MLOps

Shabab Akhter
External Lecturer
RUC

Contents

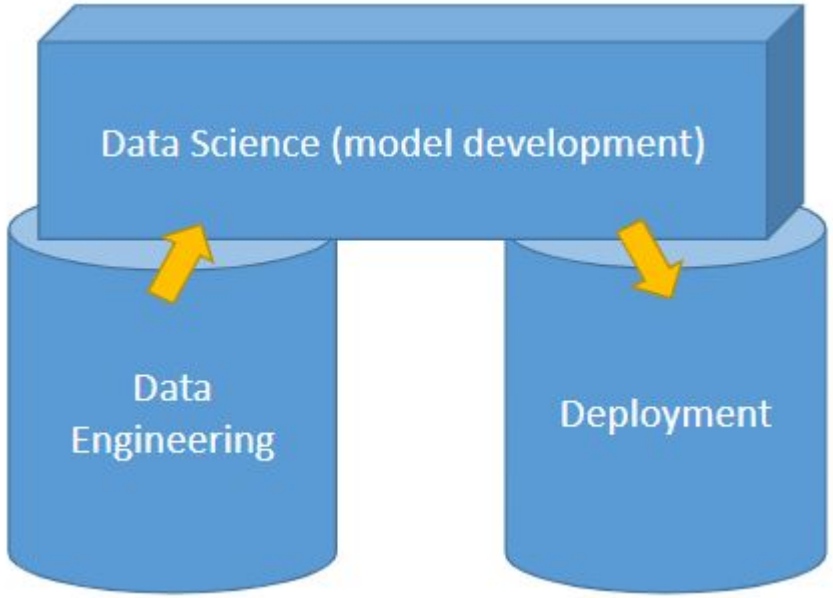
Contents

What is MLOps?

- MLOps is Machine Learning Operations
- It about handling the deployment of ML models, but also the development, especially beyond the first prototype
- MLOps = ML + DevOps (CI/CD)
- Machine Learning is not just code or software it is also DATA!
- We need to version not only code, but also models and data
- MLOps is also about how to manage model retraining
- MLOps is also about monitoring Machine Learning models
- MLOps is also about making interdisciplinary teams of both developers and operation people

