MLOps - Machine Learning in Production

How to make your models usable by millions of people

By Ammar Siddiqui

UW
DATA SCIENCE
CLUB.

Agenda

- **Why MLOps matters**
  - End-to-end process
  - Career Prospects
- **Model Development**
  - Preprocessing
  - Training
  - Experiment Tracking
  - Model Registry
- **Serving and Inference**
  - Serving Engines and APIs
  - Compilers and Hardware
- **Infrastructure**
  - Cloud and Storage
  - Monitoring, Testing, CI/CD
- **Case Study: Uber ML Platform**
  - System Design overview

UW DSC.

Workshop Goals

- Aimed towards beginner software developers and data scientists that enjoy machine learning!
- Develop a better understanding of various stakeholders, objectives, and processes involved in hosting machine learning models
- Learn about the common tools and technologies used throughout each process
- Learn how to communicate better with people regardless of which side of the pipeline they work on

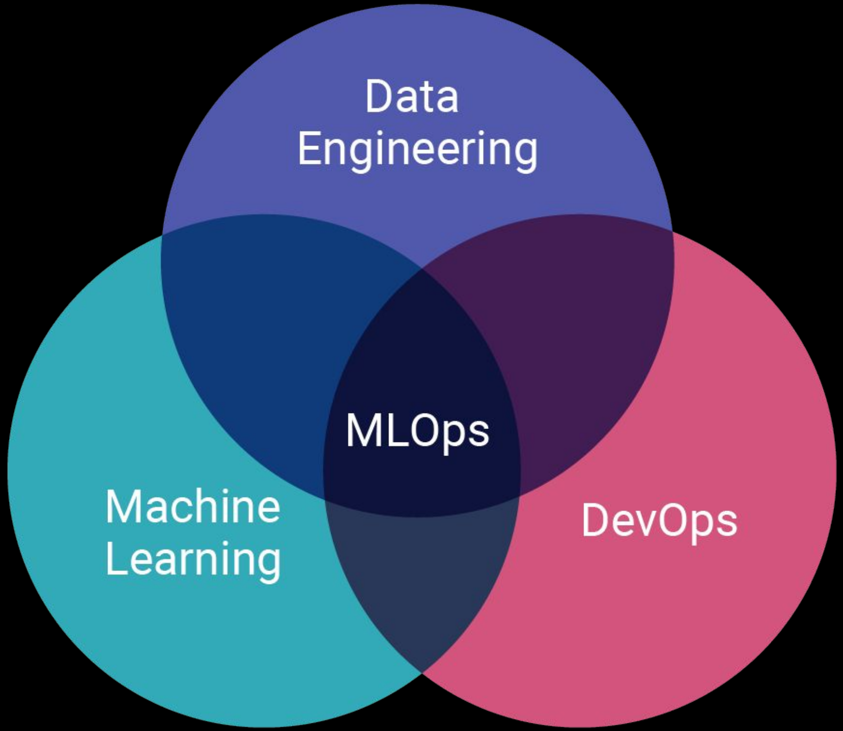
UW
DSC.

What is MLOps?

- Empowering researchers and users to build and use ML models in a better way

- Bridging ML Research and Software Engineering

- Enabling your models to be usable by many people
    - Simple API or Complex Data Pipeline



UW DSC.

Career Prospects

Software Engineer, Model Inference

OpenAI · San Francisco, CA · 1 week ago · 17 applicants

Full-time · Entry level

Software Engineer Perception, Machine Learning Infrastructure

Nuro · Mountain View, CA · 3 weeks ago · Over 100 applicants

$138,225/yr - $207,575/yr · Hybrid · Full-time · Entry level

Software Engineer, AI/ML Platform

Autodesk · Toronto, ON · Reposted 6 days ago · Over 100 applicants

Hybrid · Full-time

Machine Learning Operations Engineer

Sanctuary AI · Vancouver, BC · 3 weeks ago · 94 applicants

On-site · Full-time

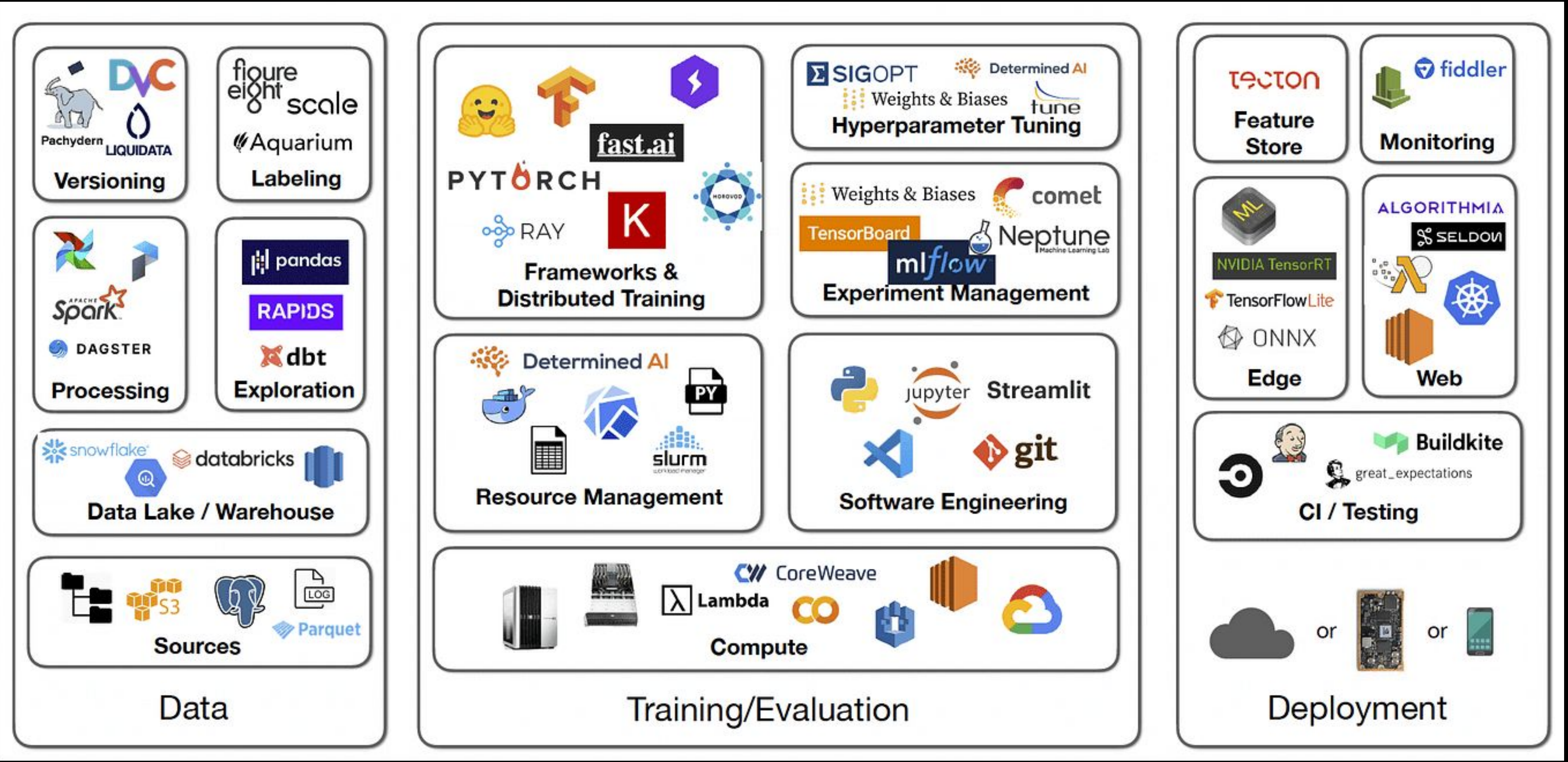
Software Engineer, Serving Infrastructure

