FEATURE STORE SUMMIT

12-13 OCTOBER | 08:30 AM - 4:00 PM PT

ORGANIZED BY HOPSWORKS

Uber Engineering

Feature Stores for ML

Michelangelo Palette at Scale

Amit Nene
Architect, Manager
Uber

Nicholas Marcott
Lead Engineer
Uber

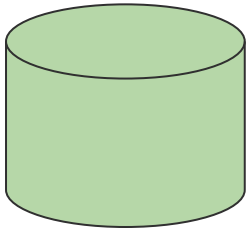
Uber Engineering

Feature Stores for ML

Michelangelo Palette

Feature Preparation
Batch + Streaming ETLs

Palette Feature Store
Online serving + Joins
Offline serving + Joins

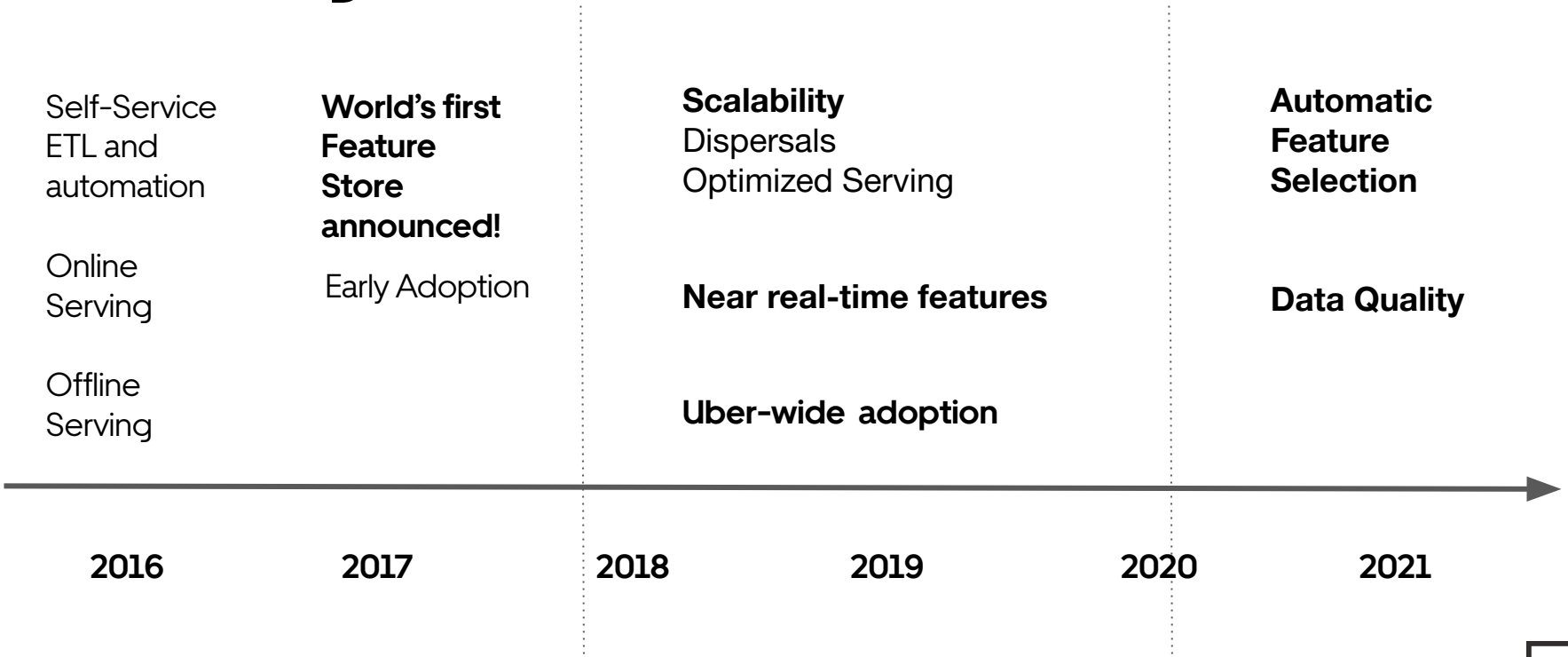


Feature Discovery
Sharing across Models
Automatic feature selection

Monitoring
Pipeline & Data Quality

Michelangelo Transformer
Model specific feature xforms

History


Self-Service ETL and automation
Online Serving
Offline Serving
World's first Feature Store announced!
Early Adoption
Scalability
Dispersals
Optimized Serving
Near real-time features
Uber-wide adoption
Automatic Feature Selection
Data Quality
2016
2017
2018
2019
2020
2021

