CS4262/5462 Tutorial

ML Operations

TA: Noppanat Wadlom
Email: noppanat@u.nus.edu

Tutorial Files

- Go to the course website (https://mlsys.io/MLsys_25Sem2.html) and download the zip file.
- Unzip it into “cs4262_5462_mlops_tutorial”, containing
  - cs4262_5462_mlops_tutorial.ipynb
  - serve.py
  - requirements.txt
  - make_submission.sh
- You may run locally or use Google Colab.

Outline

01
MLOps

02
Model development

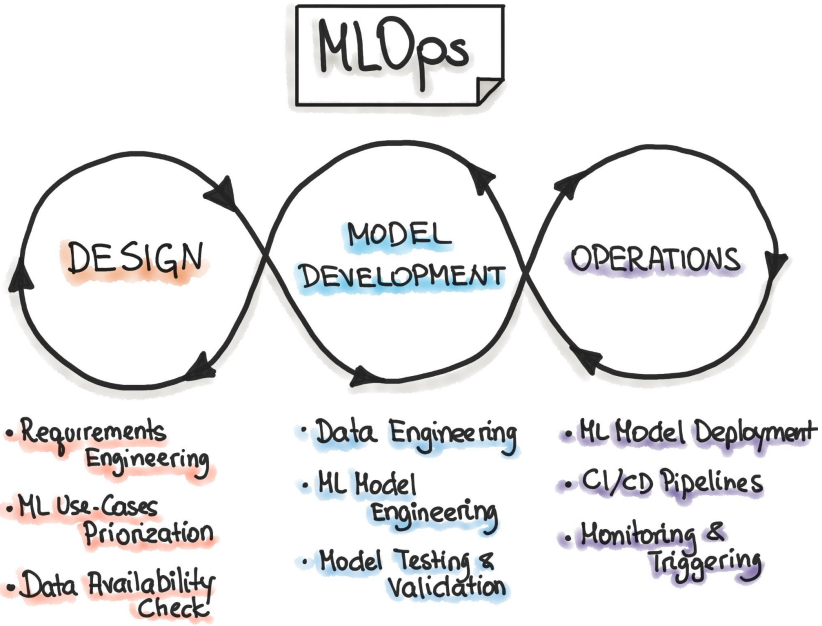
03
Deployment and monitoring

04
Tutorial guide

05
Submission

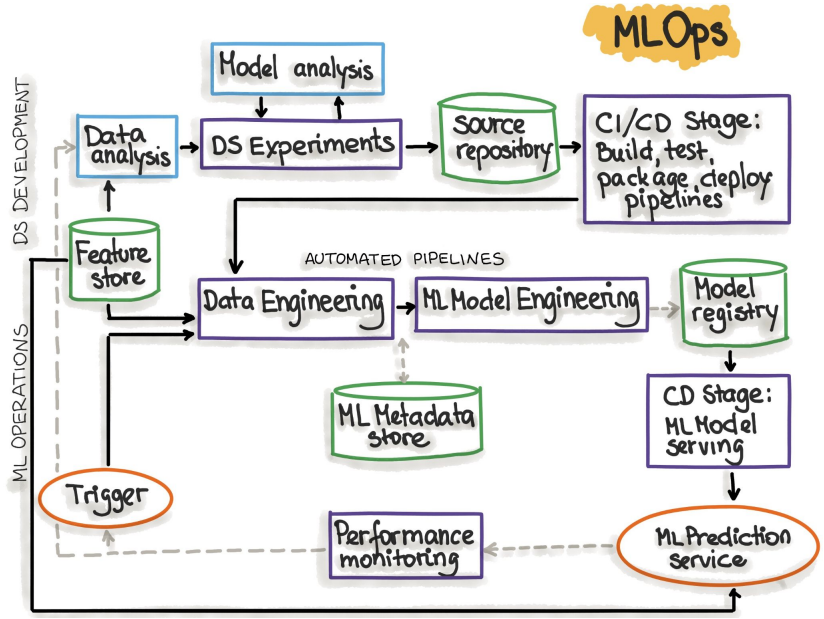
MLOps

- Process of developing and deploying ML models in a CI/CD environment in real-world production.