Background and recap

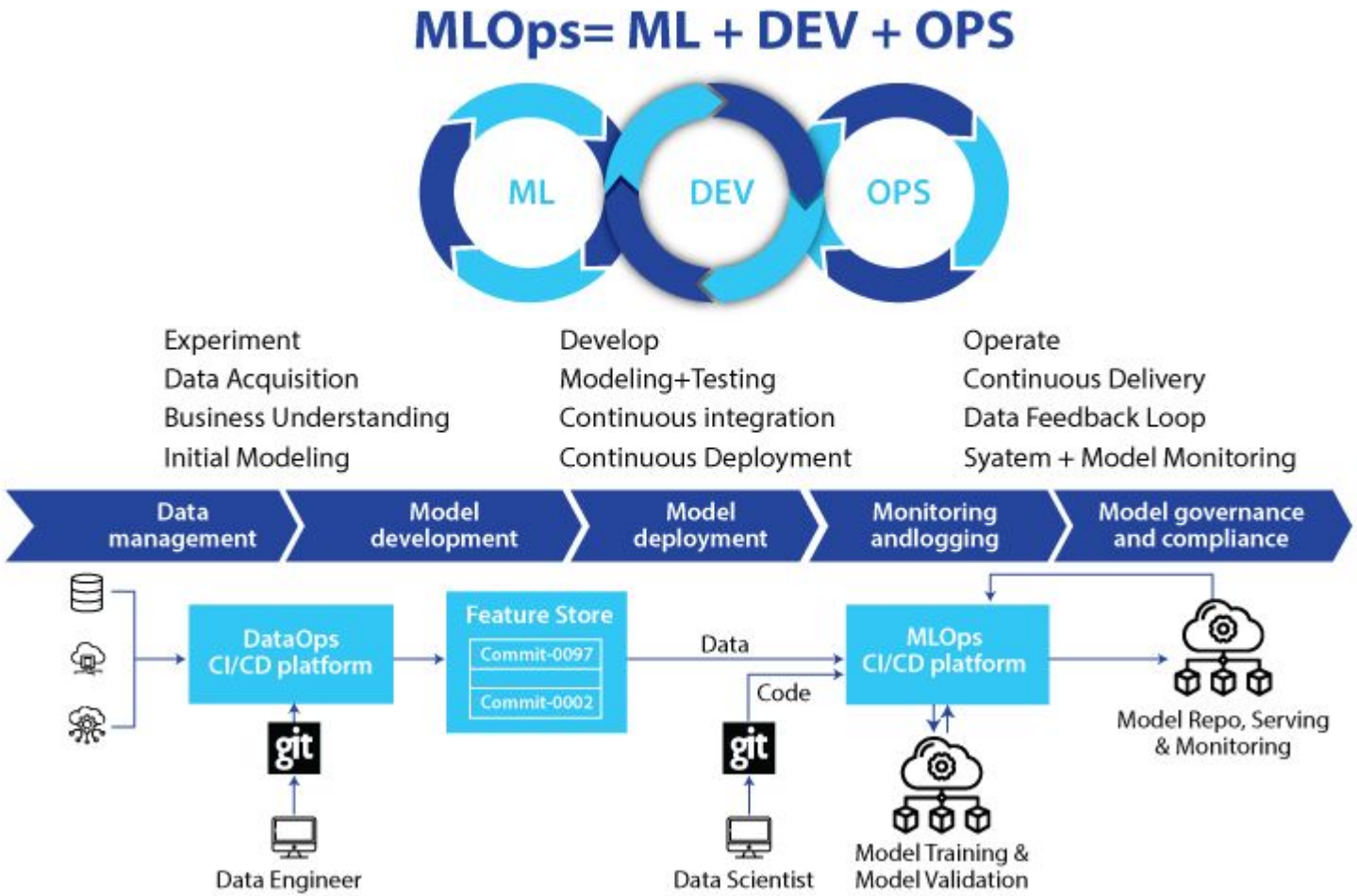
- Data engineering, Data Science (model development), and Deployment
  - We have seen how to manage various types of data infrastructure
  - We have seen how to build ML models
  - But how do we take a ML model to production (deploying it)?
    - How do we utilize predictions, recommendations, clustering, … etc. in live applications
    - We need to both ***train a model*** as well as use it to ***serve predictions*** etc.
    - The world might change, the model might drop in performance, new data might arrive, …
- Hence, it often make sense to retrain the model (in production)



What is MLOps?

- Deployment
  - A different (runtime) environment than the development environment
  - Are we using the same version of packages?
  - Using containers to package environments easily – Containers are easy to maintain, portable and fast to start up
- Architecture
  - Microservice architecture is ideal for MLOps
  - Deploy machine learning models as microservice with an API – different from training and predictions
- CI/CD deployment strategies
  - Continuous integration (CI) – code changes are continuously and frequently integrated and changes are automatically tested and merged
  - Continuous deployment (CD) – automating the release of the code validated during CI process. Ensure you always have production ready code.
- Automation and scaling
  - Automate as much as possible.
  - Feature store and experiment tracking – during the development phase
  - Containerization, CI/CD and microservice architecture – during the deployment phase
- MLOps maturity: Level of automation, collaboration, and monitoring within MLOps processes

What is MLOps?