Notion · San Francisco, CA · 1 week ago · Over 100 applicants

$160,000/yr - $220,000/yr · On-site · Full-time

MLOps Technologies

Photo Cred: Vishal Rajput

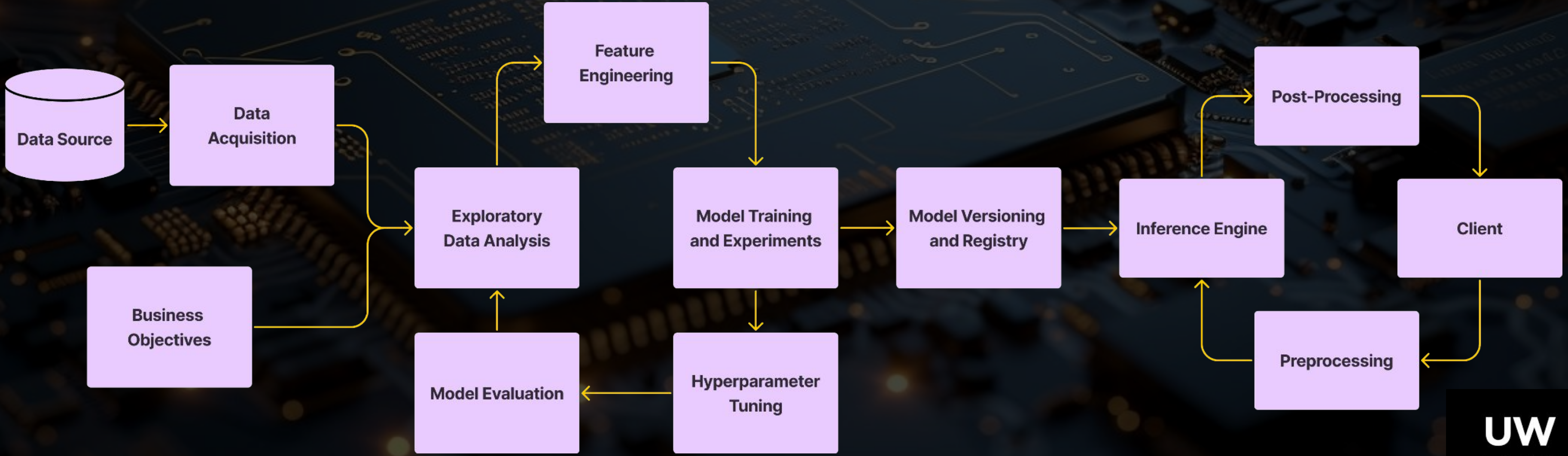


Objectives for each stakeholder

- Product Manager
  - What problem does this model help solve? -> Product or Insights
- Data Engineers
  - What data is needed and how do we get this to the researchers?
- ML Researchers / Data Scientists
  - What is the best model that we can use for this data?
  - What processing steps are required for this model?
- Software Engineers
  - How do I get outputs from this model in a performant manner?
- DevOps / Infrastructure
  - How can I ensure all components of this pipeline are scalable, fault tolerant, secure, and production-grade?

End-to-end process

- Applying software engineering principles to ML applications
    - Frictionless transfer from research to production environment
- Lifecycle -> **Data**, **Training**, **Serving**, and **Infrastructure**



UW DSC.

Section 1

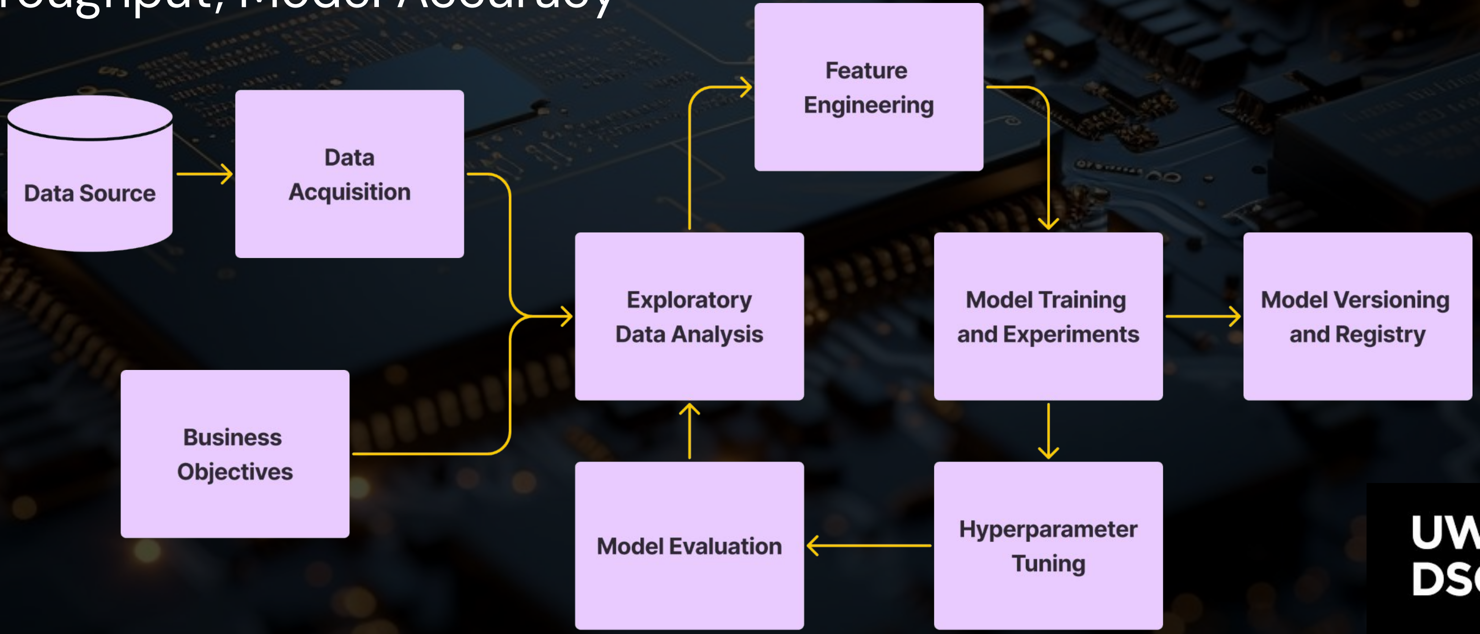
Model Development

UW
DSC.

