Gallery: A Machine Learning Model Management System at Uber

Nader Azari, Yifan Ma

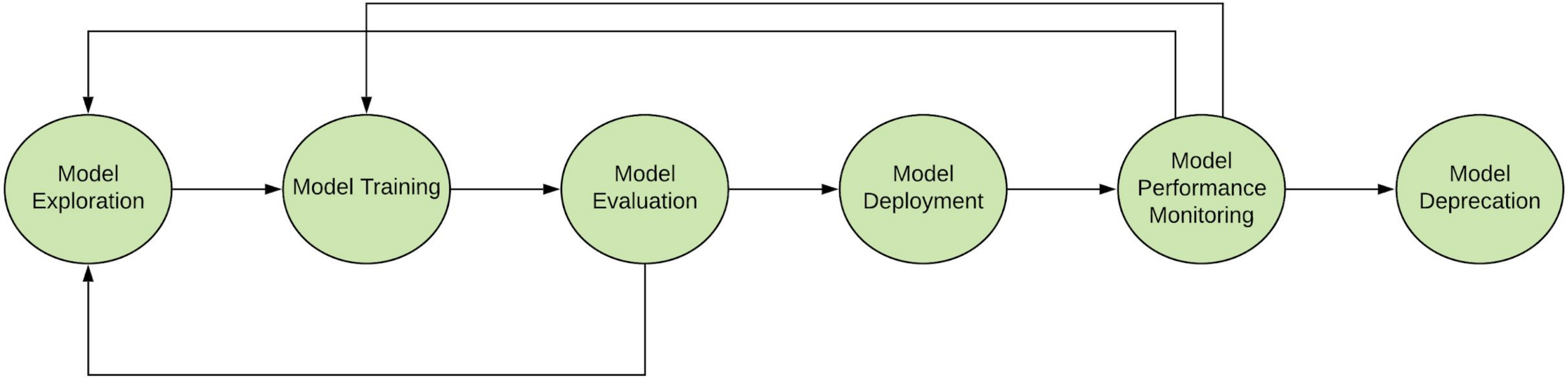
July 10, 2020

Uber

Agenda

01 Introduction
02 Design Principles
03 Gallery System
04 Case Studies
05 Lessons Learned and Next Steps
06 Q&A

Machine Learning Model Lifecycle



Uber's Scale

Cities

1000+

Countries

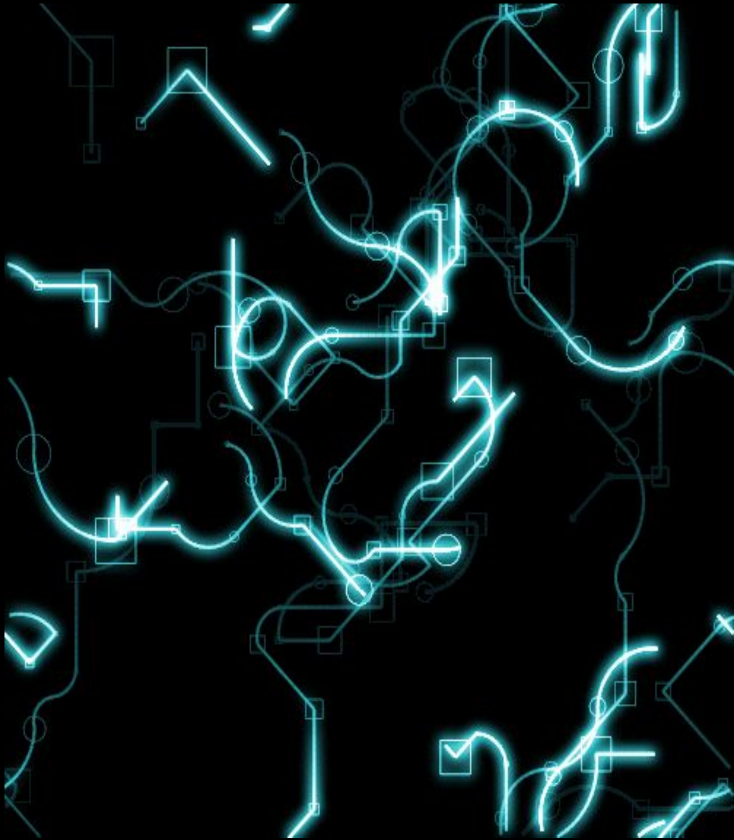
80+

Products

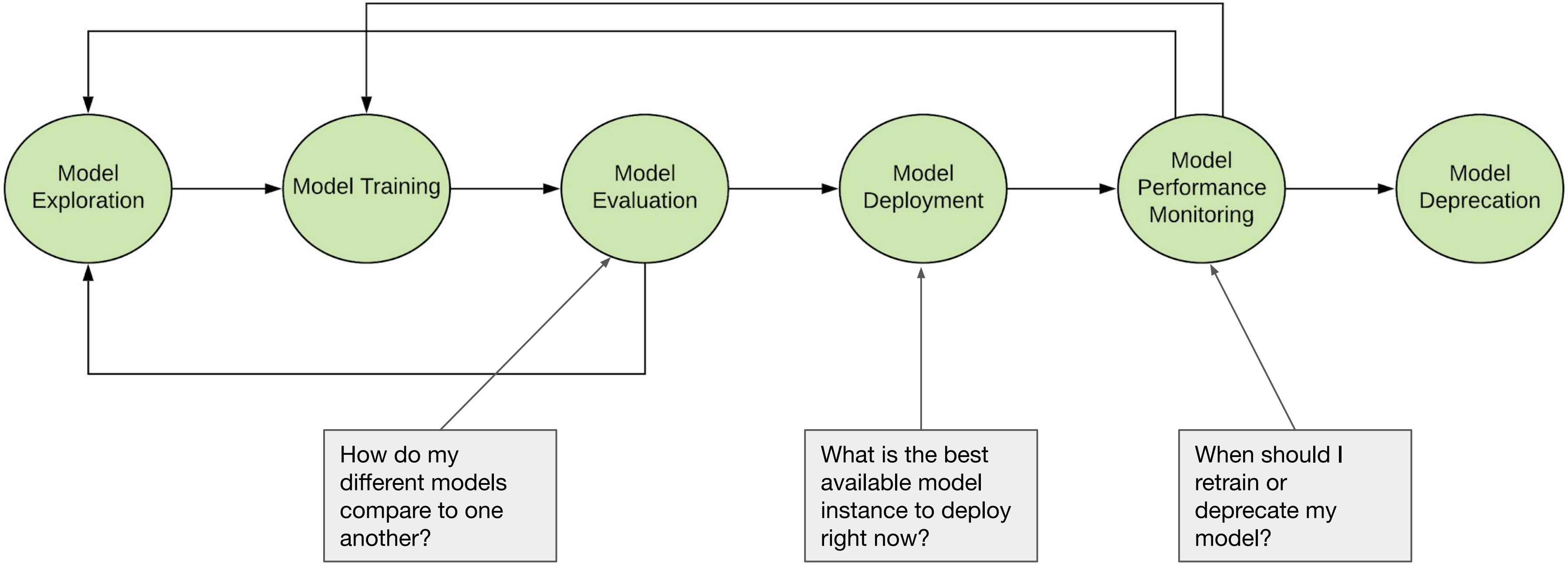
20+

Model Instances

1m+



Machine Learning Model Lifecycle



Motivating Challenges

- Managing model lifecycle at scale.
- Microservice architecture leads to **custom modeling platforms.**
- Inability to collaborate
  - The lack of a central model manager and the proliferation of modeling platforms leads to models being built in isolation and causes **cross-team incompatibility**.

Motivating Industry Example

Not many existing examples of model management. But we can draw inspiration from software development.