Problems at Scale

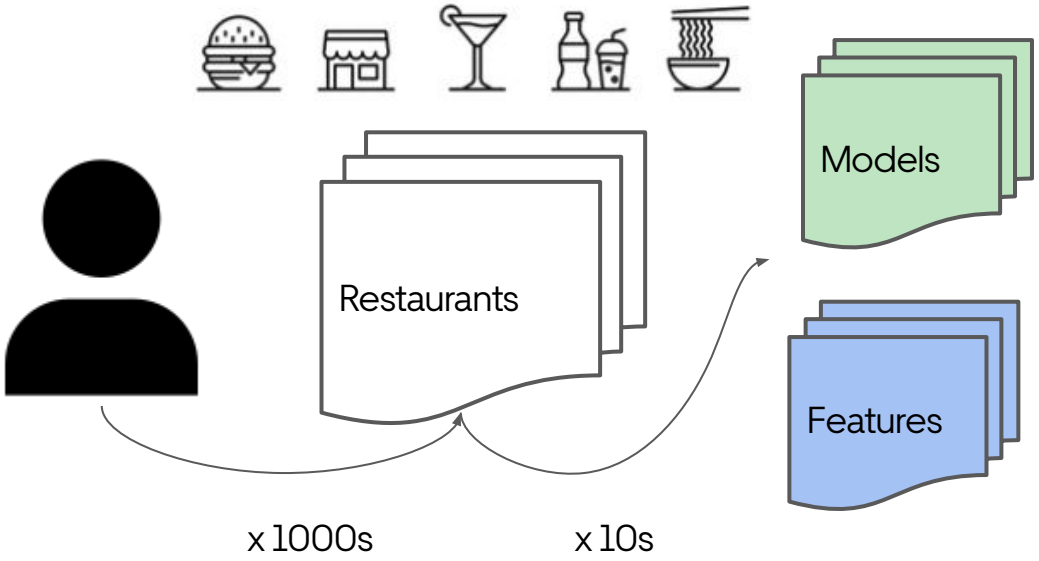
Online Serving
10s of Millions of QPS

Offline Serving
100s of TBs per training run

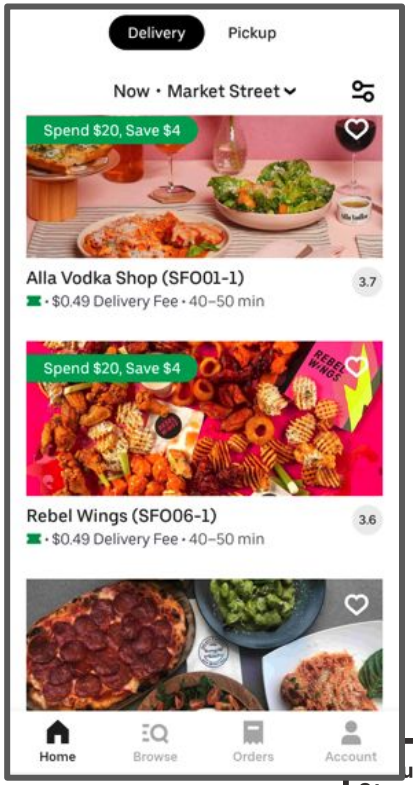
Pipeline Scale
1000s of Feature Pipelines

Discovery at Scale
Select from 100K features

Online Serving



Final ranking



Online Serving

Serving Infra

Table Consolidation

Custom Partitioning

Efficient Dispersals

Serving Infra

QPS/Latency per $

Storage Capacity Advanced uses

Local cache

JVM-local, in-memory

Thousands of cities (100s of MBs)

Hottest partitions

Remote cache

Distributed, in-memory (eg. Redis)

Millions of restaurants (100s of GBs)

Hot partitions

NoSQL

Distributed, KV-Store (eg. Cassandra)