Model Development

- End goal: Develop the best model and data transformation rules to achieve the business objective

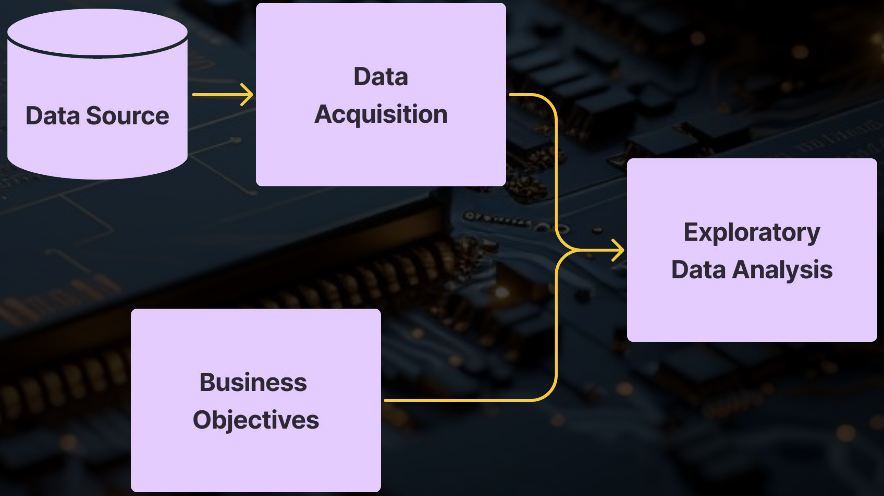
- <u>KPIs</u>: Throughput, Model Accuracy



UW DSC.

Data Ingestion

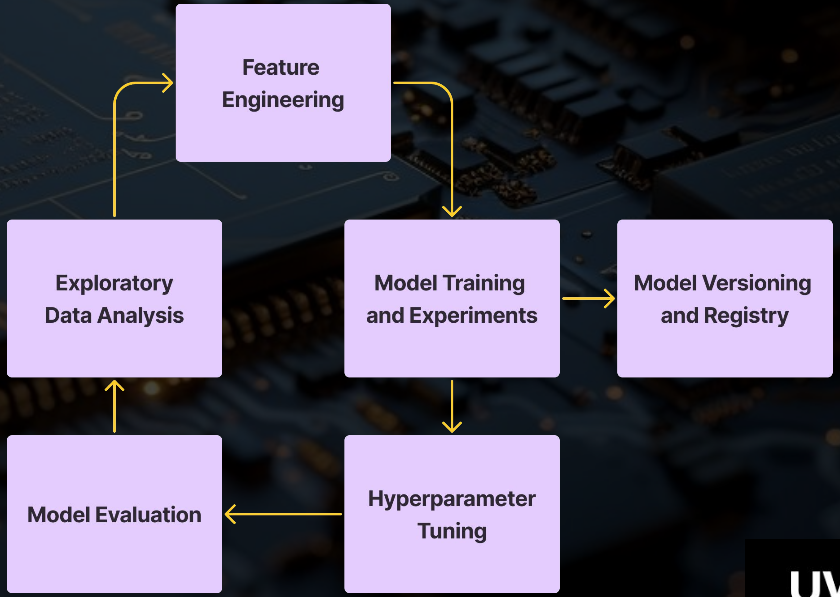
- Determining what data use and how to obtain it
- Data Source(s)
    - API calls, Events, Messages, CSV files, Scraping, etc
- Use-case dependant, usually involve an ETL process
    - Extract, Transform, Load
- <u>Tools</u>: Cassandra, RedShift, Kafka, Spark



UW DSC.

Model Training

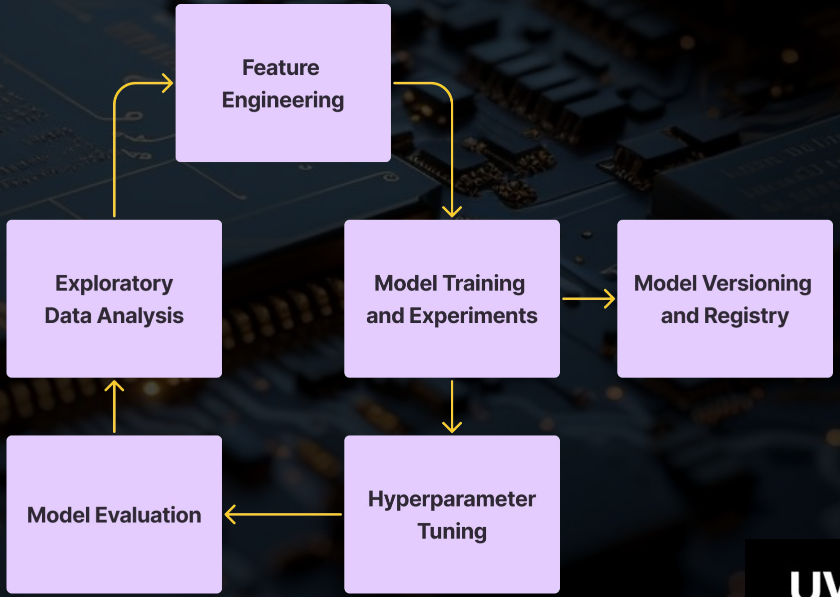
- Facilitating researchers in order to train better models
- Standardize tooling for end-to-end data science
  - EDA, Feature engineering, Training, Tuning, Evaluation
- Improve how fast the models run during training
- Tools: PyTorch, JAX, TensorFlow, Scikit-learn, Pandas, Optuna



UW DSC.