Git

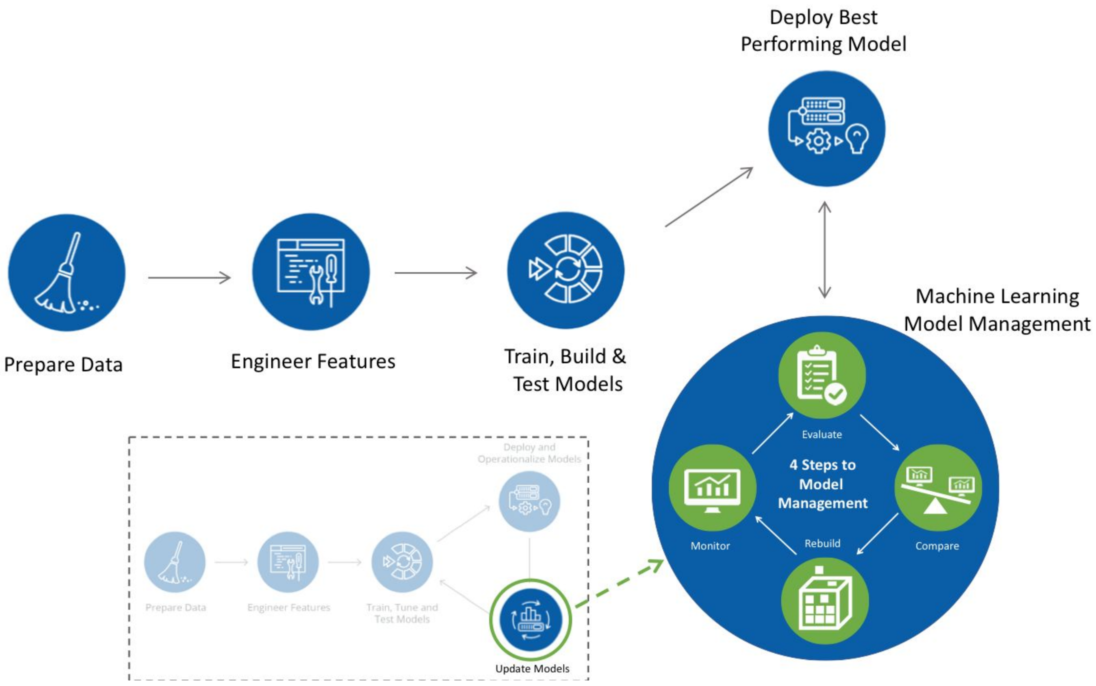
1. **Well-understood and standard API** that's compatible with all dev environments.
2. **Central repository** for all code that enforces data accuracy.
3. **Standardized schema** for versioning and referencing code.
4. Building block for **CI/CD** systems.

Michelangelo Gallery

"Git for models"

Gallery is a system that

- Is part of **Michelangelo**[1], Uber's internal ML-as-a-service platform.
- **Stores** models with associated **metadata and metrics.**
- **Versions** models and tracks dependencies to enable reproducibility.
- Provides a **search engine** to automate orchestration decisions.



Gallery[2]

1. https://eng.uber.com/michelangelo-machine-learning-platform/
2. https://community.hitachivantara.com/s/article/4-steps-to-machine-learning-model-management

Agenda

01 Introduction
02 Design Principles
03 Gallery System
04 Case Studies
05 Lessons Learned and Next Steps
06 Q&A

Design Principles

Immutable

All model instances managed by Gallery are immutable ensuring that any prediction can be tracked to a model.

Model Neutral

Models are treated as a black box and allow Gallery to manage models independently of their framework.

Framework Agnostic

APIs and features are designed to be leveraged by any modeling ecosystem to enable usage by all applications.

