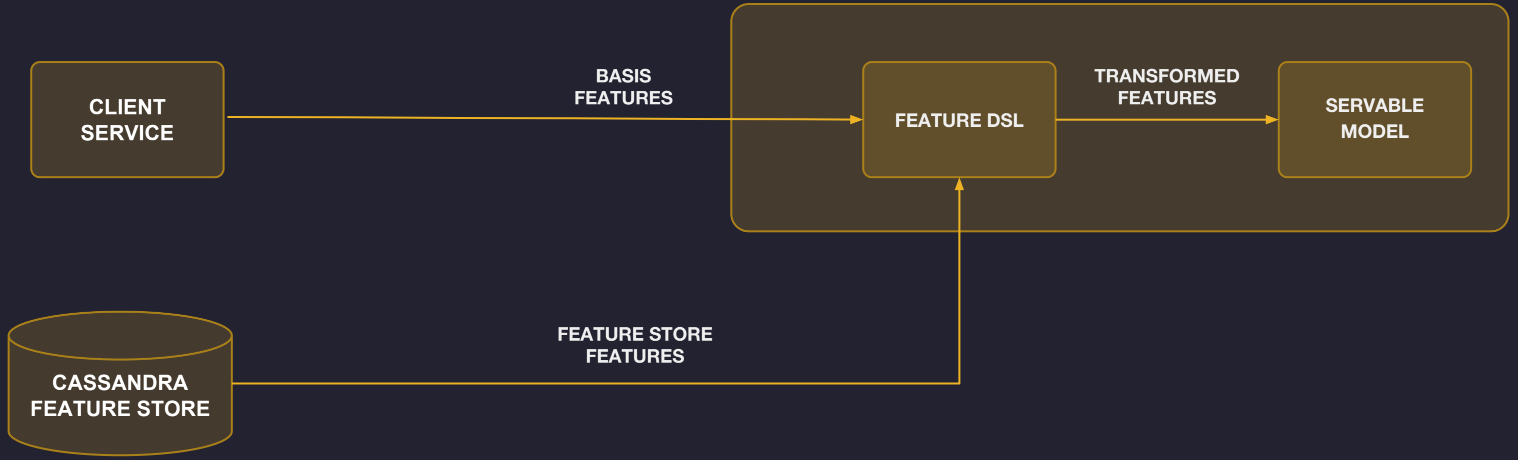


Pipeline for Offline Training with Feature Store



Pipeline for Online Serving with Feature Store

So why is this so helpful?



Partitioned Models

Problem

- Often want to train a model per city or per product
- Hard to train and deploy 100s or 1000s of individual models
- Hard to bootstrap models for new cities with sparse data