100 Millions of Users (10s of TBs)

Table Consolidation

Feature Group Table Design

- **Ownership**
- **Feature Gen Job**
- **Semantic Grouping**

Restaurant (Key)	Wait Times
Starbux	5min
Fills	10min
Black Bottle	20min

Restaurant (Key)	Ratings
Starbux	4.7
Fills	4.8
Black Bottle	4.9

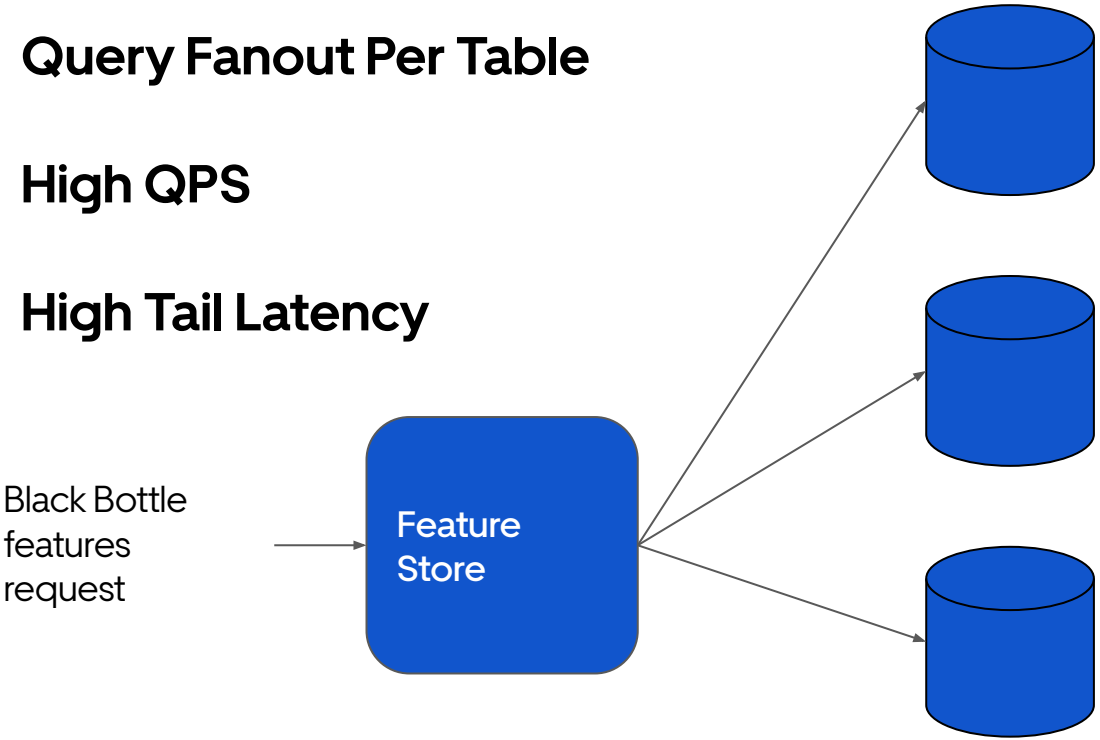
Restaurant (Key)	Embeddings
Starbux	[0.4, 0.1, ...]
Fills	[0.9, 0.2, ...]
Black Bottle	[0.5, 0.4, ...]

Table Consolidation

Query Fanout Per Table

High QPS

High Tail Latency



Restaurant	Ratings
Black Bottle	4.9

Restaurant	Wait Times
Black Bottle	20min

Restaurant	Embeddings
Black Bottle	[0.5, 0.4, ...]

Table Consolidation

Consolidated Tables

Eliminate fanout

Low Latency

Restaurant	Ratings	Wait Times	Embeddings
Black Bottle	4.9	20min	[0.5, 0.4, ...]

Custom Partitioning

Features based on 2 or more keys
Partition key = Primary Key = User + Restaurant

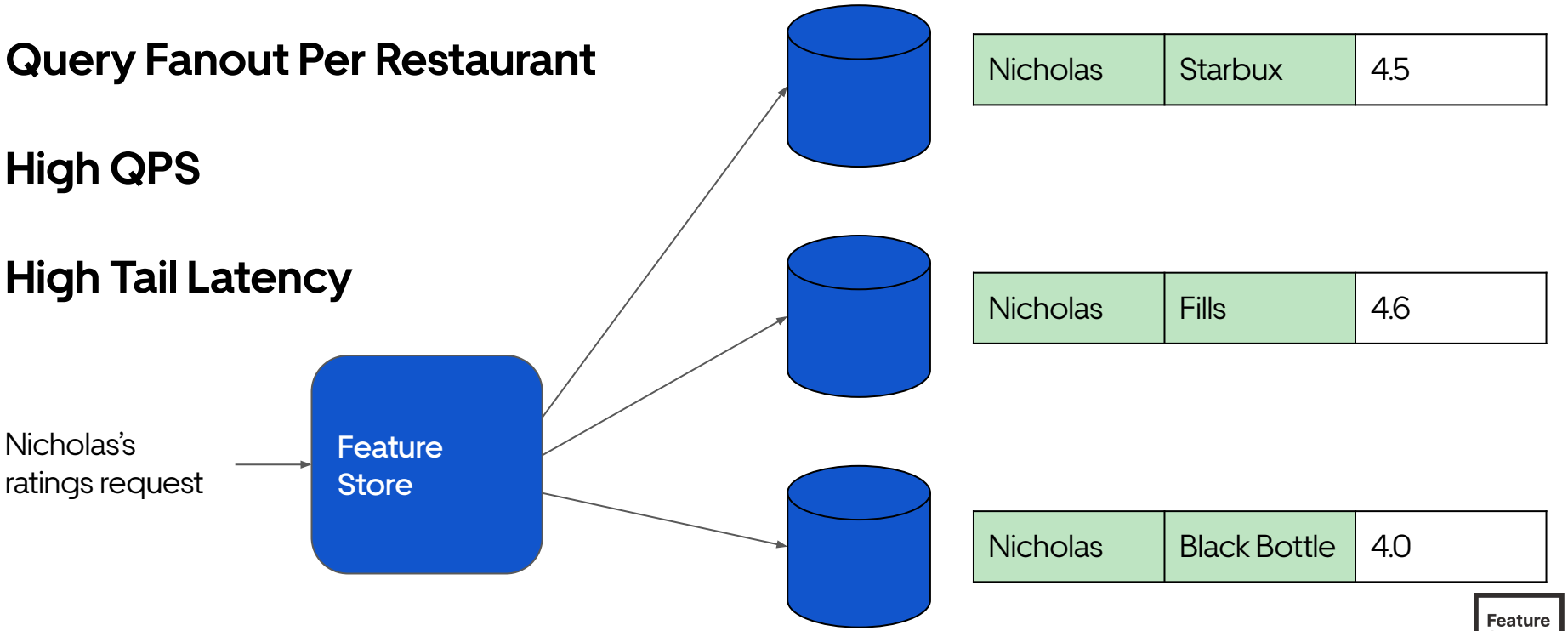
User (Key 1)	Restaurant (Key 2)	Avg Rating (Value)
Nicholas	Starbux	4.5
Nicholas	Fills	4.6
Nicholas	Black Bottle	4.0