Distributed Training

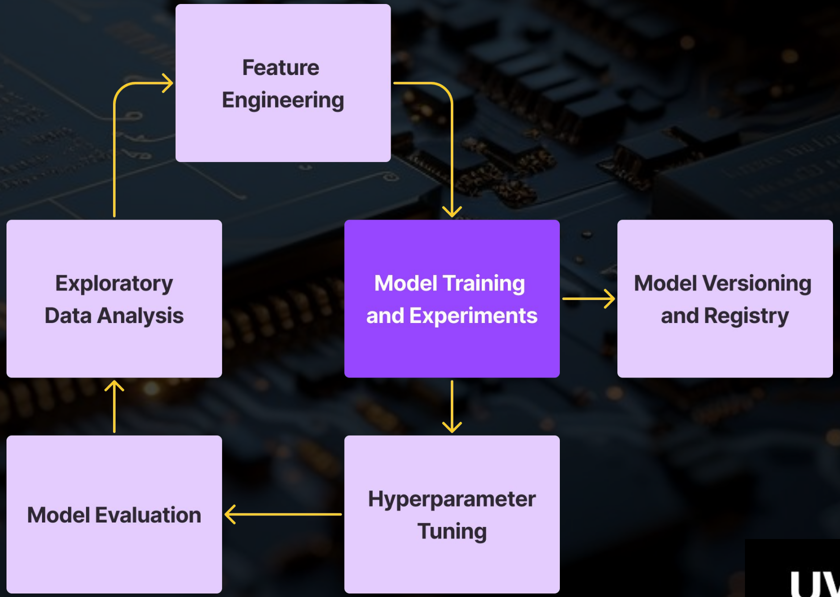
- For large models, you can schedule training runs on a GPU cluster
- Split up training across multiple devices / servers
  - Data-parallel
  - Model-parallel
    - Tensor
    - Pipeline
- Fault Tolerance -> What if a node corrupts during training?
- <u>Tools</u>: Horovod, DeepSpeed



UW DSC.

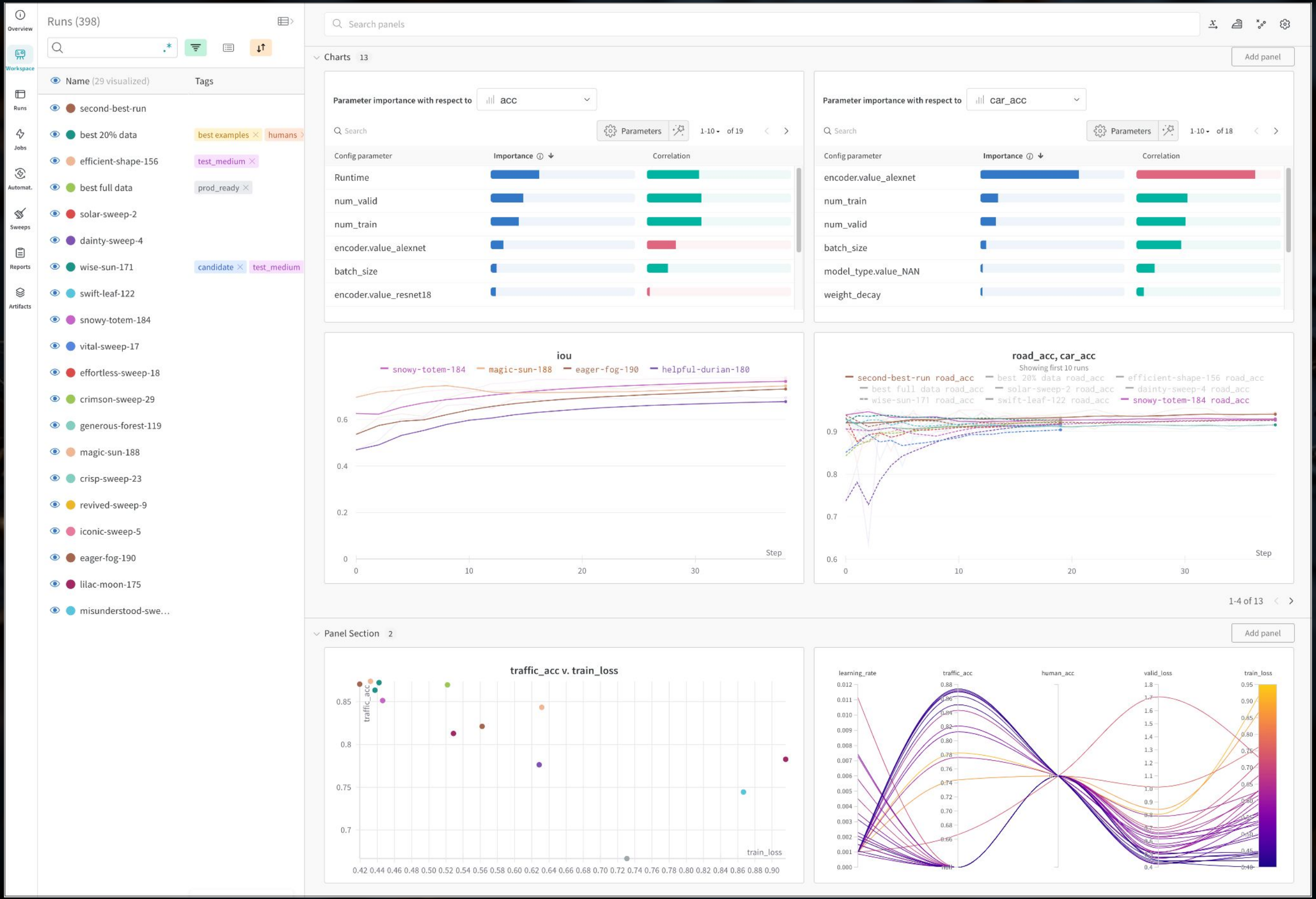
Experiment Tracking

- Tracking experiments done by researchers on new models / scripts
- Stores information about each experiment as well as associated model metadata
  - Jupyter Notebook Versioning
- Includes visualization tools for different metrics based on each epoch of the experiment
- Tools: Weights and Biases (W&B), Comet ML, MLflow, Sagemaker



UW DSC.