Partitioned Models

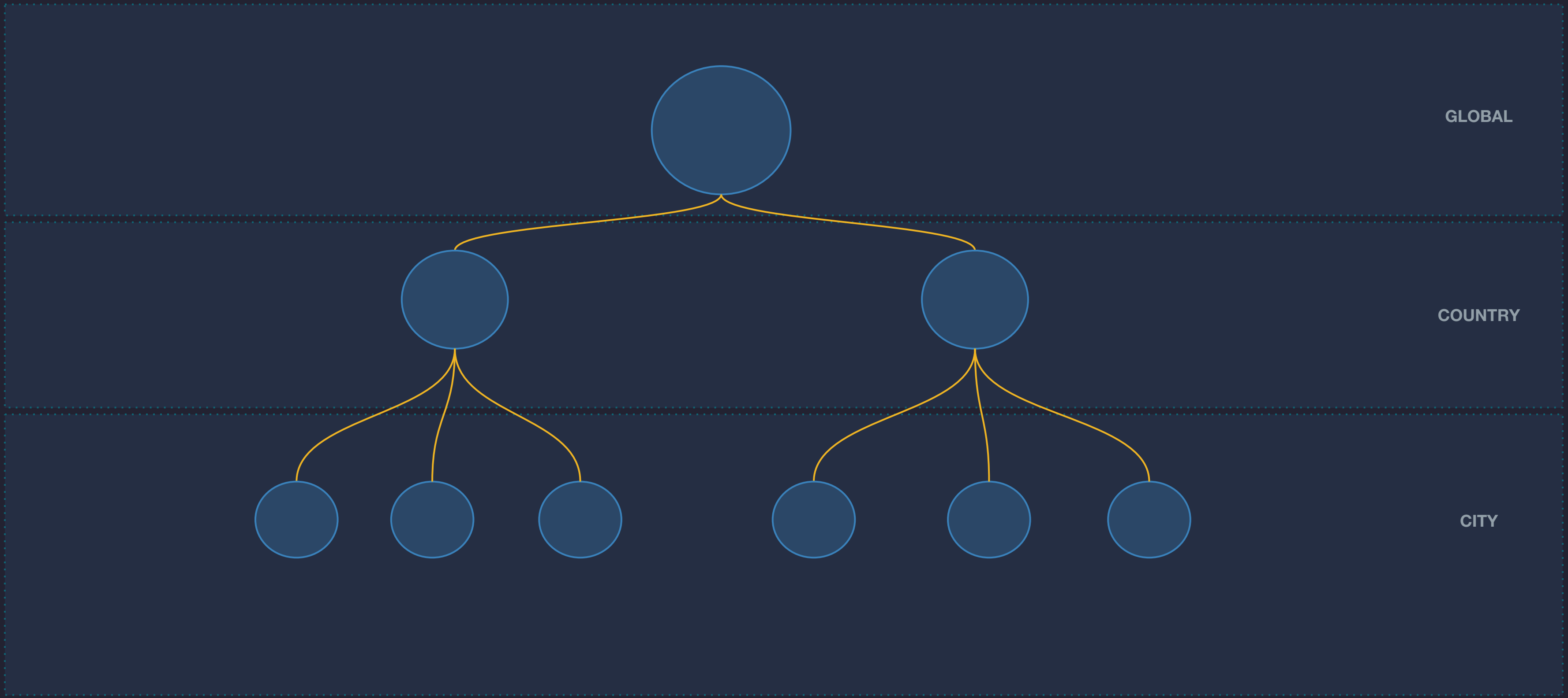
Problem

- Often want to train a model per city or per product
- Hard to train and deploy 100s or 1000s of individual models
- Hard to bootstrap models for new cities with sparse data