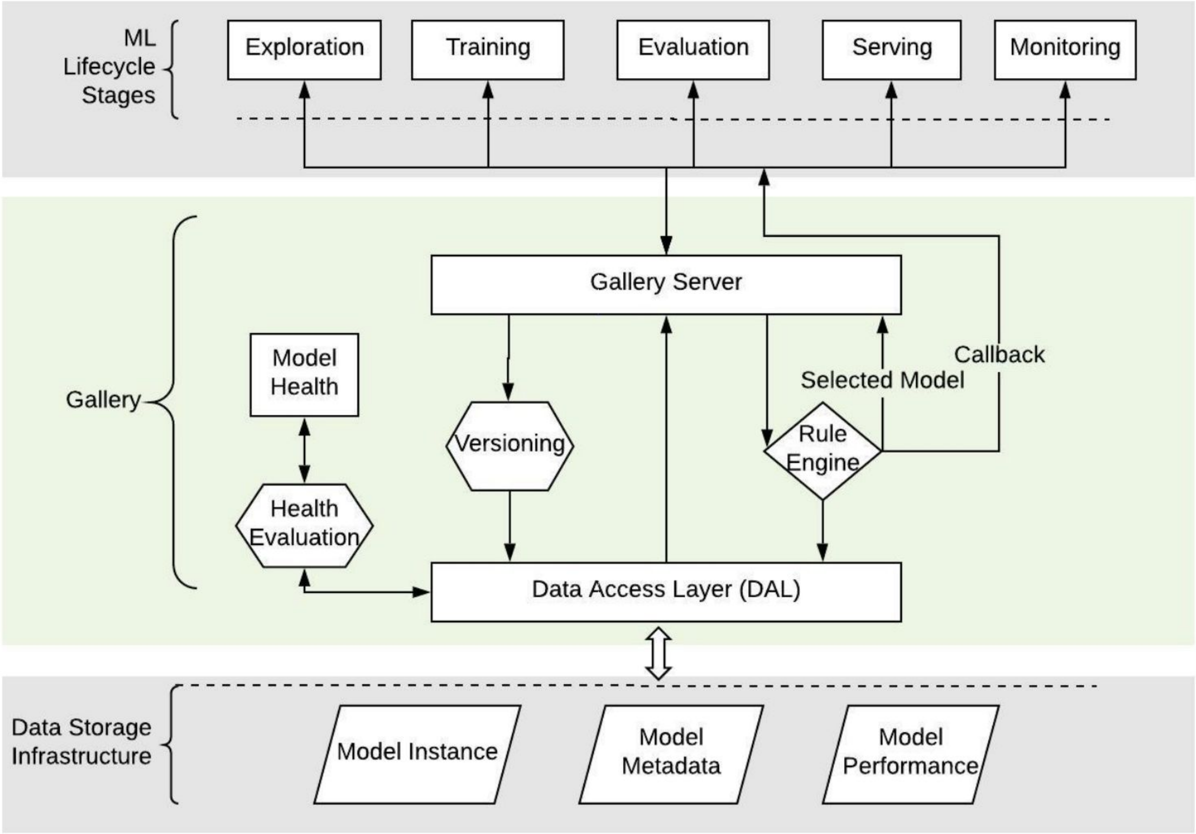
Automation

Build features that support automating model lifecycle stages to reduce manual production maintenance costs.

Agenda

01 Introduction
02 Design Principles
03 Gallery System
04 Case Studies
05 Lessons Learned and Next Steps
06 Q&A

System Architecture



Blob Storage

"Git for Models"

- Gallery provides a model format and framework agnostic API for users to commit trained models.
- Trained models can be retrieved at serving time or for adhoc analysis.
- Underlying blobs are stored in S3.

API

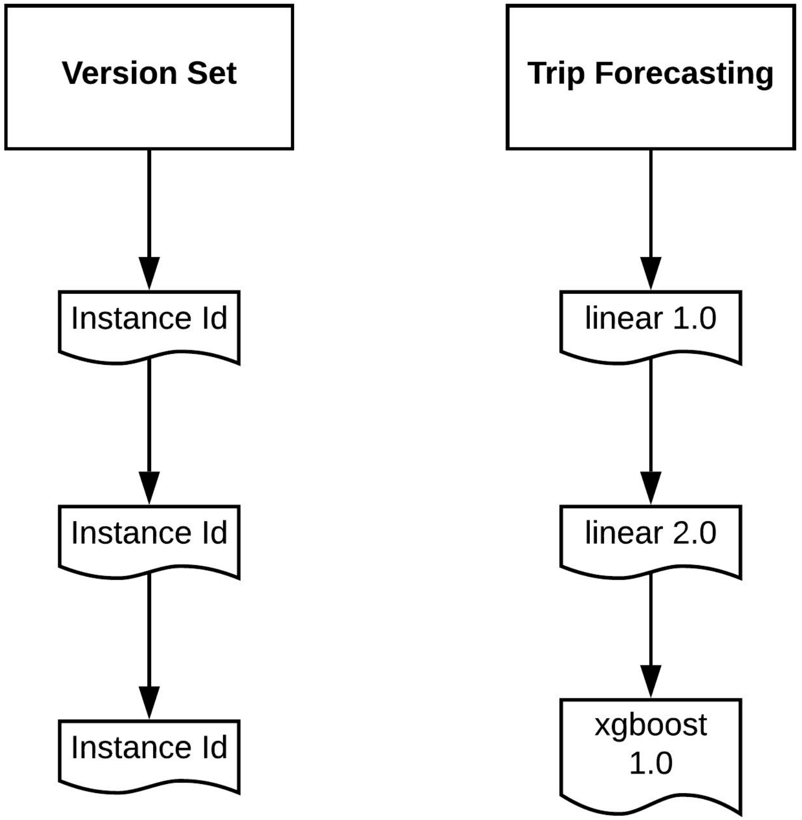
- **upload_model_blob**(project, model, file_name)
- **download_model_blob_content**(project, model)