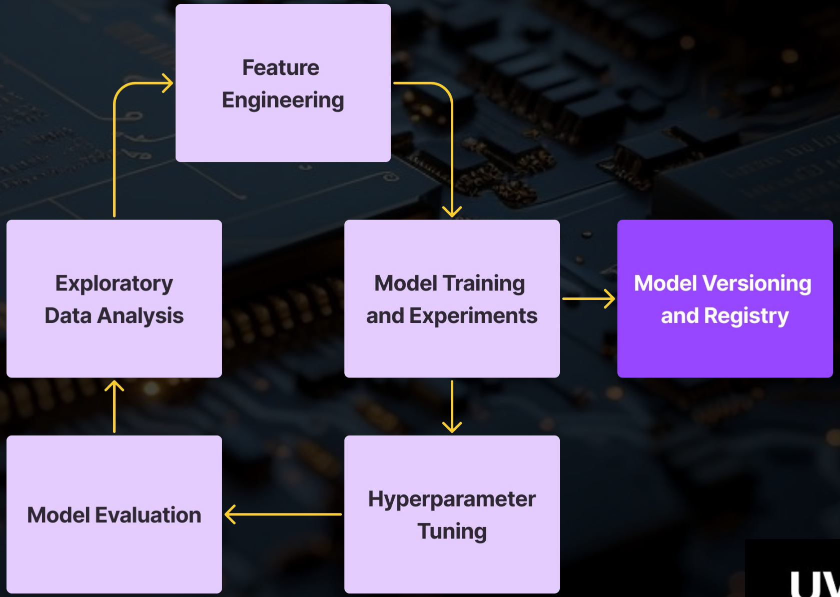
Experiment Tracking - Weights and Biases



UW DSC.

Model Registry / Versioning

- Essentially like GitHub -> version control for your models
- After you are happy with an experiment, publish to the registry
- Acts as a database for models, letting you access metadata, artifacts, and experiment results
- <u>Tools</u>: MLflow Artifact Store, Neptune.ai, AWS S3, WandB



UW DSC.

Section 2

Model Deployment and Inference

10 minute break!

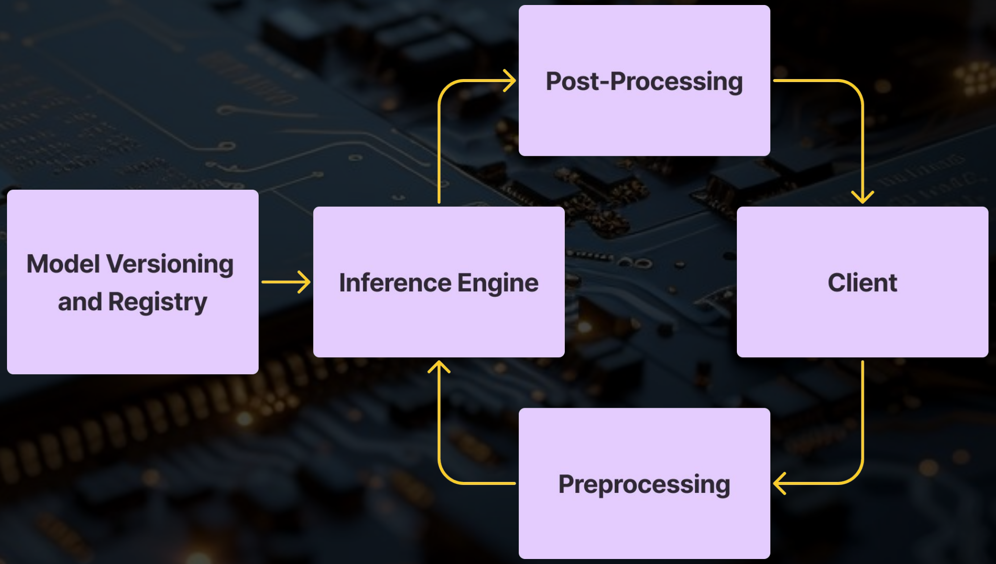
UW
DSC.

Model Deployment

- Delivering Model Outputs / Predictions to your users

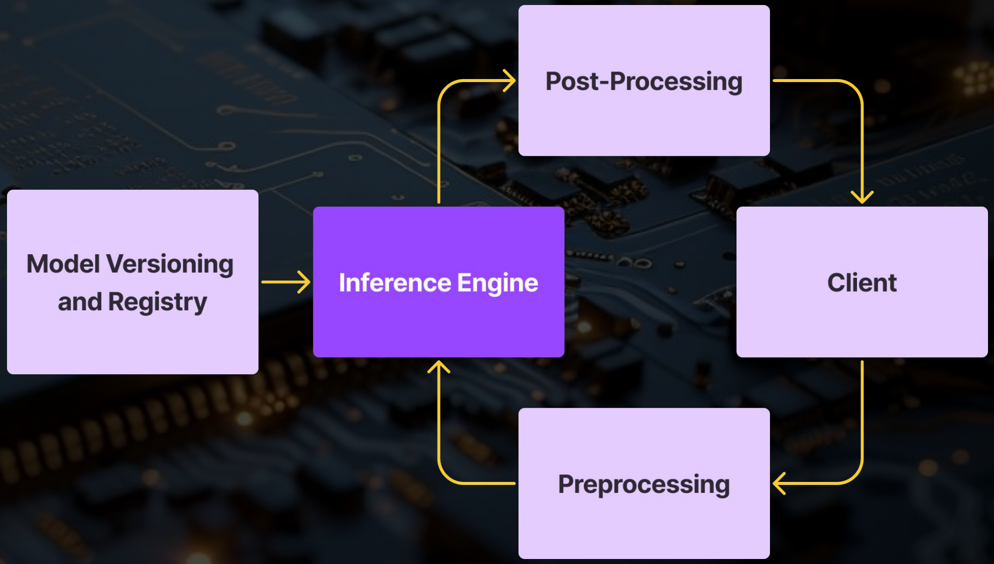
- Takes requests by a client, preprocesses request data, generates a prediction, and serves it back to the client

- KPIs: Inference Latency, Model Decay



Serving / Inference Engine

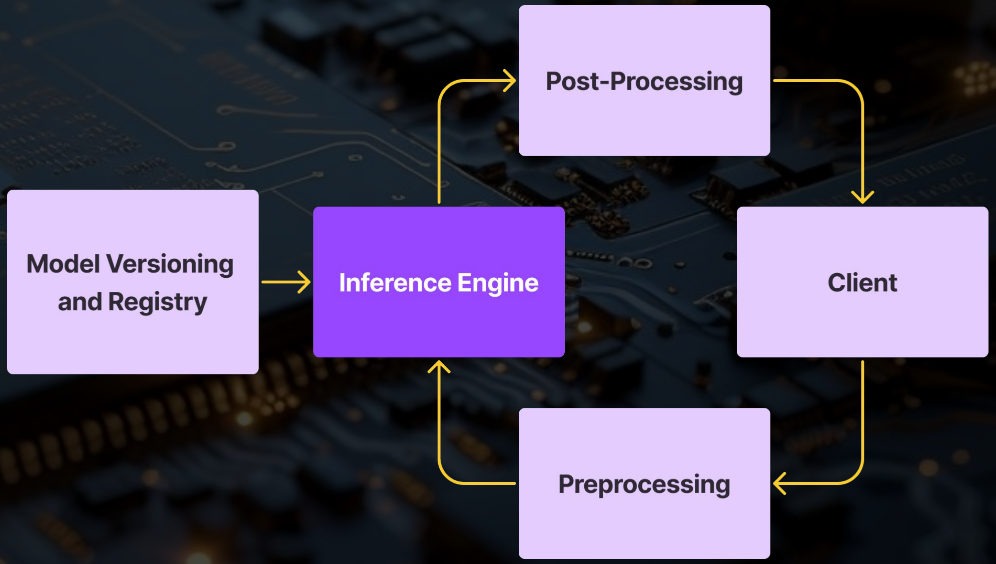
- Runtime for your model that can be scaled to many users
- Runs model in an optimized environment to generate predictions (no back-prop)
- Loads model weights into memory
- <u>Tools</u>: TF-Serving, Nvidia Triton, TorchServe, Seldon, Ray Serve



UW DSC.