Solution

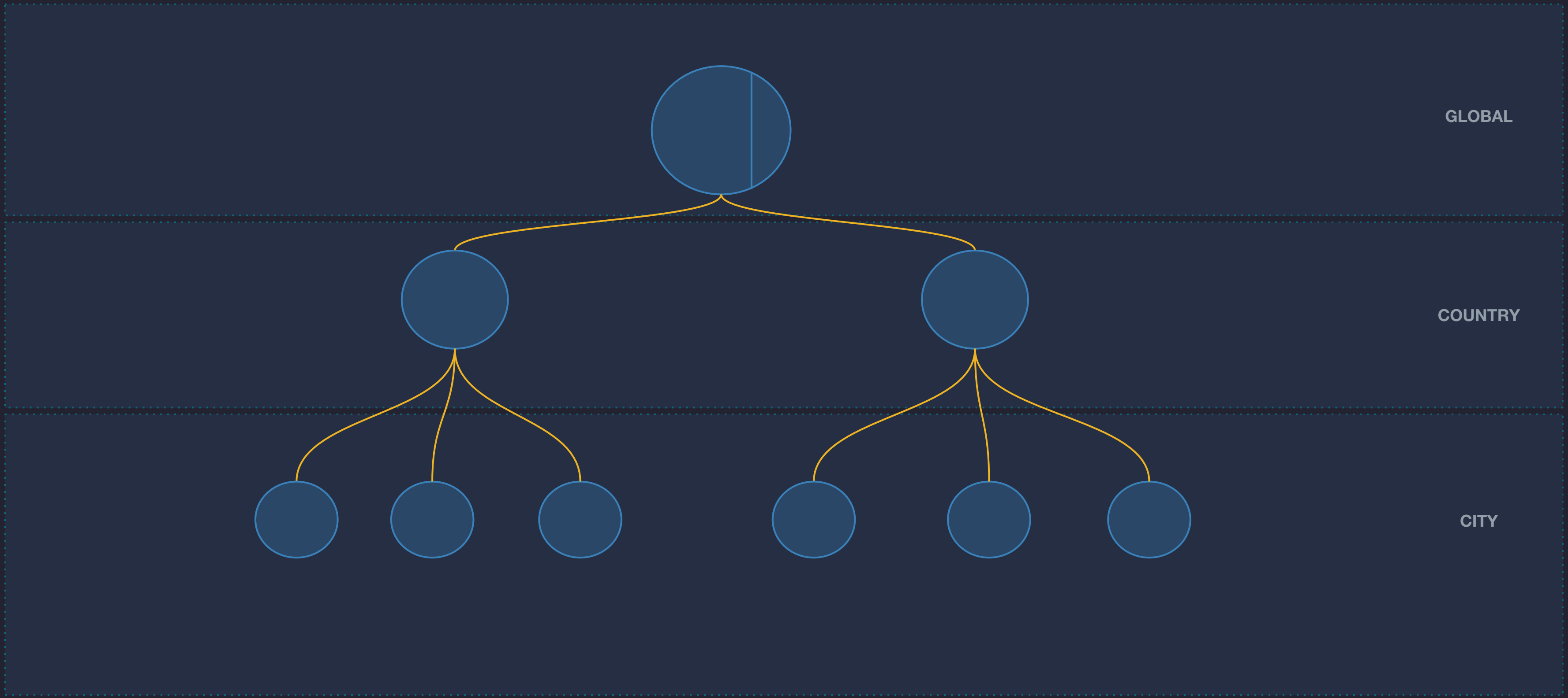
- Let users define hierarchical partitioning scheme
- Automatically train model per partition
- Manage and deploy as single logical model