MLOps

- Automation and integration with other services.
  - Continuous Integration, Delivery, Training, and Monitoring

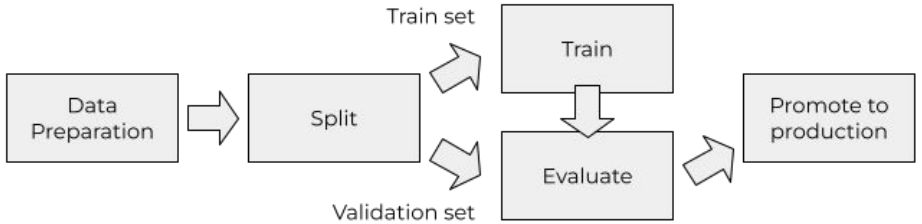


We will focus on model development, deployment, and monitoring.

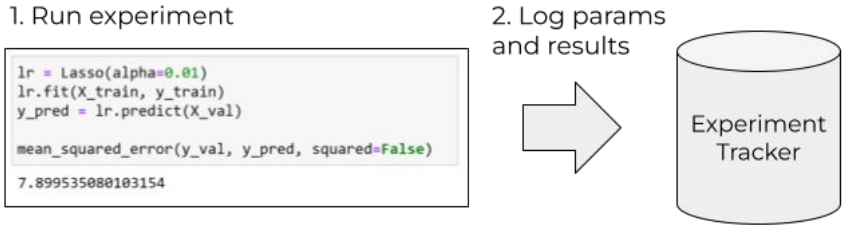
Model Development

- Model development involves
  - Training pipeline (data engineering, model training, and validation)
  - Experiment tracking

Training pipeline

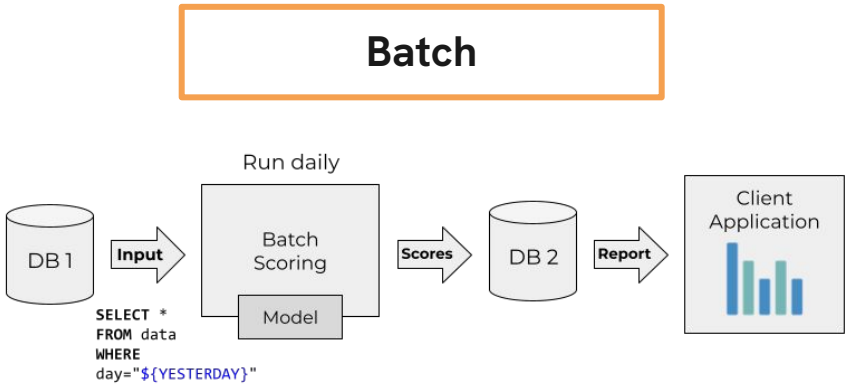


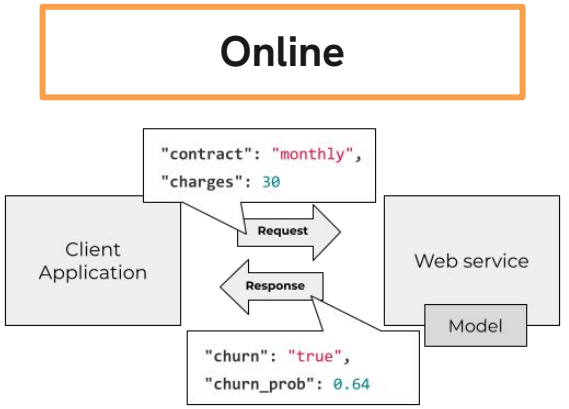
Experiment tracking



Deployment and Monitoring

- The trained and validated model is then deployed in production, integrating with the systems and APIs.
- **Modes of deployment:**
  - Batch and online (web service and streaming)