Compilers and Runtimes

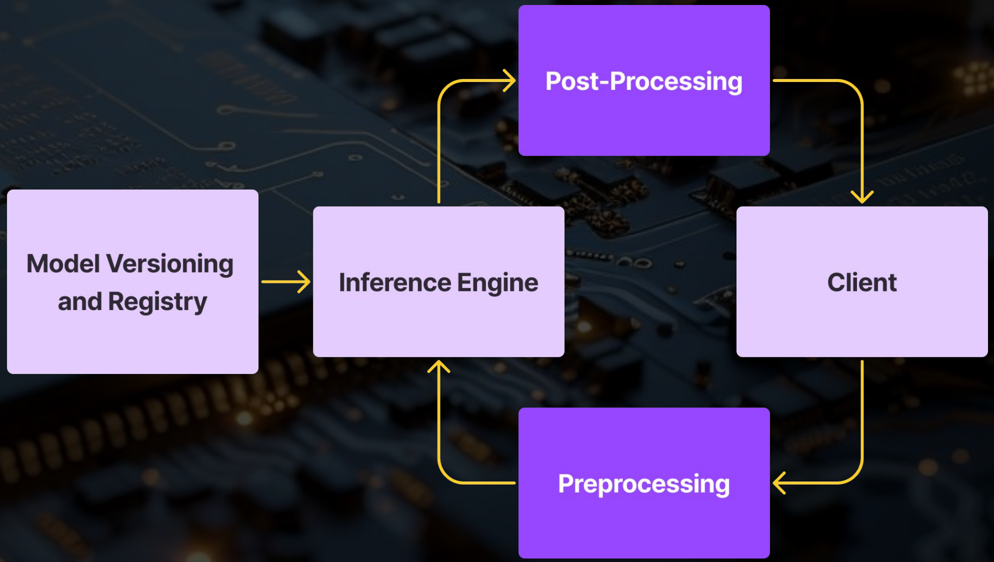
- Your ML code can be compiled into a faster format
- Operator Fusion, Kernel tuning, Quantization, CUDA
- Researchers will develop in their favourite frameworks, then model gets compiled for inference or training
    - Specialized Hardware
- <u>Tools</u>: XLA, TVM, Glow, TensorRT



UW DSC.

Data Processing (Pre and Post)

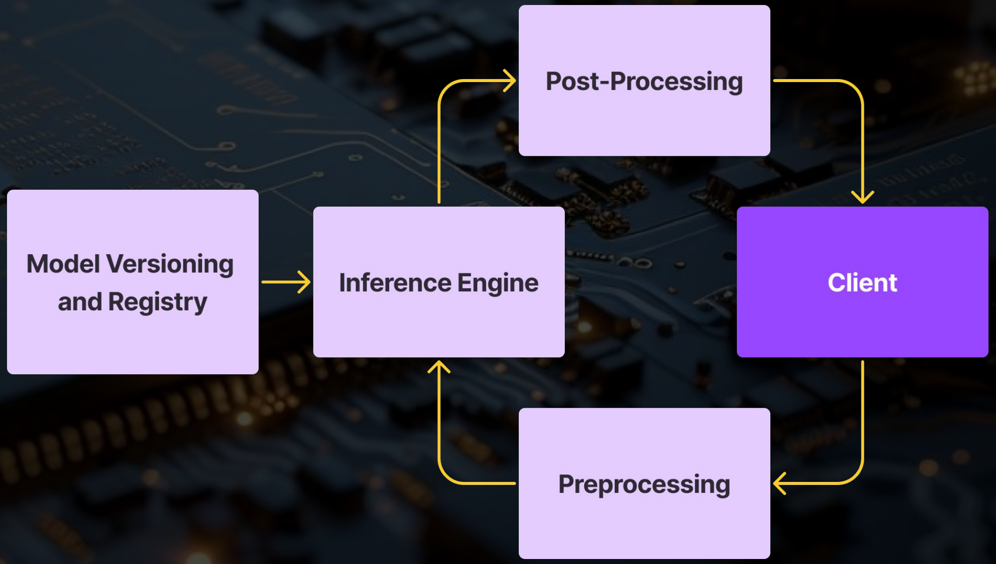
- Data Engineering -> Transform data and make it usable for the model
- Follows steps determined by research and engineering teams in order to get data to and from the model appropriately
- Post-process data from model to user-interpretable format
- Tools: Spark, Pandas, etc



UW DSC.

Client

- Entrypoint enabling users to make requests for inferences
- Can take many forms, usually a RESTful API or message queue that sends requests to be preprocessed
- Receives responses in an interpretable format
- Examples: OpenAI or Cohere API



UW DSC.

Section 3

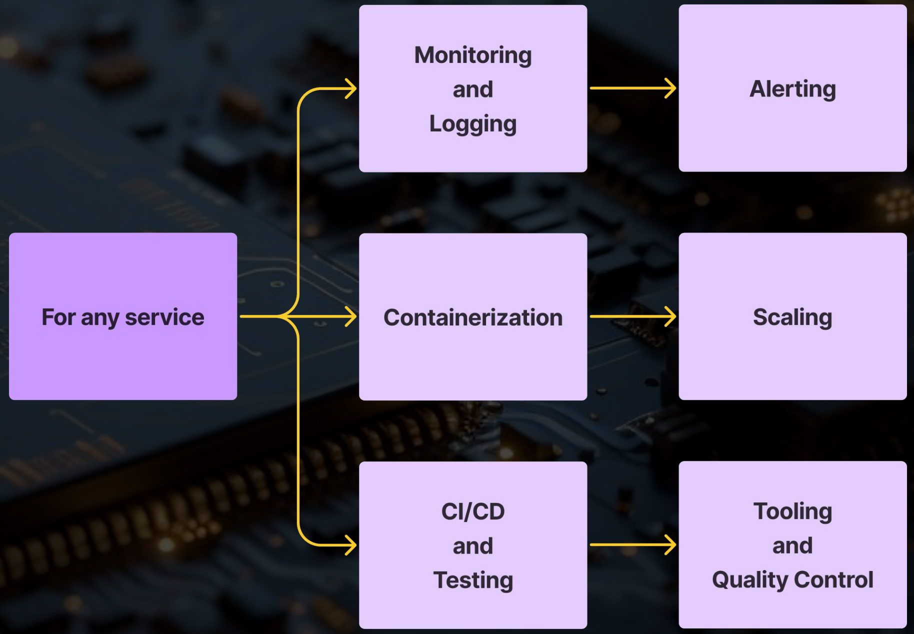
Infrastructure and Scalability

UW
DSC.