1 **Define partition scheme**



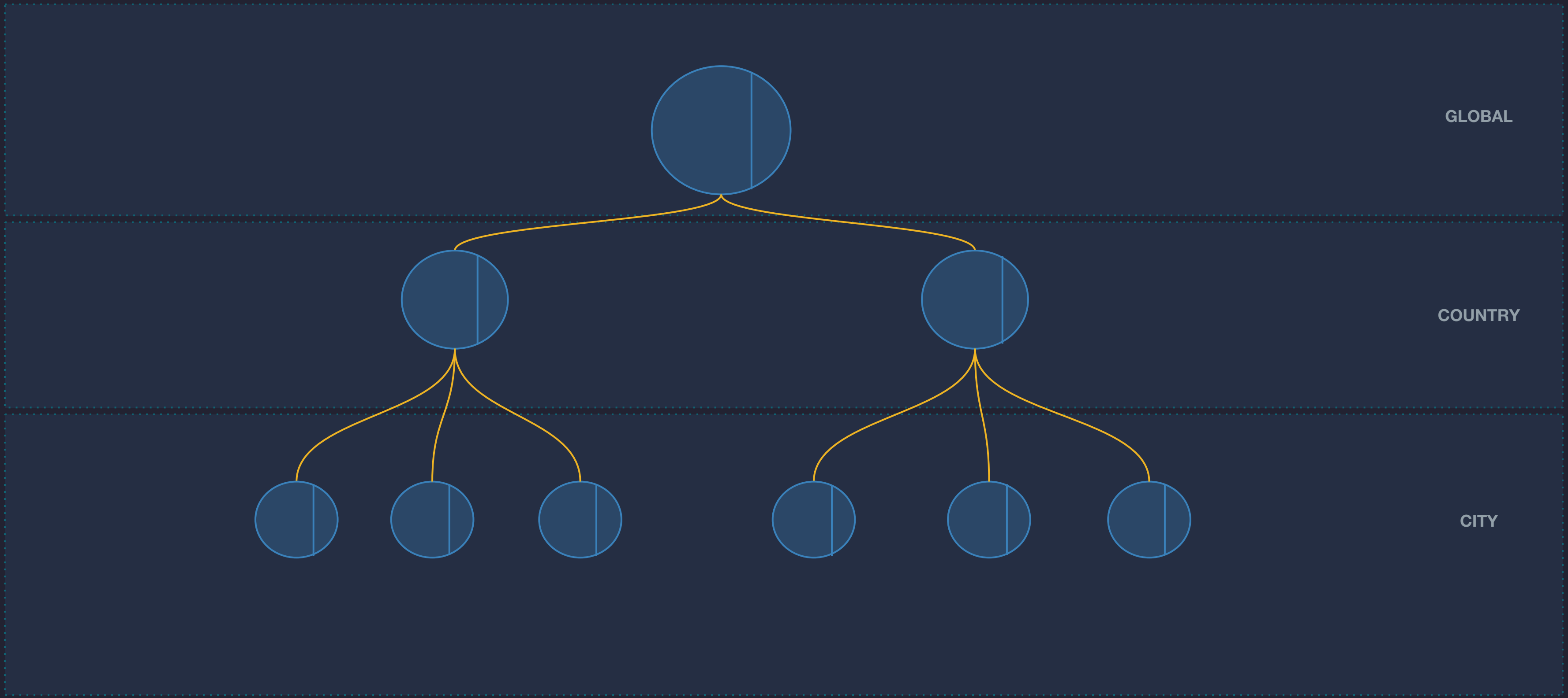
2

Make train / test split

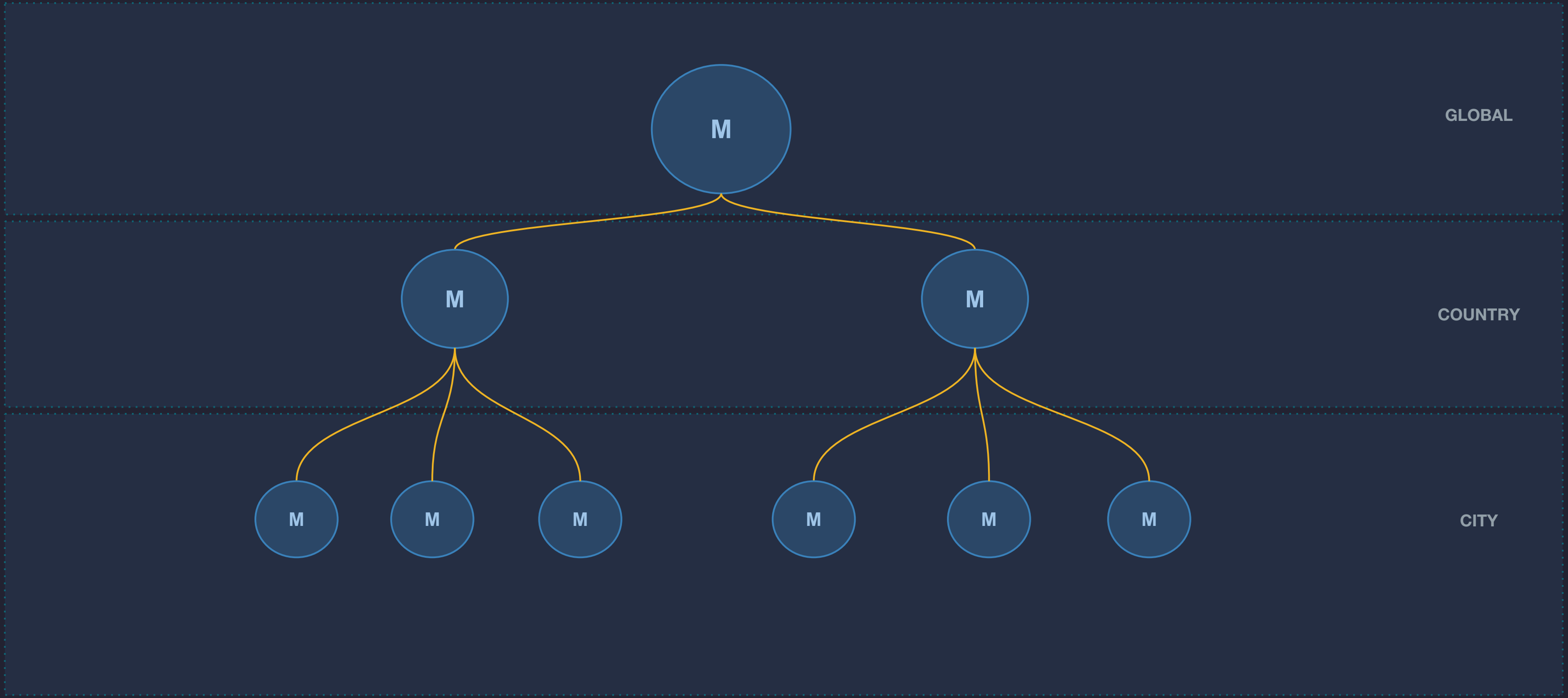


3

Keep same split and partition for each level

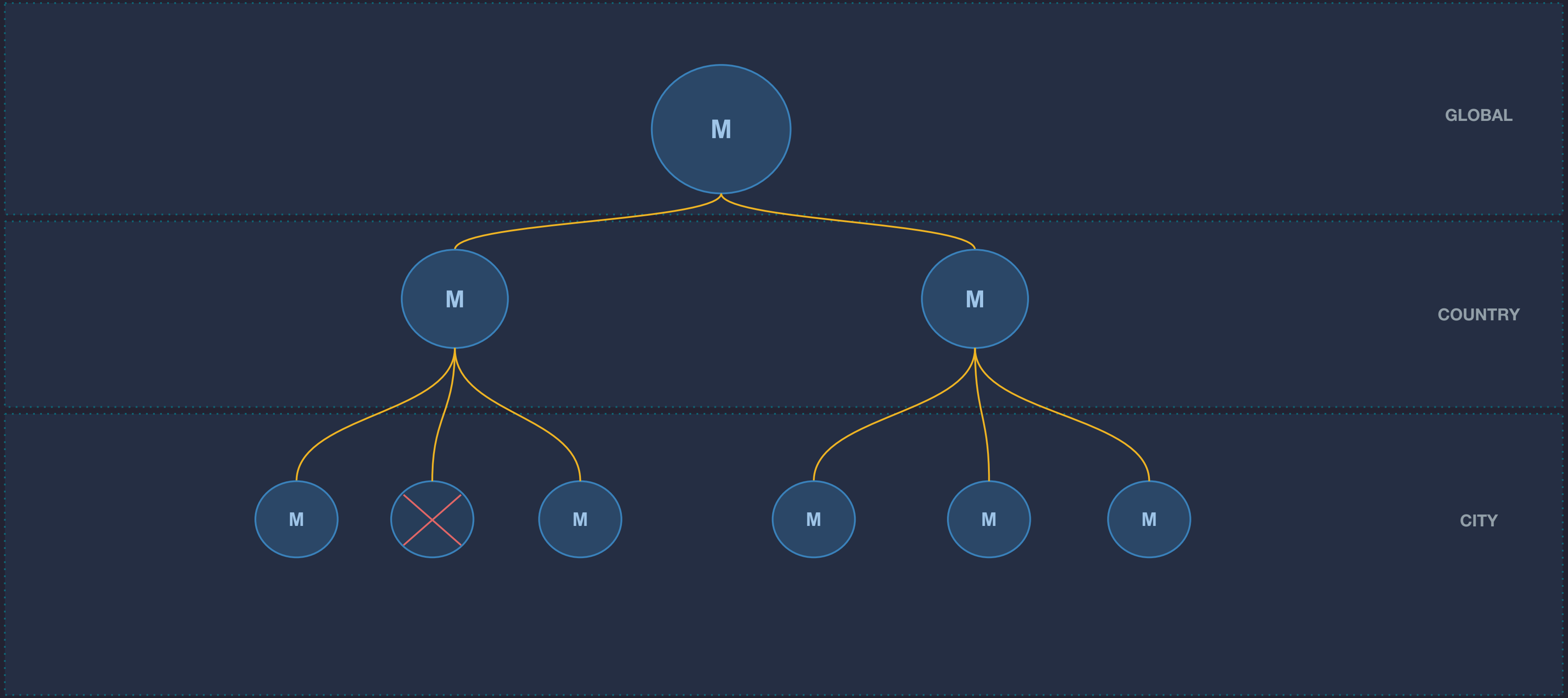