Versioning

Model instances are tagged with version ids and are associated to one another to trace lineage.

Instances solving the same business problems are grouped together as version sets.

This allows model developers to track the evolution of their models over time.



Example: VERSIONS page for models

MODELS | VERSIONS | TEMPLATES | + CREATE MODEL

All | Trained | Deployed | Search... | 1 of 2 | 10 / page

	JOB / MODEL ID	TYPE	OWNER	TRAINING TIME	PERFORMANCE			
	trip_prediction							
✓	tm20191118-210901-NZBKVFQM-ZKWWYW tm20190401-142354-PDBBLLPF	Random Forest Classificatio		--	AUC 0.7076		DEPLOY	...
	trip_conversion							
✓	tm20190620-161543-TVSKJWUH-TJKTZL Retrain tm20190523-201133-BUKPZSIT	Random Forest Classificatio		00:35:04	AUC 0.6761		DEPLOY	...
✓	tm20190619-221900-LLSUMNUE-JDOFNH Retrain tm20190523-201133-BUKPZSIT	Random Forest Classificatio		00:30:24	AUC 0.6681		DEPLOY	...
✓	tm20190523-201133-BUKPZSIT-BHPVMA Retrain tm20190523-164327-RRCVCUWZ	Random Forest Classificatio		00:25:01	AUC 0.6878		DEPLOY	...
✓	tm20190522-001948-UQEBSRXX-PKVMNM Retrain tm20190521-232724-DQTJGVZI	Random Forest Classificatio		00:38:01	AUC 0.6891		DEPLOY	...

Metadata

What is model metadata?

- Information about an ML model needed for its manageability
  - **Access, Reproducibility, Accountability, Tracking/Monitoring**
  - Includes type, owners, training config, deployment status, performance

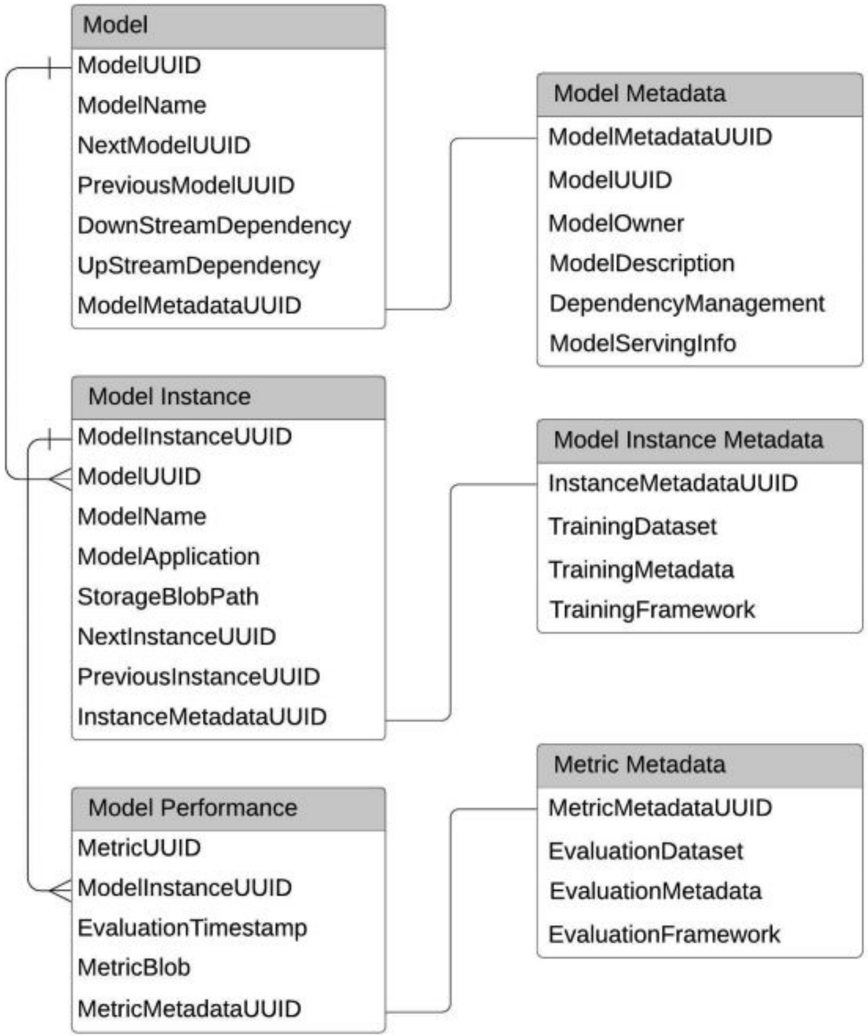
How is it used?