Contents

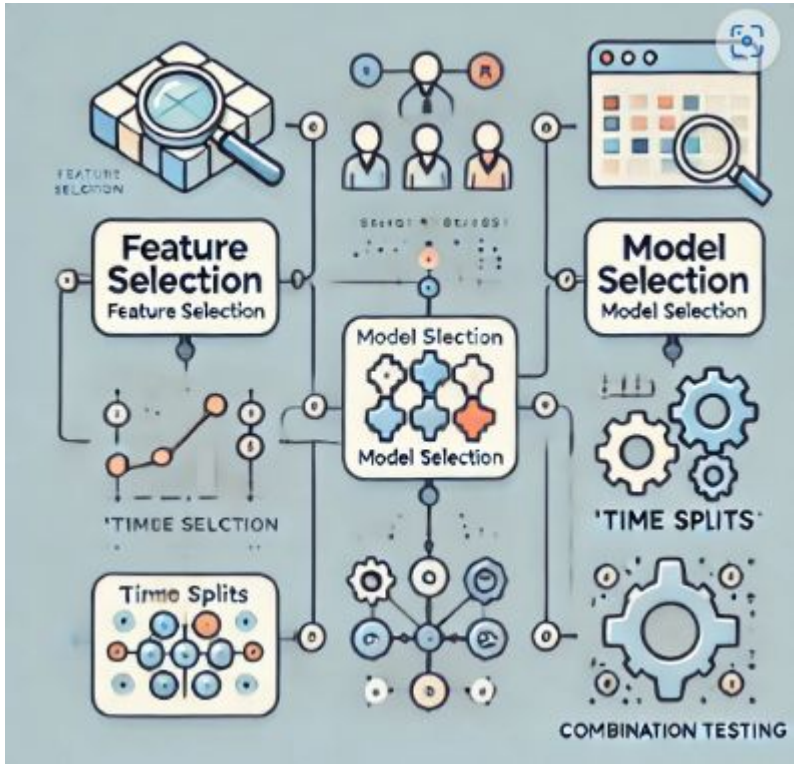
Why do we need to retrain models?

- Model retraining
  - Experimentation with different parameters to find the best model
  - Static status quo. Keeping a ML model static. Problem, “the world” (input data changes):
  - Format of data changes
  - The same thing are measured on a new way
  - People start behaving differently
- Alternative retraining of model. The same things might be problems, but we can not detect them and potentially deal with them. However, additional concerns can happen:
  - Changes in the past
  - Pre-processing needs to change
  - Inconsistency of old data

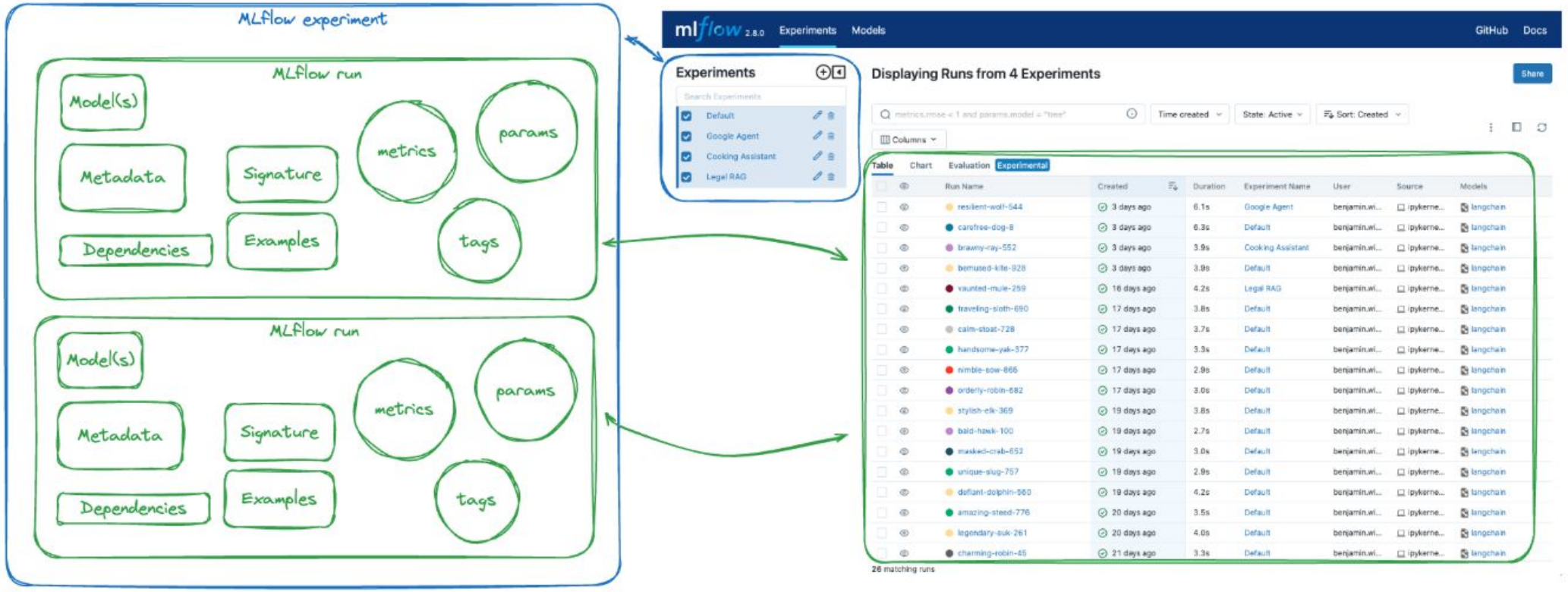
Modelling involves a lot of experimentation and you can easily lose track