Custom Partitioning

Query Fanout Per Restaurant

High QPS

High Tail Latency



Custom Partitioning

Customized partitioning

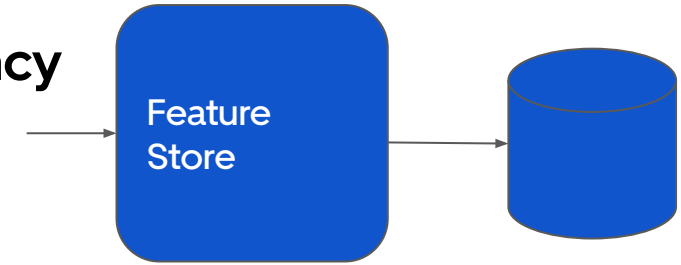
Partition Key = User

Shard Local Key = Restaurant

Eliminate fanout

Low Latency

Nicholas's ratings request



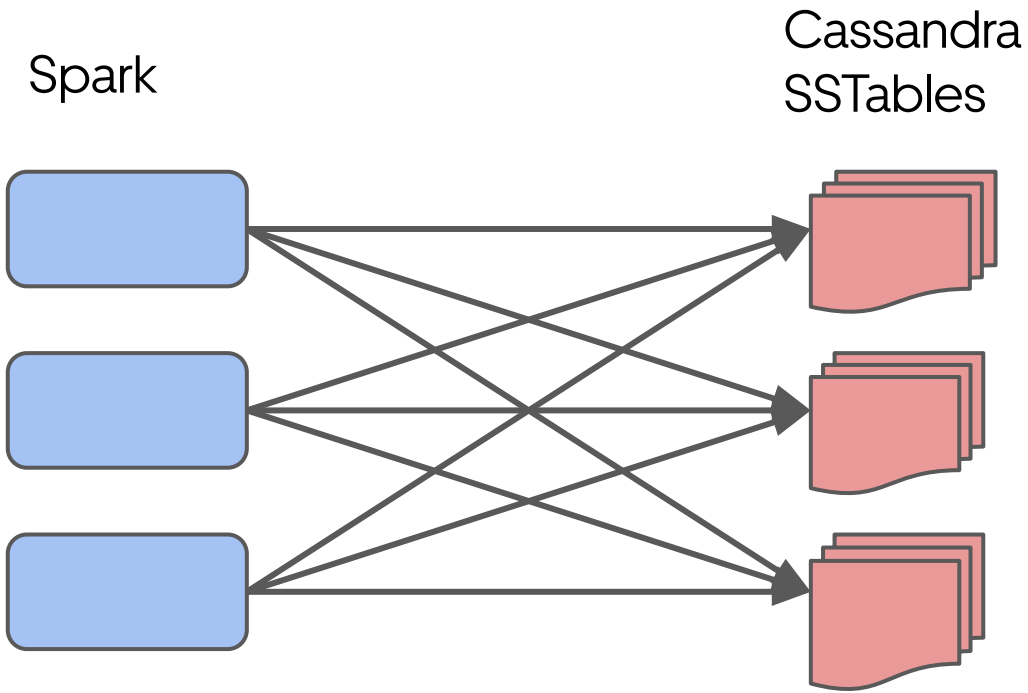
Nicholas	Starbux	4.5
Nicholas	Fills	4.6
Nicholas	Black Bottle	4.0