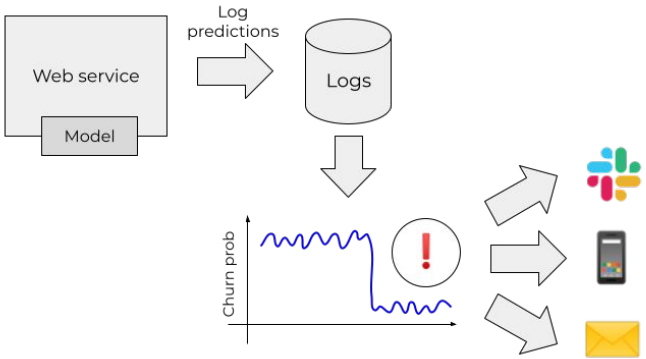


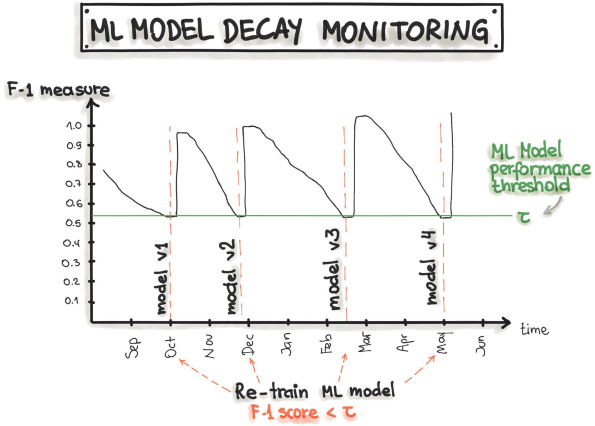


e.g., model inference server

Deployment and Monitoring

- Deployment environments are highly dynamic.
- Deployed models are continuously monitored
  - via system metrics and **prediction logs**
  - for dependency changes, computational performance, model quality drift, **data drift**, etc.





Tutorial Guide

In this tutorial, we will develop and deploy a model to predict taxi trip duration using the NYC TLC Trip Records.

- **Part 1: Data Preparation**
  - Load, prepare, and explore the taxi trip dataset.
- **Part 2: Model Development**
  - Train and evaluate the model with hyperparameter tuning experiments.
- **Part 3: Model Deployment and Monitoring**
  - Deploy the model with an inference server and monitor performance.

More detailed instructions are in the provided Jupyter notebook.

Tutorial Guide

Part 1: Data Preparation

- We will use a subset of data from the NYC TLC Trip Records (data dictionary).
- To prevent leakage, we will use
  - Jan 2022 for training
  - Feb 2022 for validation
  - Mar 2022 for testing (simulating client requests)

```
[ ]  train_df.head()
```

Python

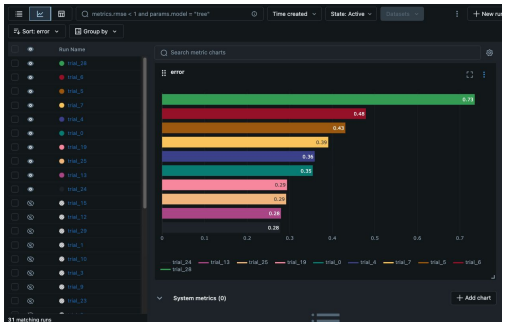
	VendorID	store_and_fwd_flag	RatecodeID	PULocationID	DOLocationID	passenger_count
54553	2	N	1.0	166	74	2.0
16955	2	N	1.0	25	66	1.0
10792	2	N	5.0	49	49	1.0
41227	2	N	1.0	95	197	3.0
45340	2	N	1.0	74	75	1.0

Tutorial Guide

Part 2: Model Development

- We will tune the hyperparameters and train a RandomForestRegressor model to predict trip duration.
- We will use Optuna for hyperparameter optimization and MLflow for experiment tracking.

```python
import optuna

def objective(trial):
    x = trial.suggest_float('x', -10, 10)
    return (x - 2) ** 2

study = optuna.create_study()
study.optimize(objective, n_trials=100)

study.best_params  # E.g. {'x': 2.002108042}
```



OPTUNA

mlflow

Tutorial Guide

Part 2: Model Development

- **(Task 1) 2.1 Hyperparameter tuning (40pt)**
  - Adjust the hyperparameter suggestion range
  - Adjust the number of study trials
  - Capture one or more screenshots of MLflow's experiment tracking UI
  - Describe the best model's input features and parameters in the report.