Infrastructure and Scaling

- There is a huge difference between POC / Research Code and production-level code

- Having a resilient infrastructure enables you to:
  - Scale to more user demand
  - Avoid bugs sneaking into a production release
  - Use better tooling and standard practices as a developer / researcher



UW DSC.

Batch vs Real-time Processing

- Real-time / Stream / Online: processing data in real-time, as soon as it arrives
  - Usually for inference
  - Examples: Spark Streaming, Kafka

- Batch / Offline: processing a set of data, usually in scheduled intervals or after an appropriate amount (batch size) is collected
  - Usually for training
  - Examples: Hadoop MapReduce, Spark, Polars, Pandas

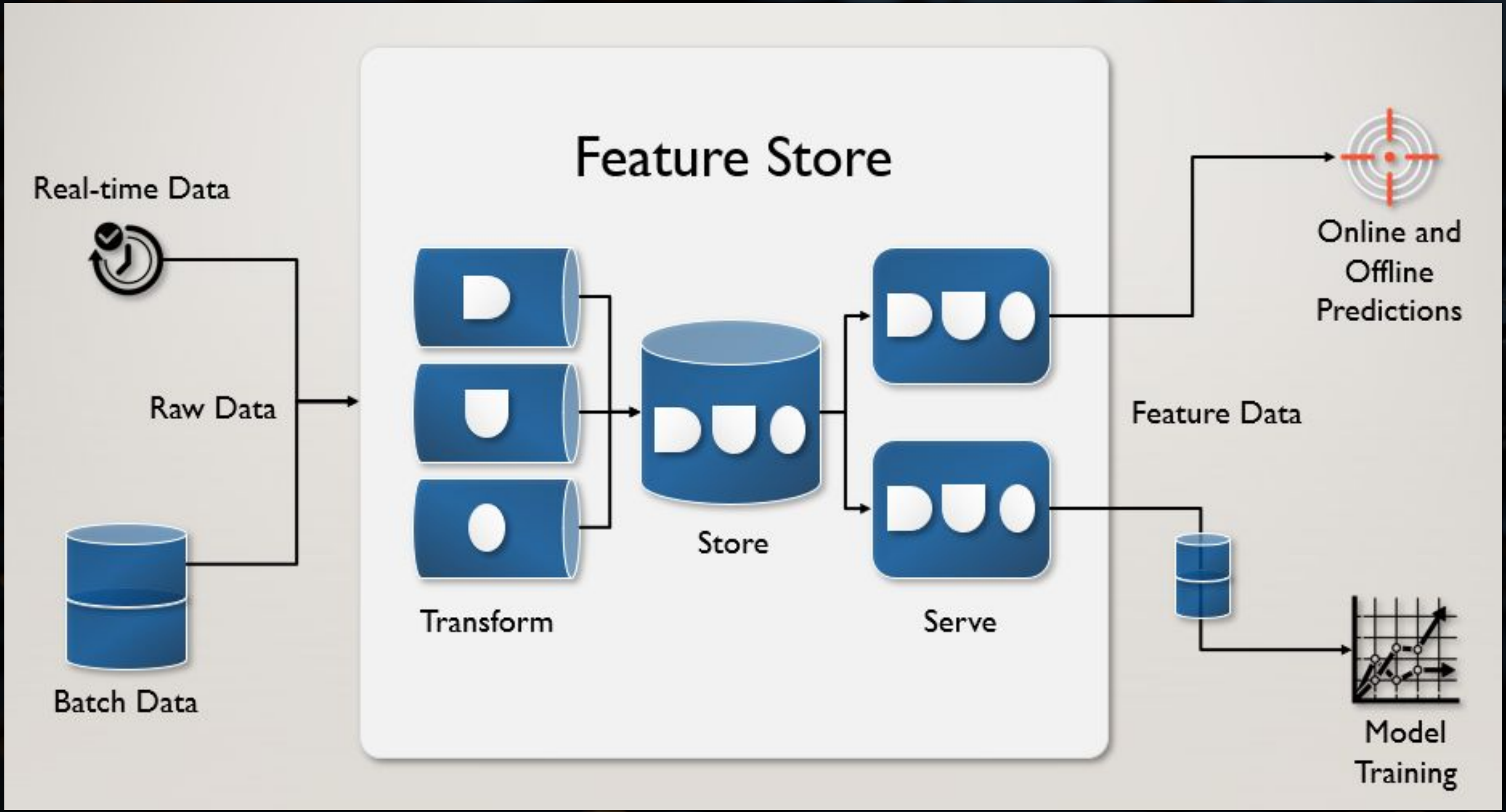
UW
DSC.

Feature Stores and Caching

- Transformation Rules -> Preprocessing and EDA / Feature Engineering
- One central system for storing any data used for training or for inference
- Applies the same rules for incoming data in order to keep consistent with deployed models and models under development

UW DSC.

Feature Store Example



UW DSC.

Model Decay and Drift

- Model Performance can deteriorate over time, we need an indicator for deployed models that can help detect this

- Try training a model on data from 6 months ago, and test on recent data <- from Chip Huyen

- Example: stock prediction model would decay in performance over time as new market data comes in
    - A model with training data from 2010-2019 (bull run) wouldn't be as good as a model with knowledge of current market (recession)

UW
DSC.

