UNIVERSITÀ DEGLI STUDI DI TRIESTE

Class 14: MLOps Practical

Master Course:

Data-driven Systems Engineering (ML Operations)

440MI and 305SM

Today's goal:

- Practical Exercise:

  Building an End-to-End MLOps Pipeline

  MLOPS Strategies

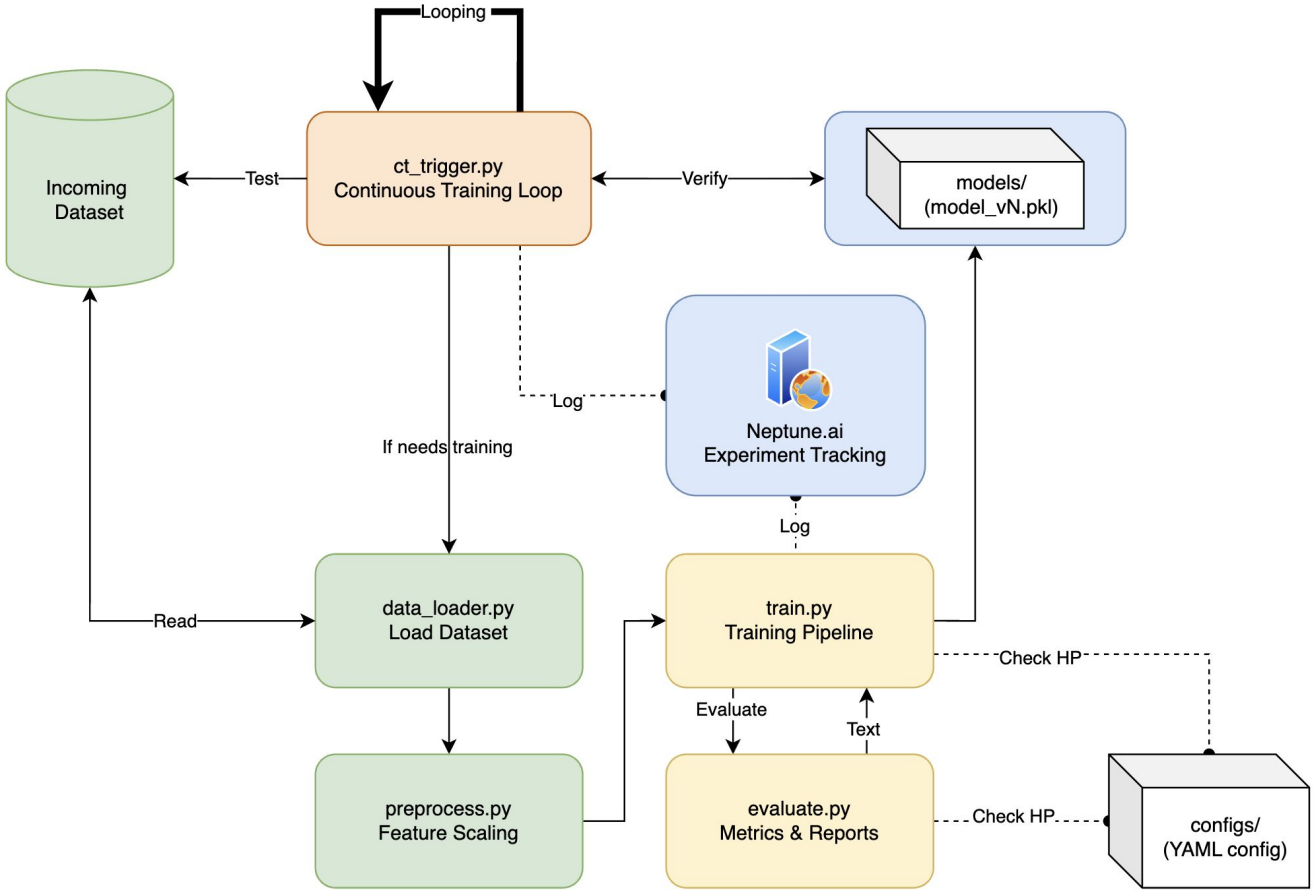
Objective

Implement a full MLOps workflow including:

- Modular ML development
- Experiment tracking with Neptune.ai
- Simple automated model versioning
- Continuous Training (CT) with drift detection
- Simulation of real-world ML lifecycle

Objective

Implement a full MLOps workflow (overview):



Project Structure

configs/

- YAML configuration (model params, CT settings, Neptune keys)

src/

- `train.py`: training + evaluation + versioning + Neptune logging
- `ct_trigger.py`: continuous training + drift detection + triggers retraining
- `data_loader.py`: loads dataset
- `preprocess.py`: handles feature scaling
- `evaluate.py`: computes metrics
- `utils.py`: versioning utilities

models/

- Automatically generated model artifacts (`model_v1.pkl`, `model_v2.pkl`, etc.)

Project Structure

What is YAML used for?