A typical data science workflow involves a lot of experimentation:

1. Train using different set of features
2. Train using different time cuts
3. Train using different models
4. Train using different parameters
5. All the above can be done interchangeably as well
6. The order / magnitude when it comes to experimentation is too large to track manually



MLFlow significantly simplifies the entire experimentation process



Exercise

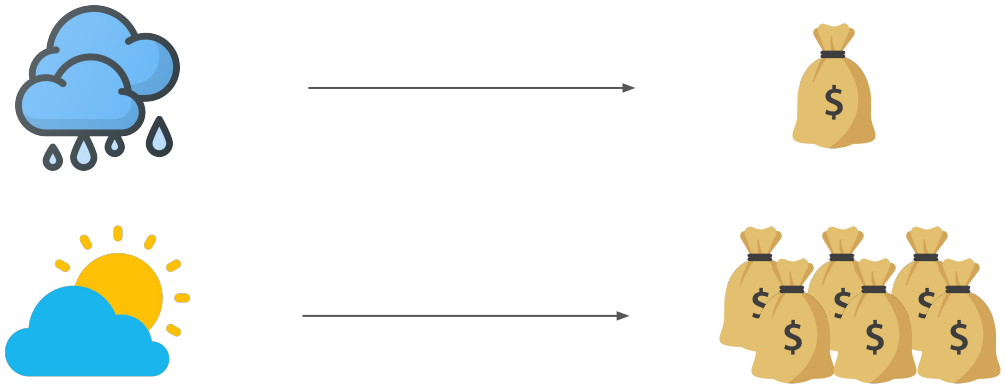
1. Let us see an example of how MLFlow is used to track experiments
2. Retrain models and track the experiment using MLFlow

Contents

Let's use a real life example to understand the basics of drift

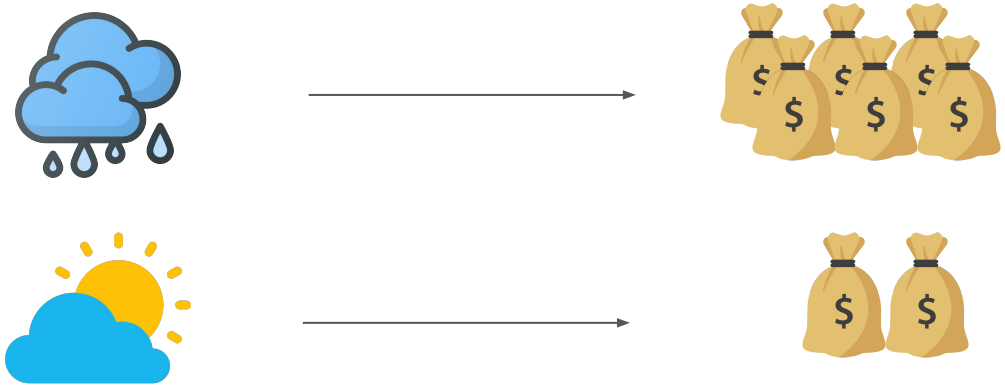
You have a food court and you are building a model to predict sales of food in this food court.