Section 4

UBER Machine Learning Platform Case Study

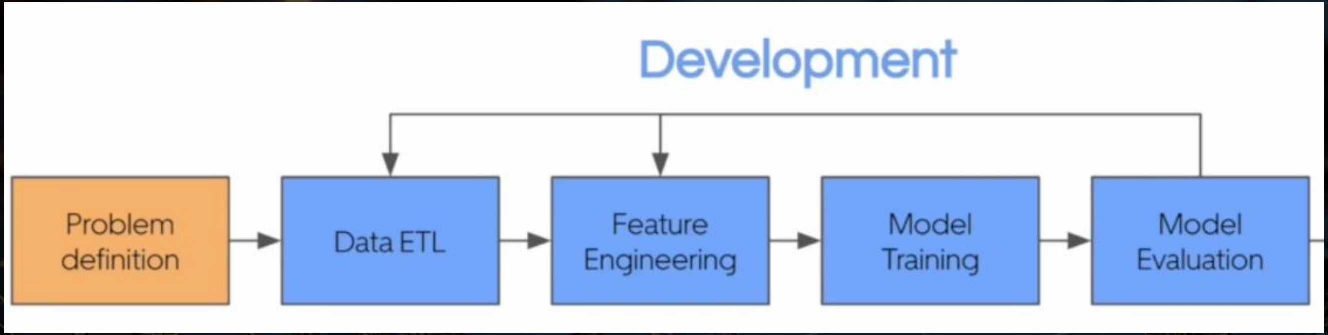
UW
DSC.

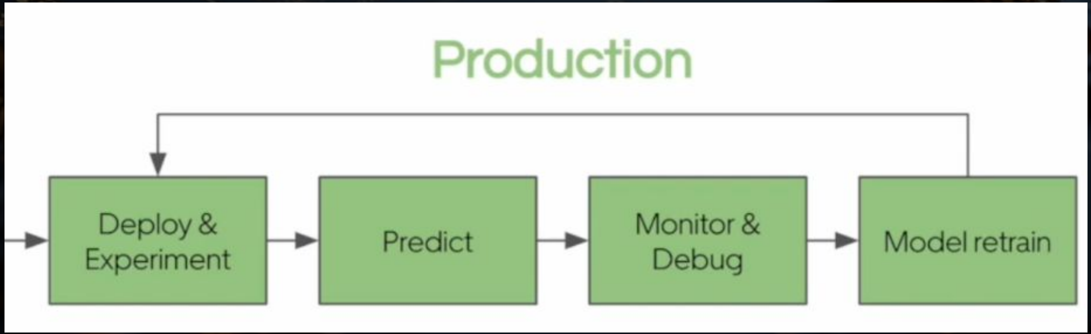
ML Use Cases at Uber

- Estimated Time of Arrival
- Driver pricing based on demand
- Recommender and Search systems
  - Uber Eats recommendations
  - Product Recommendations
- Driver mask and helmet verification

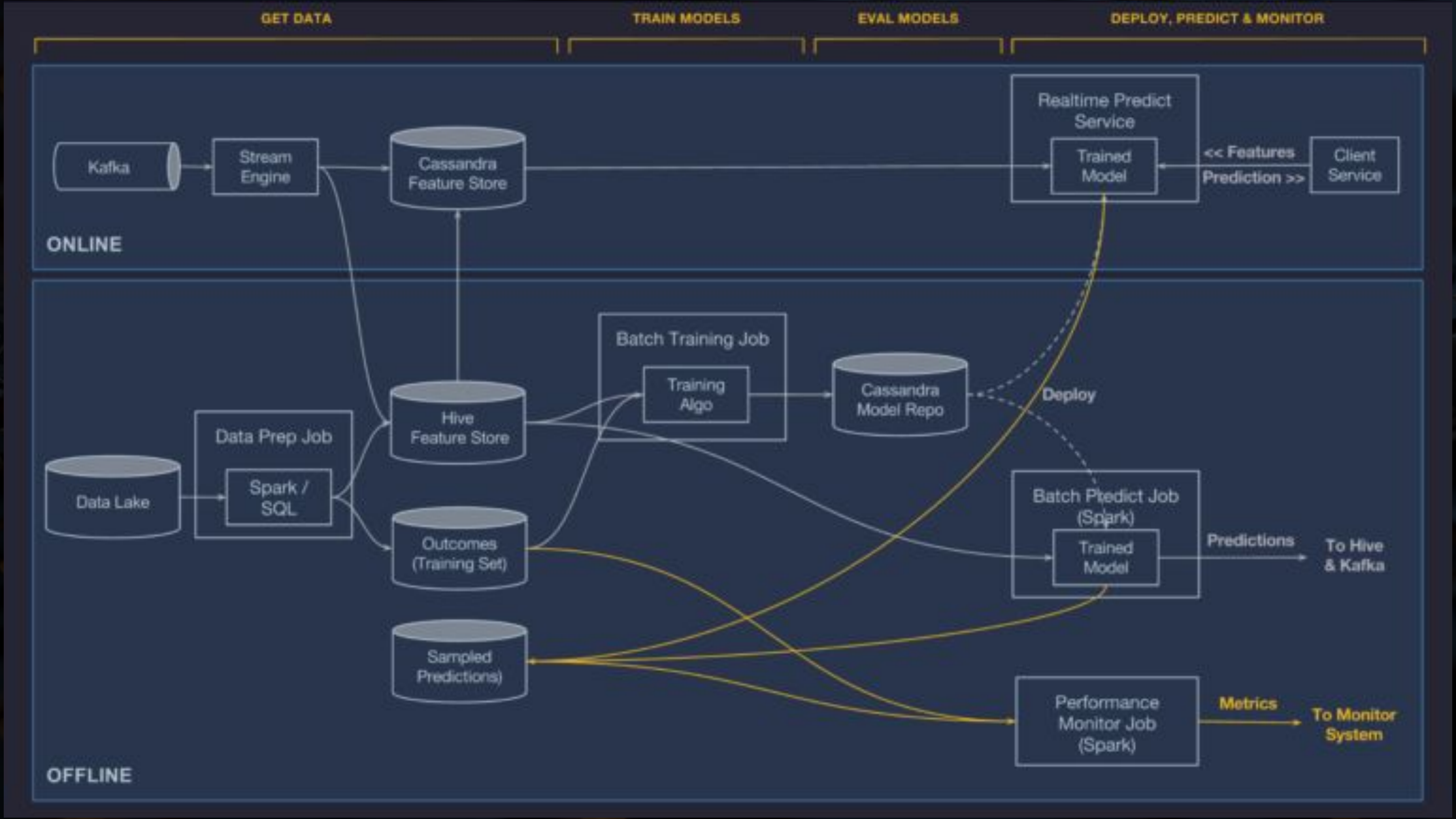
UW
DSC.

MLOps at Uber





Case Study: Uber's ML Platform



Extra components needed for LLMs

- Large multi-GPU Inference Server
  - Fine-tuning experiments on new data

- Response toxicity classification
  - Reinforcement Learning from Human Feedback
  - Responsible AI

- Prompt Engineering and Retrieval Augmented Generation (RAG)

Thank you for listening!

UW
DSC.