- Centralizes all pipeline parameters
- Makes the training and CT pipeline **reproducible and configurable**
- Avoids hard-coded values in Python scripts
- Facilitates experimentation by changing only a config file

configs > ! config.yaml

```yaml
model:
  type: random_forest
  n_estimators: 120
  max_depth: 5
```

```python
import yaml

def load_config():
    with open("configs/config.yaml", "r") as f:
        return yaml.safe_load(f)

def train():
    config = load_config()

    # Initialize Neptune run
    run = neptune.init_run(
        project=config["neptune"]["project"],
        api_token=config["neptune"]["api_token"]
    )
    run["config"] = config
```

Project Structure

What is Neptune?

Neptune is a **Metadata Store for MLOps** — a tool designed to log, organize, compare, and monitor everything that happens in an ML pipeline.

It is widely used by Data Scientists, ML Engineers, and MLOps teams for:

- Experiment tracking
- Model metadata logging
- Dataset versioning
- Monitoring training + evaluation
- Tracking continuous training (CT)
- Reproducibility and governance

https://neptune.ai/

```python
import neptune

run = neptune.init_run(project="user/project", api_token="TOKEN")

run["params"] = params_dict
run["metrics/accuracy"] = 0.92
run["artifacts/model"].upload("models/model_v3.pkl")

run.stop()
```

Project Narrative

This project implements a complete, end-to-end MLOps pipeline designed to simulate how machine learning systems are developed, trained, monitored, and automatically retrained in real production environments.
It begins with a centralized YAML configuration file, which defines model parameters, dataset settings, Neptune credentials, and the rules governing continuous training. This configuration serves as the single source of truth for the entire system, ensuring reproducibility and decoupling logic from code.

The training workflow is implemented in `train.py`, which loads the YAML configuration, retrieves the dataset through `data_loader.py`, preprocesses it using `preprocess.py`, and trains a machine learning model—such as a Random Forest classifier—according to the specified parameters. After training, the model is evaluated with metrics computed in `evaluate.py`, and the system automatically assigns a version number to the model using utilities provided in `utils.py`.
The resulting artifacts, including `model_vN.pkl` and the associated scaler, are saved in the `models/` directory. During this process, Neptune.ai is used to log all relevant metadata such as hyperparameters, metrics, artifacts, and the entire configuration file. This enables complete traceability and comparison across different training runs.

In parallel, the pipeline includes a Continuous Training (CT) component implemented in `ct_trigger.py`. This script periodically loads the latest trained model and monitors new incoming data—simulated in this project—to detect potential data drift by comparing statistical properties of the new data to those of the original training distribution. Drift values and their corresponding plots are logged into Neptune, creating a visual timeline of model degradation or behavioral changes.

When drift exceeds a threshold defined in the YAML configuration, the CT component automatically triggers a new training run, which produces a new model version. The cycle repeats continuously, enabling the pipeline to adapt to evolving data patterns without manual intervention.

Project Team:

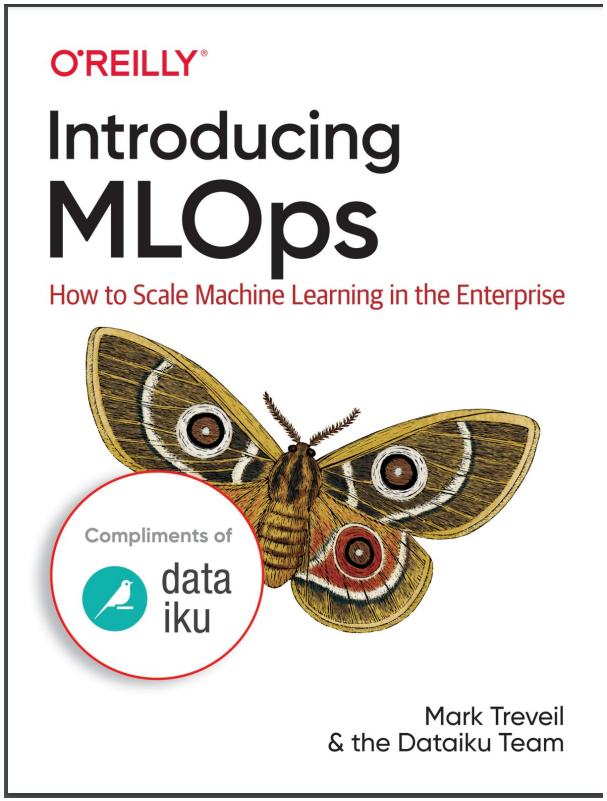
1. Project Owner
2. Project Manager (+Scrum Master)
3. Data Scientist
4. Software Developer
5. Client (Prof. Sylvio)

Expected Deliverables:

- Sprints about 45 min
- Meetings about 5 min

Sprint	Milestone	Key Deliverables
1	Initialization	Baseline training + version 1 model + Neptune run + Git Repo
2	Modeling	First drift detection + Data Change Simulation
3	Drift & CT	Auto-retraining + drift plots
4	Versioning + Validation	Multiple model versions + parameter changes + End-to-end test + release candidate + final summary

Textbook


O'REILLY
Introducing
MLOps
How to Scale Machine Learning in the Enterprise
Compliments of
data
iku
Mark Treveil
& the Dataiku Team