After some analysis you find that weather is strongly correlated. You have modelled following relationship:



1 year later, the model does not work, and relationship seems to have changed

This is the new relationship. Any idea why it could be like this?



The management decided to add roofing to the food court

ML models lose performance over time due to drifting patterns in the real world

Drift occurs when the statistical properties of data change over time, potentially making models less accurate.

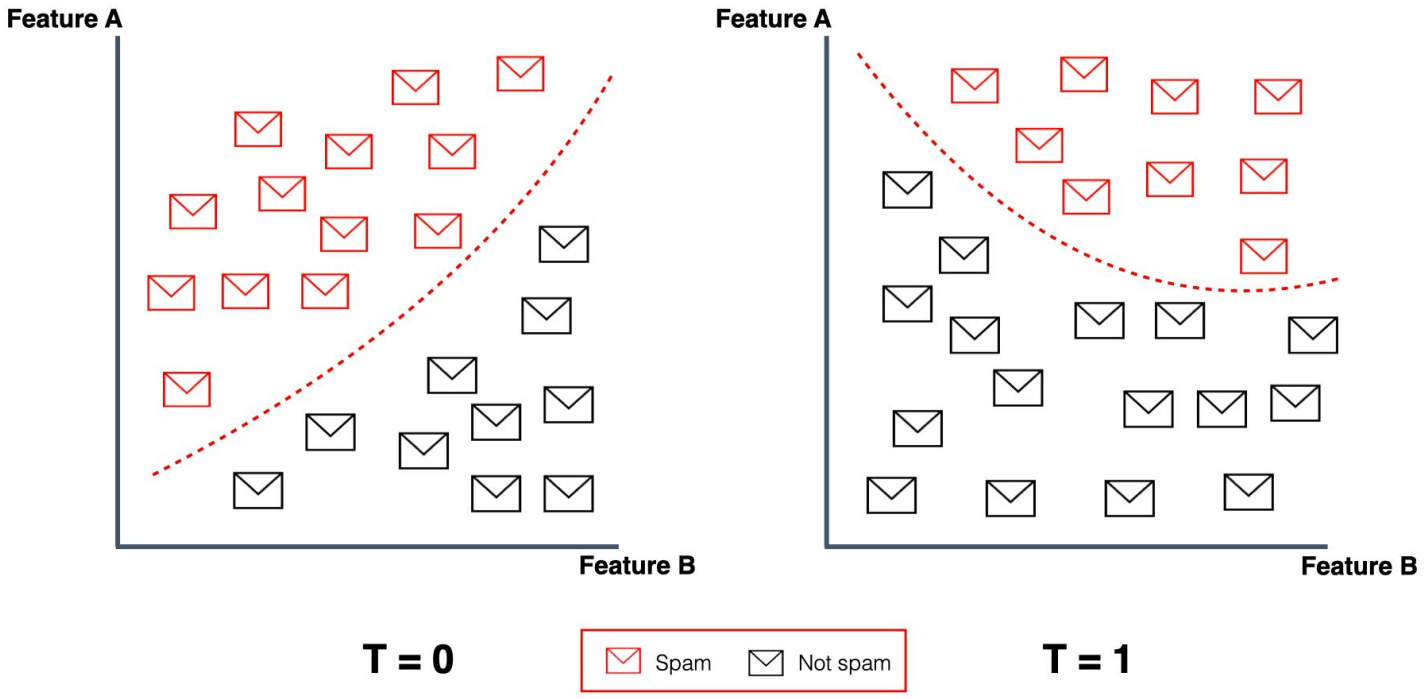
Types of Drift:

- **Concept Drift:** The underlying patterns or relationships in the data change (e.g., sensor degradation over time).
- **Feature Drift:** The distribution of individual features changes, affecting the model's predictions.
- **Label Drift:** The distribution of the labels (or outcomes) changes.