Efficient Dispersals

Too many SSTable files created

Compaction falls behind

High Read Latency

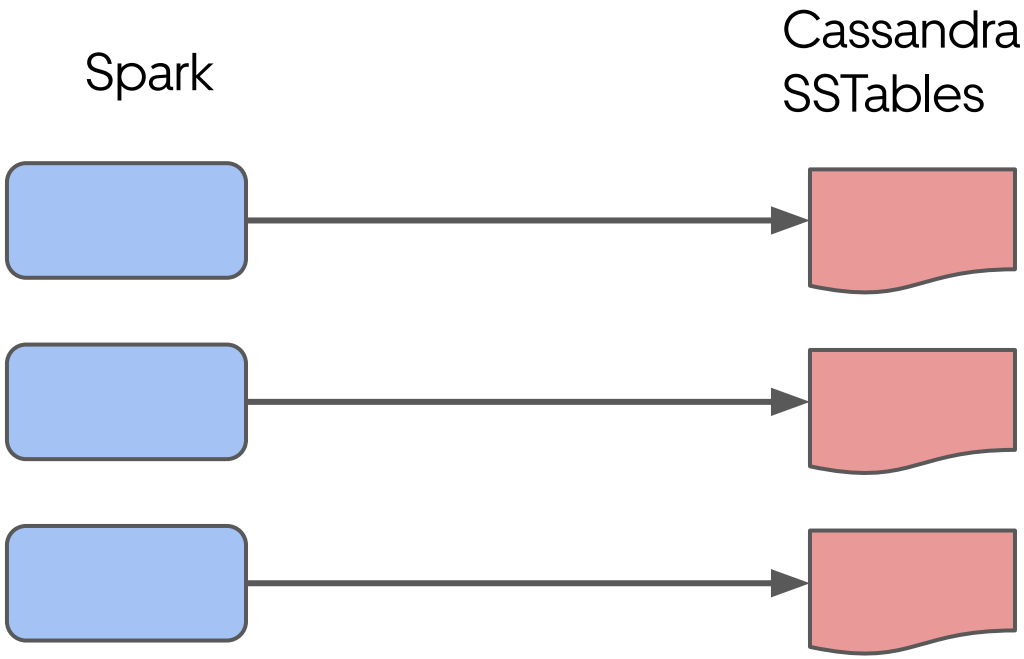


Efficient Dispersals

Cassandra and Spark partition functions aligned

Tuning Compaction algorithms

Consistency, Read repair, GC, SSD config, etc.



Offline Serving

How to tune Spark for huge joins?

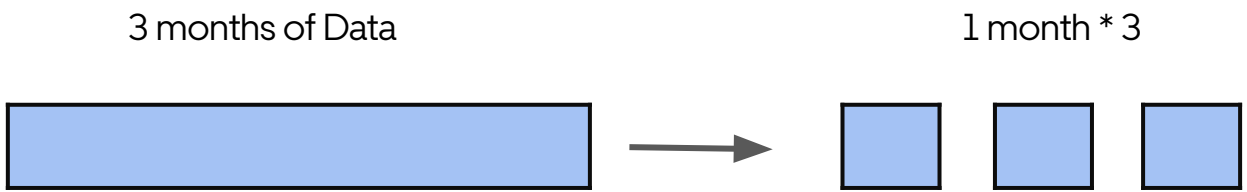
How to avoid skewed joins and OOMs?

How to speed up joins?

Batching

Break join into manageable batches

Fixed tuning per batch



Delta

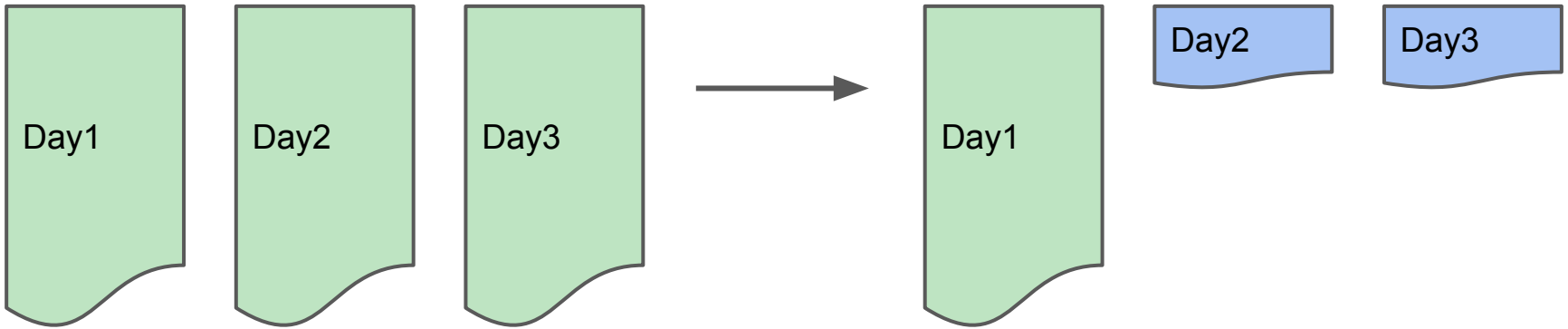
Process updates instead of entire snapshots

