Machine Learning Systems and MLOps

Guide to AI and Data Courses · Part VI: Systems and Applications

Lifecycle | Versioning | Deployment | Monitoring and drift | Automation | Neighbouring courses

Sercan KÜLCÜ · Giresun University · Computer Engineering

© Sercan Külcü. All rights reserved. · Interactive HTML version of these slides: https://sercankulcu.github.io/files/ai_courses/slides/Chapter_35_MLOps.html

INTRODUCTION

What Question Does It Ask?

> **How is a model taken from the lab into production and kept running there?**

MLOps applies software engineering and DevOps practice to ML so that models reach production and run reliably there. The course covers versioning data, code and models together, automating the **pipeline**, deploying, and monitoring the model in production.

At a glance

APPROACH software engineering meets ML	**INPUT** trained model, data pipeline	**OUTPUT** a running, monitored service
SUCCESS MEASURE latency, uptime, data drift	**MATH LOAD** low	**DATA NEEDS** streaming production data

A Short History

2009
The first DevOpsDays is held in Ghent; DevOps, uniting development and operations, spreads.

2014
Docker 1.0 is released and Kubernetes announced: containers become the common unit of deployment.

2015
Sculley et al. expose hidden technical debt in ML systems: model code is a small part of the system.

2017
Google presents TFX and Uber presents Michelangelo; in-house ML platforms come into view.

2018
Databricks open-sources MLflow: experiment tracking and a model registry.

2021
Andrew Ng calls for data-centric AI: focus on data quality more than on the model.

THE COURSE

Core Concepts

Pipeline
Automated steps from data preparation to deployment.

Versioning
Tracking and rolling back data, code and model together.

Experiment tracking
Logging the parameters, metrics and outputs of every run.

Model registry
Approved model versions kept in one place.

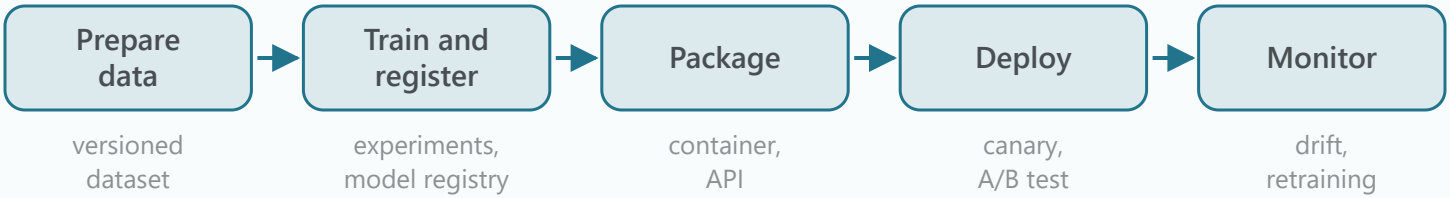
Feature store
The same features used in training and in production.

CI/CD/CT
Continuous integration, delivery and continuous training.

Model serving
Running as an online API or as a batch prediction job.

Data drift
Production data moving away from training data; must be monitored.

A Typical Workflow



Example: A bank's credit model is retrained every month on new applications. The best registered version is packaged in a container, first exposed to a small share of traffic, and rolled out to everyone if nothing breaks. When the applicant distribution drifts, the monitoring dashboard alerts the team and the cycle restarts.

THE COURSE

What Does the Course Cover?

1. ML systems in production: requirements, technical debt
2. Data pipelines, data validation and versioning
3. Feature engineering and feature stores
4. Experiment tracking and reproducible training
5. Model evaluation: slice analysis and model tests
6. Packaging: containers and the model registry
7. Model serving: online APIs and batch prediction
8. Faster inference: latency, scale and cost
9. Deployment strategies: canary, shadow, A/B tests
10. Monitoring: data drift, concept drift, alerts
11. Continuous training and CI/CD pipelines
12. Responsible ML: auditing, documentation, model cards

Skills you will gain

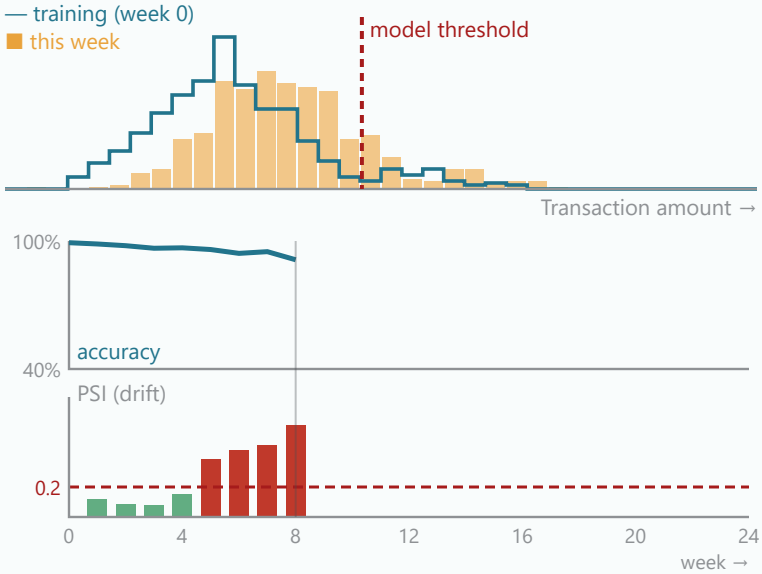
versioning with Git and DVC | experiment tracking with MLflow | packaging models with Docker | serving a model as a REST API | building pipelines with Airflow | monitoring dashboards in Prometheus