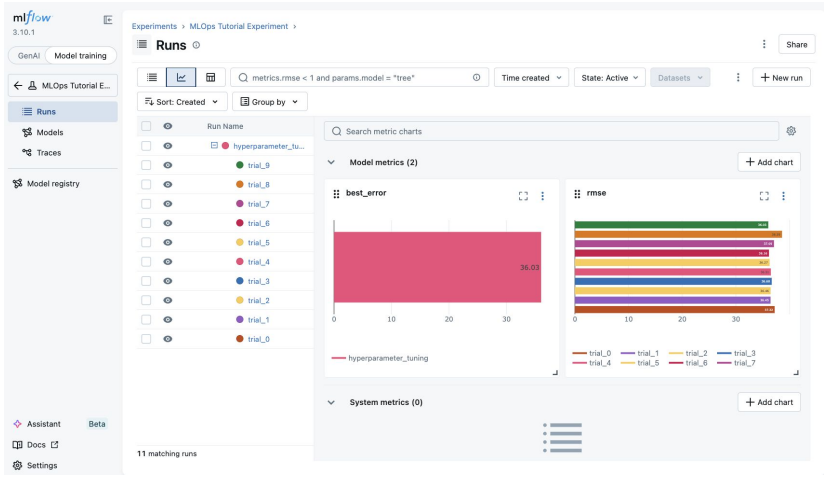
```python
def objective(trial: optuna.Trial) -> float:
    with mlflow.start_run(run_name=f"trial_{trial.number}", nested=True) as child_run:
        # TODO: Suggest hyperparameters for the trial
        # ═══ Start of your code ═══
        selected_features = ["PULocationID", "DOLocationID", "trip_distance"]
        max_depth = trial.suggest_int("max_depth", 2, 8)
        n_estimators = trial.suggest_int("n_estimators", 10, 30, step=10)
        max_features = trial.suggest_float("max_features", 0.2, 1.0)
        params = {
            "max_depth": max_depth,
            "n_estimators": n_estimators,
            "max_features": max_features,
        }
        # ═══ End of your code ═══
```

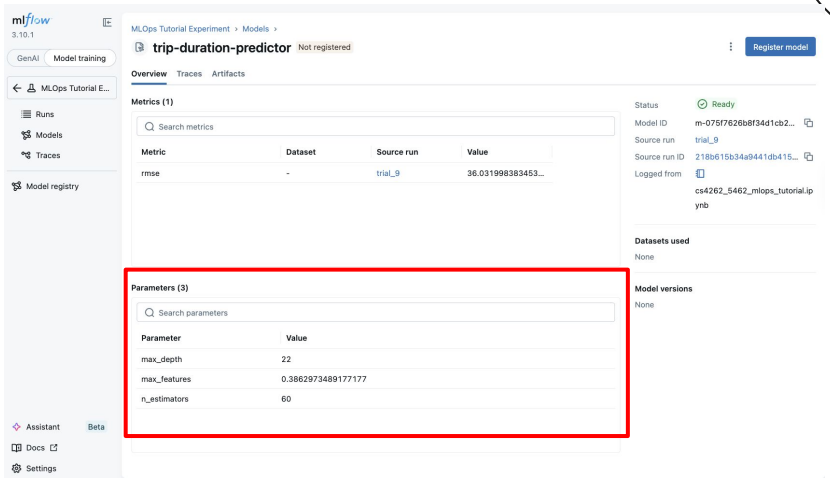
```python
with mlflow.start_run(run_name="hyperparameter_tuning") as run:
    # Log the experiment settings
    n_trials = 1  # TODO: Adjust the number of trials as needed
    mlflow.log_param("n_trials", n_trials)

    # Optimize the objective function
    study = optuna.create_study(direction="minimize")
    study.optimize(objective, n_trials=n_trials)
```

Tutorial Guide

Part 2: Model Development

- **(Task 1) 2.1 Hyperparameter tuning (40pt)**
  Go to "http://127.0.0.1:5001" or the URL that you set for the MLflow server.