	Full Snapshot		
	Num orders (Feature Values)		
User (Key)	**Day 1**	**Day 2**	**Day 3**
Nicholas	10	10	10
Amit	21	21	22

	Delta Snapshot		
	Num orders (Feature Values)		
User (Key)	**Day 1**	**Day 2**	**Day 3**
Nicholas	10	-	-
Amit	21	-	22

Delta

10X or more reduction in Spark shuffle cost



Other Join Optimizations

Spark optimizations

Job Scheduling, Map-side joins, Filtering, Large Containers

Join reuse framework

Reuse joins across multiple training iterations (eg. hyperparameter tuning)

Pipelines at scale

Are 1000s of Pipelines are producing reliable features ?

How can we avoid debugging of issues at Training time ?

How can we enforce accountability ?

Uber Data Quality

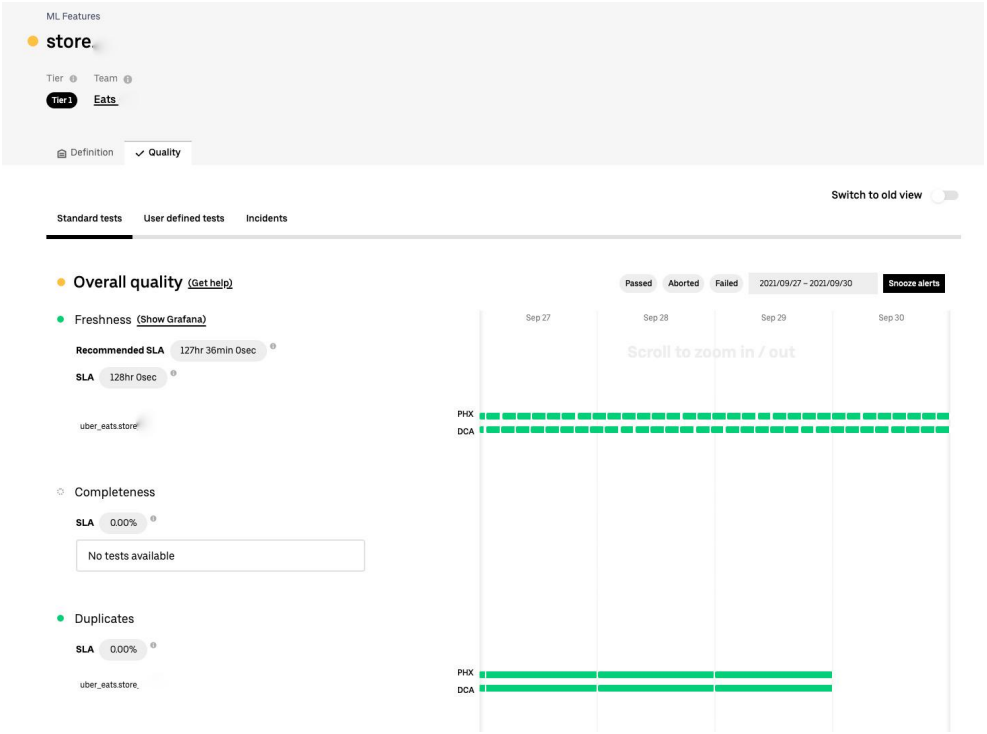
Metadata

Lineage

Tiering

Test Registration

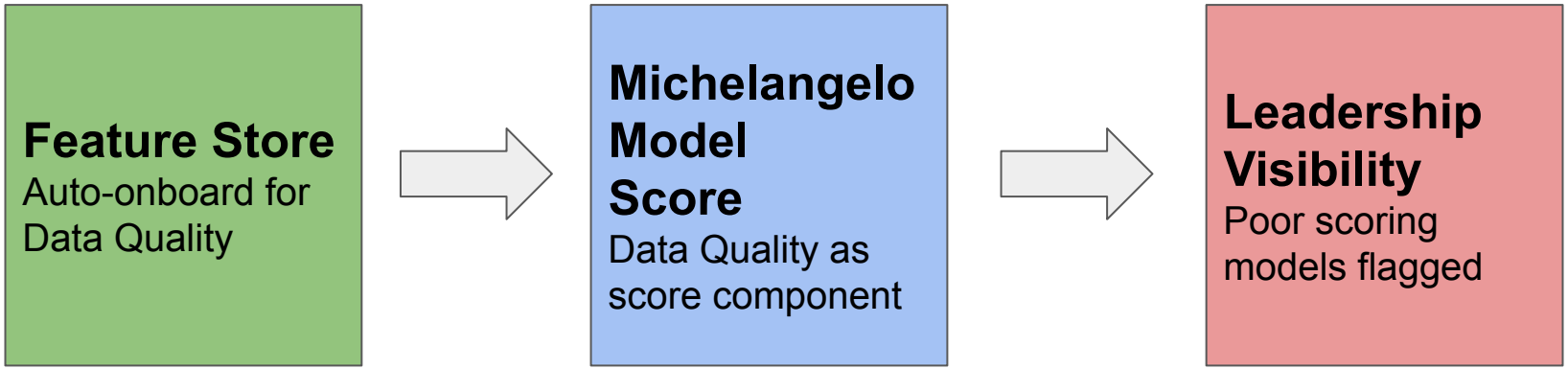
Ownership



Uber Engineering

Feature Stores for ML

Tying to ML Quality



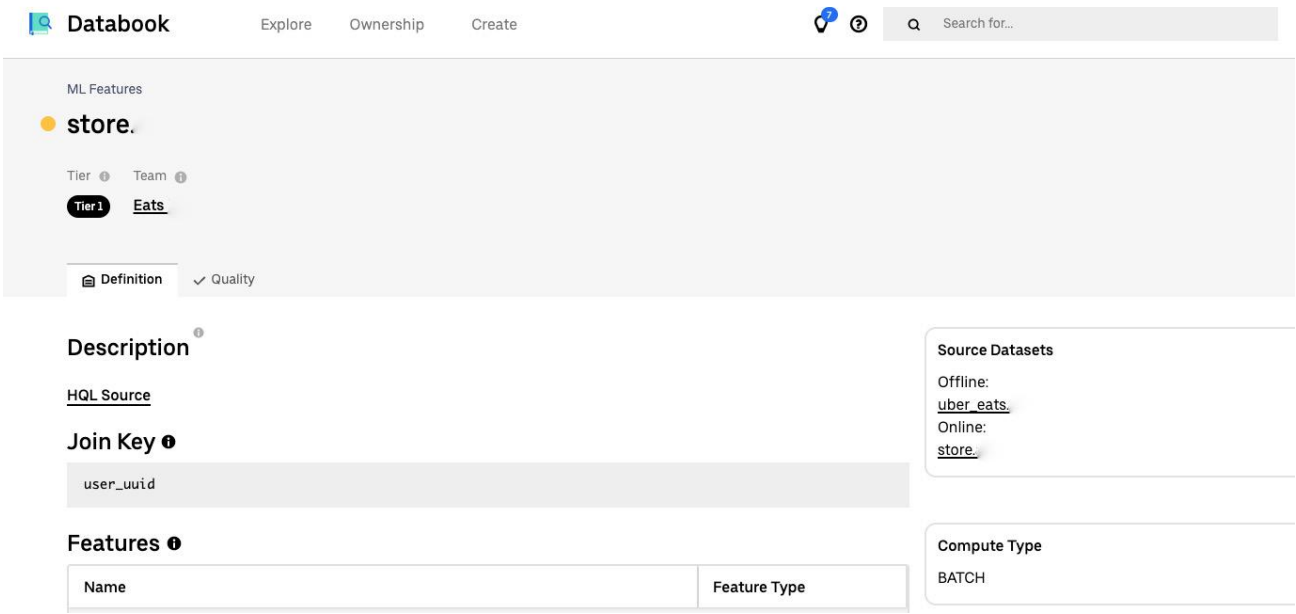
Discovery at Scale

How to choose relevant features from repository of 100K+ ?

How do we avoid redundant building of features ?

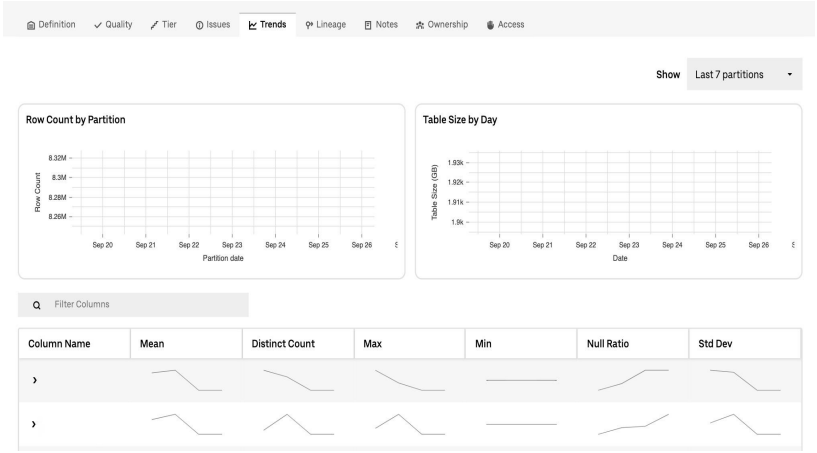
Feature Store Search

Search by entity, key, name, etc.