A typical semester; the order and coverage vary between universities.

TRY IT

The Model Silently Degrades in Production

A bank's fraud model flags unusually large transactions, while prices rise a little every week. Advance the weeks, watch drift (PSI) and accuracy, and retrain the model when the alarm goes off.



Week in production: **8**

Retrain | Reset

Week 8: PSI = **0.61** → **ALARM**
Accuracy **91.2%** (99.2% in its training week)
Model trained on week 0 data.

PSI compares this week's distribution with the training data over ten bins. Above **0.2** counts as an alarm. Labels arrive late; PSI is computed at once.

A model is not built once and left alone like ordinary software: MLOps sets up **monitoring**, alarms and an automated retraining pipeline. Drift is caught before accuracy drops.

CONTEXT

Prerequisites

Courses worth knowing before this one, and why they are needed.

Course	Why it is needed
Machine Learning	deploying, monitoring and retraining models
Operating Systems	processes, memory and resource management: the concepts containers rely on
Object Oriented Programming with Java	the habit of writing modular, testable code
Software Engineering	version control (Git), testing and continuous integration

CONTEXT

Are You Ready for This Course?

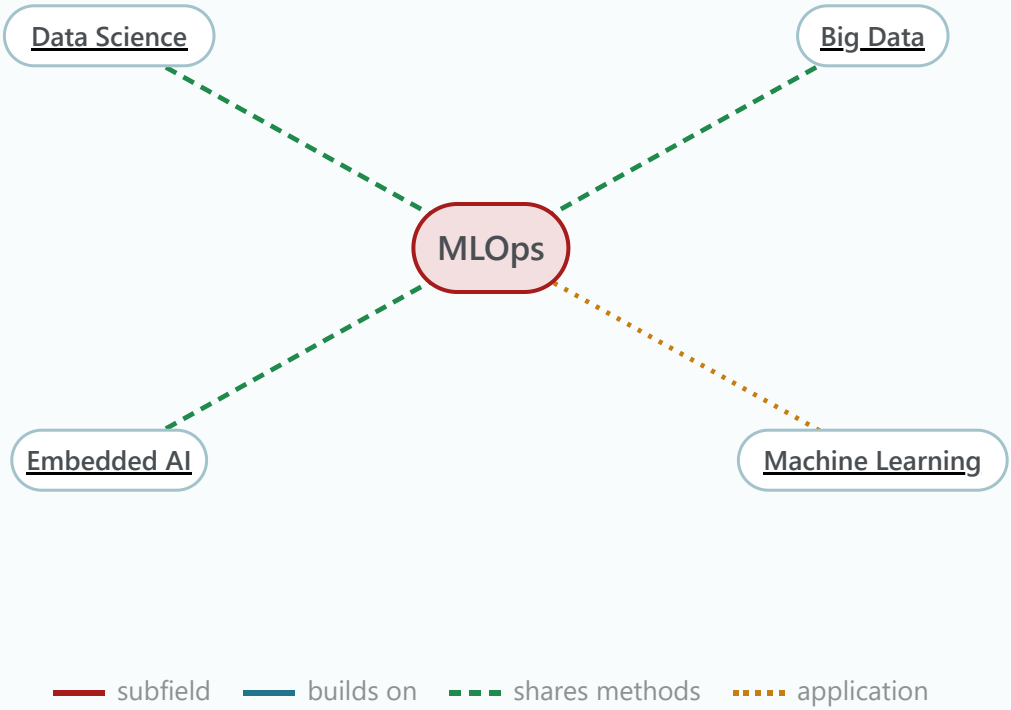
Ask yourself; for every "no", the right-hand column shows where to start.

Ask yourself	If not, start with
Can you train a classifier and evaluate it with cross-validation?	Machine Learning
Can you create and merge branches in Git and resolve a merge conflict?	A short Git tutorial and a software engineering course
Can you manage processes, environment variables and permissions from the shell?	An operating systems course and an intro to the Linux shell
Can you write a modular Python project that has unit tests?	Object-oriented programming, then a short pytest tutorial

If you can say "yes" to most of these, you are ready; one or two gaps can be closed during the semester.