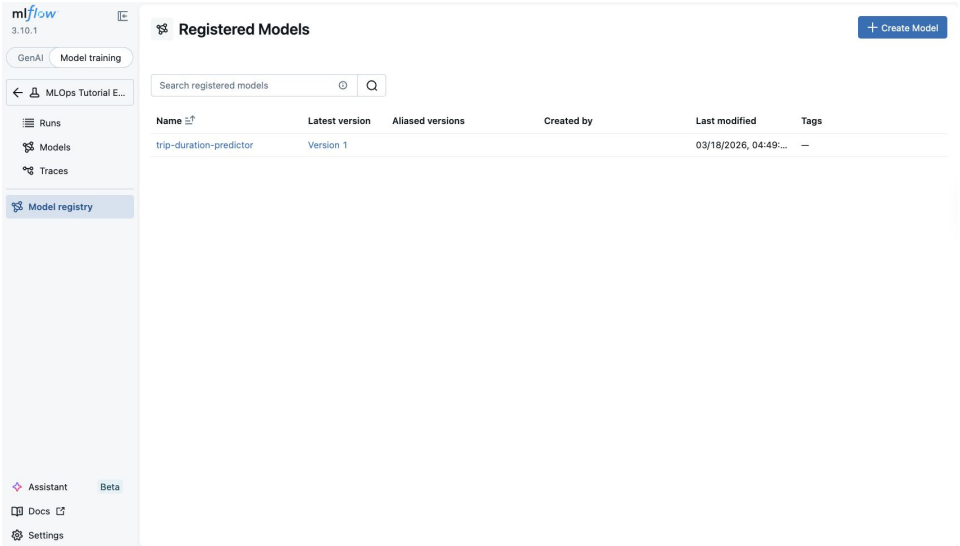


***At least 10 trials.**

Tutorial Guide

Part 2: Model Development

- **(Task 2) 2.2 Register the best model (10pt)**
  - Capture a screenshot of MLflow's model registry UI.



Tutorial Guide

Part 3: Model Deployment and Monitoring

- **(Task 3) 3. Model Deployment and Monitoring (50pt)**
  - Implement model prediction and logging in the “/invocations” endpoint in “serve.py”.
  - Implement the log parsing logic in the “/monitor” endpoint in “serve.py”.
  - Generate the monitoring report (“monitoring_report.html”).