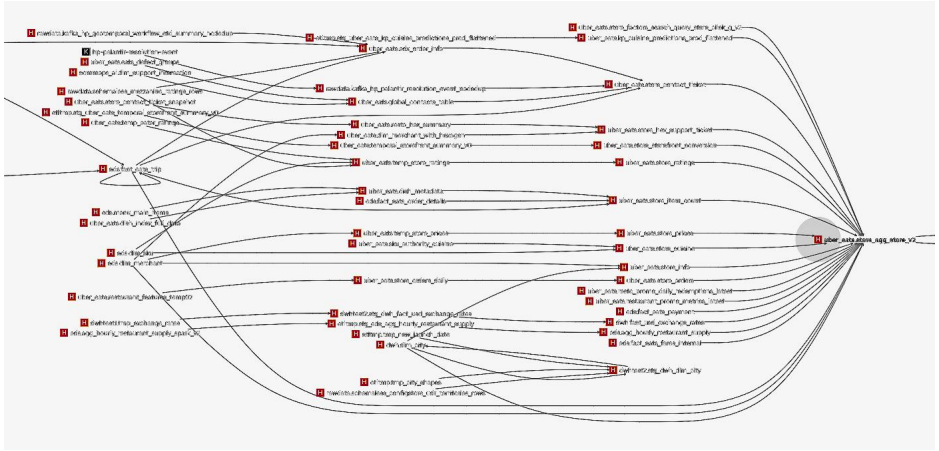


Data Browser

Stats, Data quality



Lineage



Automatic Feature Search

Entropy of a feature

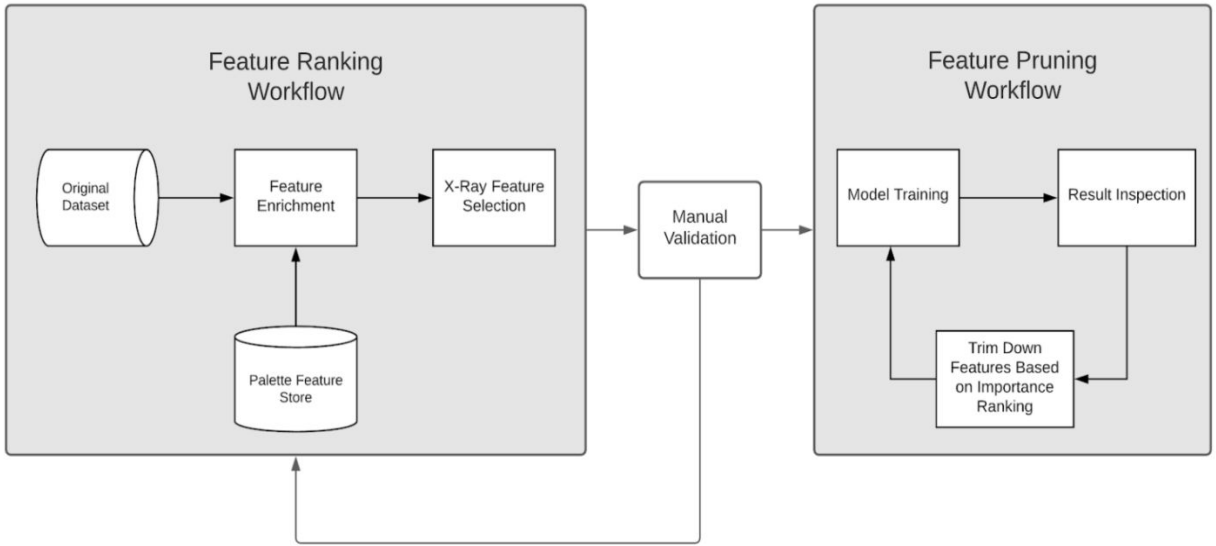
$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

Mutual Information: Feature and Label

$$I(X;Y) = H(X) - H(X|Y)$$

Search Feature Store by Join Key, rank by MI

Michelangelo Training workflow



High level workflow of optimal feature discovery

Where we're headed

Recommendation Systems
Uber Search Engine
Inline execution of Models

Feature Intelligence
Lineage across Data and ML
Data Mining tools

Embeddings
Vector types, Versioning,
Discovery

Near real-time features
Aggregation infra
Seamless backfills

Thank you!

Do you have any questions?

https://www.linkedin.com/in/amitabh-nene/

https://www.linkedin.com/in/bmarcott/