4 Train model for every node

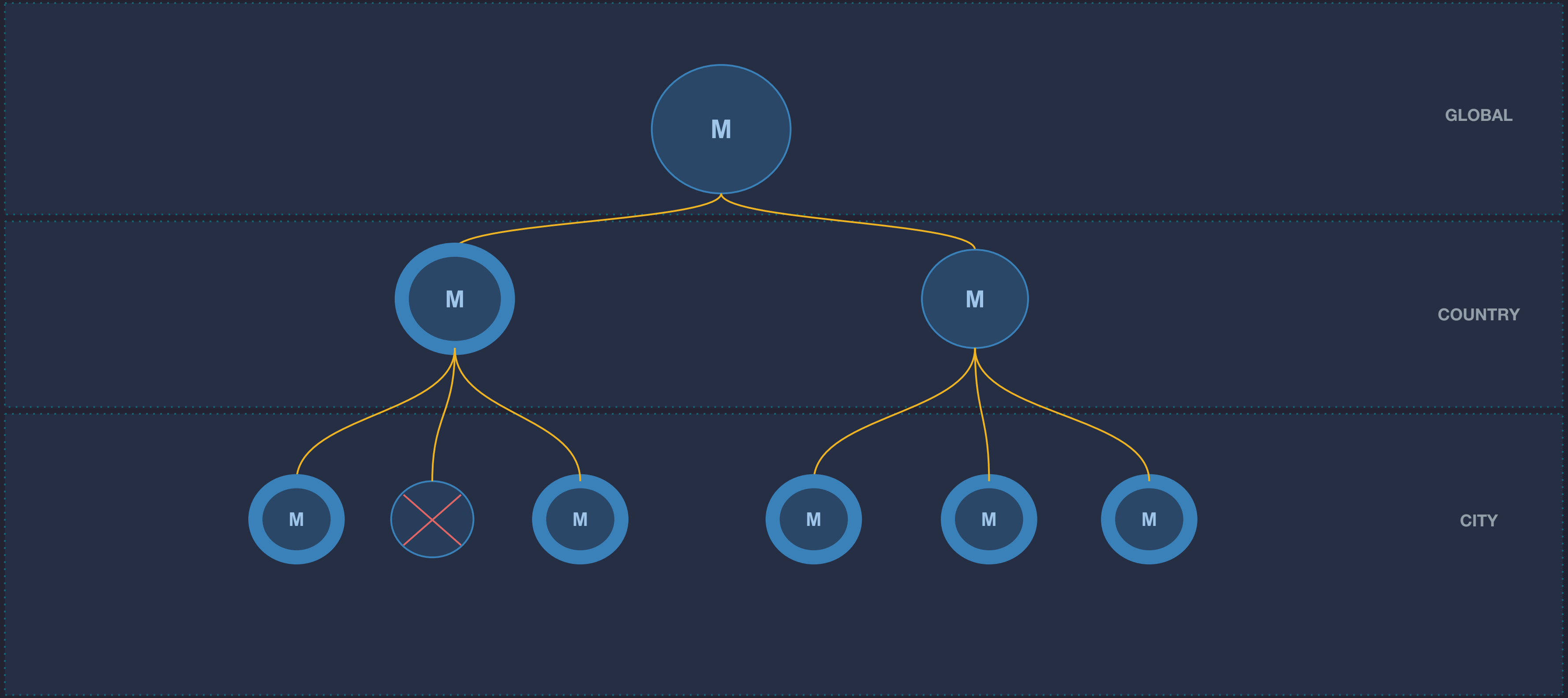


5

Prune bad models



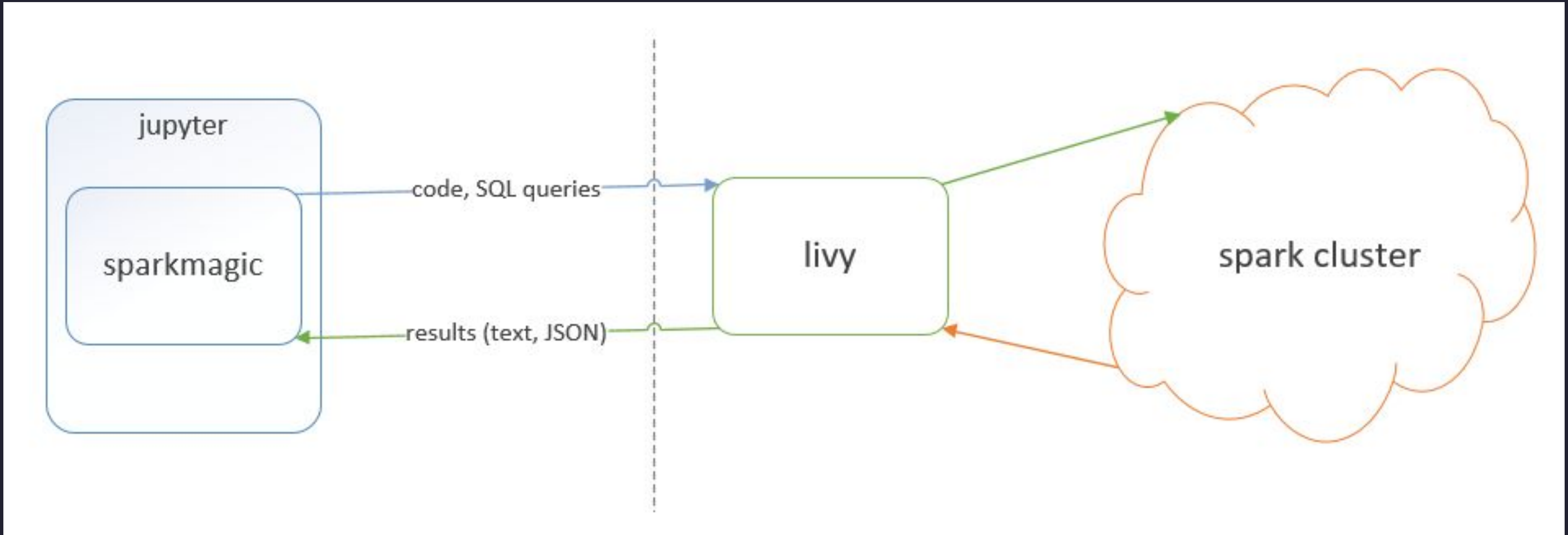
6 At serving time, route to best model for each node



AGENDA

- Motivating Example
- Scala in Experimentation
- Scala in Machine Learning
- **sparkmagic**

sparkmagic


jupyter
sparkmagic
code, SQL queries
results (text, JSON)
livy
spark cluster

sparkmagic

- **Not constrained by imperative language interfaces like PySpark and SparkR**
- Can process a tremendous amount of data

Thank you!
Questions?

eng.uber.com/michelangelo
eng.uber.com/xp
eng.uber.com/experimentation-platform
uber.com/careers

nick.jones@uber.com


UBER