- **Search** for models.
- **Select** and **compare** models based on performance.

Metadata

Example fields that are stored and searched against:

- City
- Product (e.g., UberX, Uber Eats)
- User Tags
- Model Type (e.g., regression, classification)
- Features
- Model Performance (e.g., test AUC, train AUC, serving AUC)



Model Search

- Based on model metadata exported to **Elasticsearch** for indexing
- Common searches
  - **Latest instance** of a model version
  - **Latest instance** of a model version within a **performance constraint**
  - **Best performing instance** of a model version
- Based on the search result
  - **Retrain** a model to improve performance

Listing 1: Model Selection Rule Example

```
{
  "team": "forecasting",
  "uuid": "316b3ab4-2509-4ea7-8025-ca879dac61",
  "rule": {
      "GIVEN": modelName ==
        "linear_regression"
        AND model_domain == "UberX",
      "WHEN":  "metrics["mae"] <= 5",
      "ENVIRONMENT": "production",
      "MODEL_SELECTION":
      "a.created_time > b.created_time"
    }
}
```

Agenda

01 Introduction
02 Design Principles
03 Gallery System
04 Case Studies
05 Lessons Learned and Next Steps
06 Q&A

Case Study - Marketplace Forecasting

Marketplace Forecasting generates spatio-temporal predictions for a variety of applications.

- Prior to Gallery, the Marketplace Forecasting faced 4 major problems:
  - Where to **store** all their models?
  - How to **organize** and **search** those models?
  - How to **track** which model produced a forecast?
  - When to **re-train** and deploy models?

- These 4 problems limited **scalability**, **velocity**, **observability**, and **accuracy**.

Case Study - Marketplace Forecasting

Integration with Gallery has resulted in:

Reduced Deployment Time

The unified model storage interface and data model has reduced manual deployment time from **2 hours to 0**.

Improved Forecasting Accuracy

Dynamic model selection via the Galley Rule Engine has reduced forecast **MAPE by 10+%.**

Case Study - Marketplace Simulation Platform

The Marketplace Simulation Platform[1] hosts a simulated world with driver-partners and riders, mimicking scenarios in the real world.

Model Reusability

Gallery's storage API allowed users to reuse models across multiple simulations, rather than recomputing on-the-fly.

Train/Serving Decoupling

With Gallery, training was decoupled from the simulator leading to **8GB memory reduction** and **one hour CPU** saving per simulation.

1. https://eng.uber.com/simulated-marketplace/

Agenda

01 Introduction
02 Design Principles
03 Gallery System
04 Case Studies
05 Lessons Learned and Next Steps
06 Q&A

Lessons Learned

1. Common ML Tools
    a. Build reusable components that plug into diverse modeling applications.
2. Model Reproducibility
    a. Triaging and debugging issues requires the ability to reproduce models and predictions at any point of the model lifecycle.
3. Tiered Service Offering
    a. Offer modular features that can be incrementally adopted by customers.

Next Steps

- Track the **cost** of training models to compute ROI.
- Model **lineage** and **dependency** tracking.
    - How does the performance of one model impact downstream models?
- Automate model **experimentation** to **shadowing.**
    - Safely deploy new models with ability to rollback.

Thank you!

Questions/Comments?