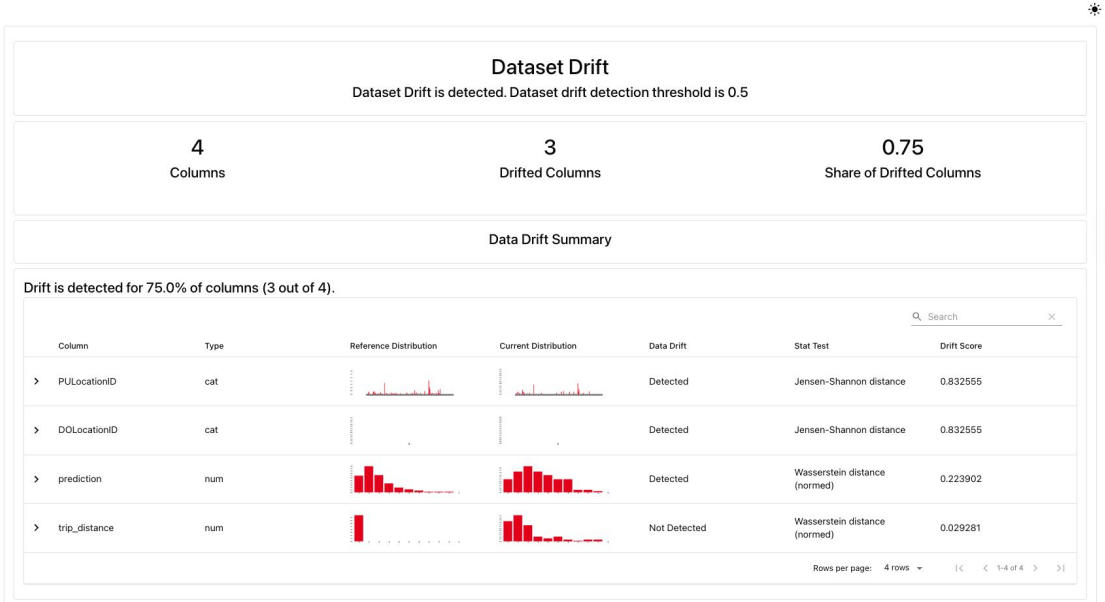
```python
@app.post("/invocations")
def invocations(
    request: Request, pred_request: PredictionRequest
) -> PredictionResponse:
    features = _prepare_features(
        pred_request.inputs,
        request.app.state.feature_names,
        request.app.state.categorical_features,
    )
    log_file_path: Path = request.app.state.log_file_path
    model = request.app.state.model

    Implement with Codex
    # TODO: Implement model prediction and logging
    # Your code should:
    # 1. Use the loaded model to predict on the prepared features.
    # 2. Log the input features and predictions to `log_file_path`
    #    (in any format you like, e.g., JSONL).
    # ═══ Start of your code ═══
    predictions = []
    raise NotImplementedError()
    # ═══ End of your code ═══
```

```python
@app.get("/monitor")
def monitor(request: Request) -> HTMLResponse:
    log_path = request.app.state.log_file_path
    if not log_path.exists() or log_path.stat().st_size == 0:
        return HTMLResponse("<p>Not enough data yet.</p>")

    # Load the reference data
    reference_df = request.app.state.reference_df
    data_definition = DataDefinition(
        categorical_columns=request.app.state.categorical_features,
    )
    reference_dataset = Dataset.from_pandas(
        reference_df, data_definition=data_definition
    )

    Implement with Codex
    # TODO: Load and parse the logged prediction data into `current_dataset`
    # ═══ Start of your code ═══
    current_dataset: Dataset = None
    raise NotImplementedError()
    # ═══ End of your code ═══
```

Tutorial Guide

Part 3: Model Deployment and Monitoring

- (Task 3) 3. Model Deployment and Monitoring (50pt)



Dataset Drift

Dataset Drift is detected. Dataset drift detection threshold is 0.5

4	3	0.75
Columns	Drifted Columns	Share of Drifted Columns

Data Drift Summary

Drift is detected for 75.0% of columns (3 out of 4).

Search

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score
PULocationID	cat			Detected	Jensen-Shannon distance	0.832555
DOLocationID	cat			Detected	Jensen-Shannon distance	0.832555
prediction	num			Detected	Wasserstein distance (normed)	0.223902
trip_distance	num			Not Detected	Wasserstein distance (normed)	0.029281

Rows per page: 4 rows 1-4 of 4

EVIDENTLY AI

Evidently: an open-source Python library to evaluate, test, and monitor ML and LLM systems.

Submission

- Three tasks
  - 2.1 Hyperparameter tuning (40pt)
  - 2.2 Register the best model (10pt)
  - 3. Model Deployment and Monitoring (50pt)
- Create a zip file named “submission.zip” containing
  - **cs4262_5462_mlops_tutorial.ipynb**: The notebook file with all cell outputs.
  - **serve.py**: The fully implemented serving script.
  - **development_report.pdf**: A brief report (1-2 pages) containing
    - Screen capture(s) of MLflow’s experiment tracking UI
    - Description of the best model’s input features and hyperparameters
    - A screen capture of MLflow’s model registry UI
  - **serving.log**: The serving log generated by serve.py.
  - **monitoring_report.html**: The monitoring report generated by Evidently.
- Submit “submission.zip” to Canvas.

References

- MLOps principles: https://ml-ops.org/content/mlops-principles
- MLOps components: https://www.geeksforgeeks.org/machine-learning/mlops-components-machine-learning-life-cycle/
- MLOps in 10 minutes: https://datatalks.club/blog/mlops-10-minutes.html
- IBM MLOps: https://www.ibm.com/think/topics/mlops#1703063717
- Lightweight MLOps Zoomcamp: https://github.com/alexeygrigorev/lightweight-mlops-zoomcamp

THANK YOU