CONTEXT

Neighbouring Courses



OVERLAP

Overlapping Courses

Relation	Course	Shared ground
application	This course applies the methods of Machine Learning	deploying, monitoring and retraining models
shares methods	Shares methods with Data Science	experiment tracking, data versioning
shares methods	Shares methods with Big Data Systems	data pipelines
shares methods	Shares methods with Embedded AI	deploying and updating models on devices

DIFFERENCES

Where It Differs (1/2)

Criterion	MLOps	Machine Learning	Big Data
Core question	How is a model taken from the lab into production and kept running there?	How does a program generalize from examples?	How is data that does not fit on one machine stored and processed?
Approach	software engineering meets ML	statistical learning	distributed storage and computation
Input	trained model, data pipeline	labeled or unlabeled examples	huge, fast-flowing data
Output	a running, monitored service	a model that makes predictions	processed data, query results
Success measure	latency, uptime, data drift	error on unseen data	scalability, latency, cost
Math load	low	high: linear algebra, probability, optimization	low
Data needs	streaming production data	moderate to large	very large

- **Machine Learning:** ML teaches how to build a good model; this course teaches how to keep it running reliably in production for months.
- **Big Data:** Big data systems store and process data at scale; this course manages the lifecycle of models trained on that data.

Where It Differs (2/2)

Criterion	MLOps	Data Science	Embedded AI
Core question	How is a model taken from the lab into production and kept running there?	How do we ask the right questions of data and turn the answers into decisions?	How do AI models run on small, low-power devices?
Approach	software engineering meets ML	question, data, analysis, story	resource-constrained inference
Input	trained model, data pipeline	raw, messy data	trained model, device constraints
Output	a running, monitored service	report, visualization, model	small, fast model
Success measure	latency, uptime, data drift	impact on decisions, accuracy	latency, energy, memory
Math load	low	medium: statistics	medium
Data needs	streaming production data	moderate to large	training data already exists

- **Data Science:** Data science runs an analysis that answers a question; this course builds a service that runs, is monitored and is renewed.
- **Embedded AI:** Embedded AI fits the model onto a small device; this course deploys, versions and monitors it on servers.

PRACTICE

Applications

Recommendation services
Models answering millions of requests a day.

Fraud detection
Decisions in milliseconds, frequent retraining.

Demand forecasting
Batch prediction jobs that run every night.

Search ranking
Models tested continuously with A/B tests.

Credit risk
Auditable, versioned and monitored models.

LLM services
Serving large models at scale and at high cost.

PRACTICE

Risks and Ethics

Silent decay
Without drift monitoring a model degrades with no errors; many demand forecasts lost their accuracy overnight in COVID-19.

Training-serving skew
If a feature is computed differently in training and production, a model that tested well predicts wrongly live.

Feedback loops
A model retrained on the outcomes of its own decisions, such as a recommender or credit model, amplifies its own bias.

Untraceable versions
A model whose training data and code are unknown cannot be explained in a complaint or audit, nor rolled back.

In depth: Explainable AI and AI Ethics.

PRACTICE

Tools and Datasets

Tools and libraries

MLflow | DVC | Docker | Kubernetes | Kubeflow | Apache Airflow | Prometheus

Classic example scenarios

Credit model retrained monthly | Nightly demand forecast | Real-time fraud serving | Recommender with A/B testing | Simulating data drift | Rolling back a bad release

The course teaches a **process**, not an algorithm: a small model is turned into an end-to-end pipeline, from training to monitoring.

PRACTICE

Common Misconceptions

Common belief	In fact
Once the model is trained, the job is done.	Data changes in production; an unmonitored model silently degrades over time.
MLOps is just installing tools.	Tools support the process; the real work is versioning, testing, ownership and rollback.
High test accuracy means good production performance.	Production data may differ from training data; latency and cost are part of success too.
Most of an ML system is the model.	Model code is a small part next to data, configuration, serving and monitoring infrastructure.

NEXT STEPS

After This Course

Courses that build on or extend this one

This course is mostly an end point: it builds on the others.

Related careers

MLOps engineer | Machine learning engineer | Data engineer | Platform engineer | Site reliability engineer (SRE)

MLOps is the backbone of the systems part of the guide: big data systems handle the training data, GPU programming the training speed, and embedded AI deployment at the edge. After this course, cloud platforms, distributed training or serving large language models are natural next steps.

NEXT STEPS

Further Reading

- Huyen, *Designing Machine Learning Systems*, O'Reilly, 2022.
- Burkov, *Machine Learning Engineering*, True Positive Inc., 2020.
- Lakshmanan, Robinson and Munn, *Machine Learning Design Patterns*, O'Reilly, 2020.
- Sculley et al., "Hidden Technical Debt in Machine Learning Systems", *NeurIPS*, 2015.

The End

Chapter 35: Machine Learning Systems and MLOps

← Previous chapter · Map of the field · Next chapter →