What causes it?

- Environmental Changes: External factors like temperature, humidity, or sensor aging.
- Model Degradation: Over time, IoT systems may become less accurate as new patterns emerge.
- Sensor Failure or Drift: Sensors can degrade or be influenced by noise, leading to skewed data.

Concept drift



Concept drift refers to a change in data patterns and relationships over time.

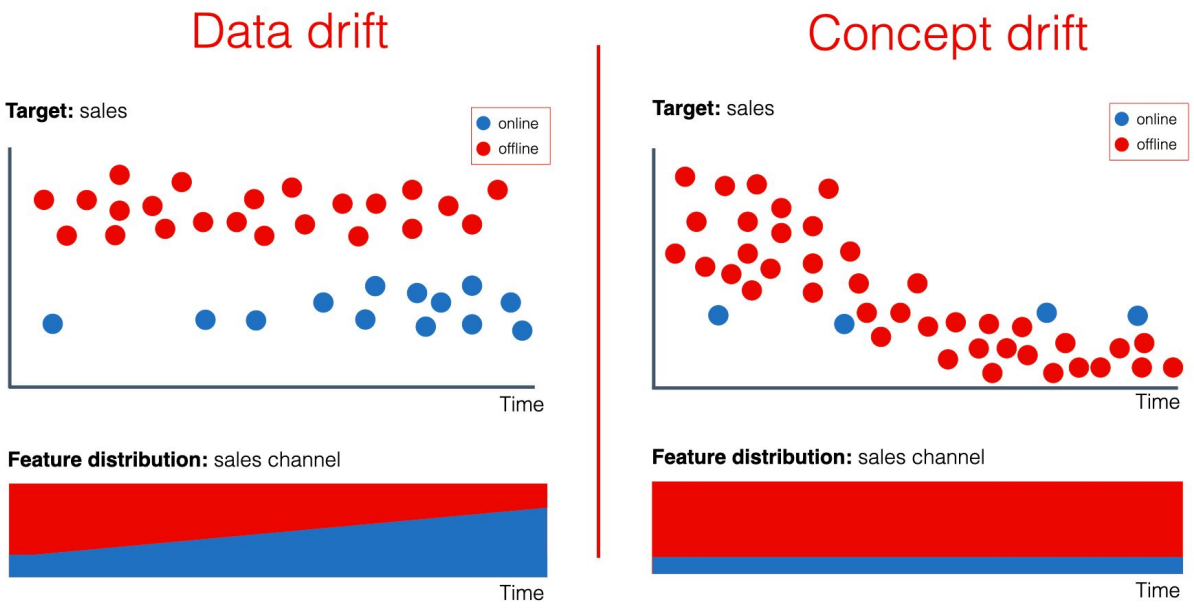
Imagine you have an ML model predicting whether emails are spam.

You train it using a massive dataset of emails, and it becomes pretty good at it. But as time passes, how people and services compose and send email changes. On top of it, spammers adapt and become better at mimicking legitimate emails.

Now, what used to be obvious spam suddenly looks like regular email, and vice versa. Your once-accurate model starts making mistakes.

This change is an example of concept drift.

Data drift vs Concept drift



With data drift, the relationship between sales channels and volumes remains stable, but the distribution of channels shifts. With concept drift, you can observe a decrease in average sales in an offline channel

The difference: Data drift refers to the shifts in input feature distributions, whereas concept drift refers to shifts in the relationships between model inputs and outputs.

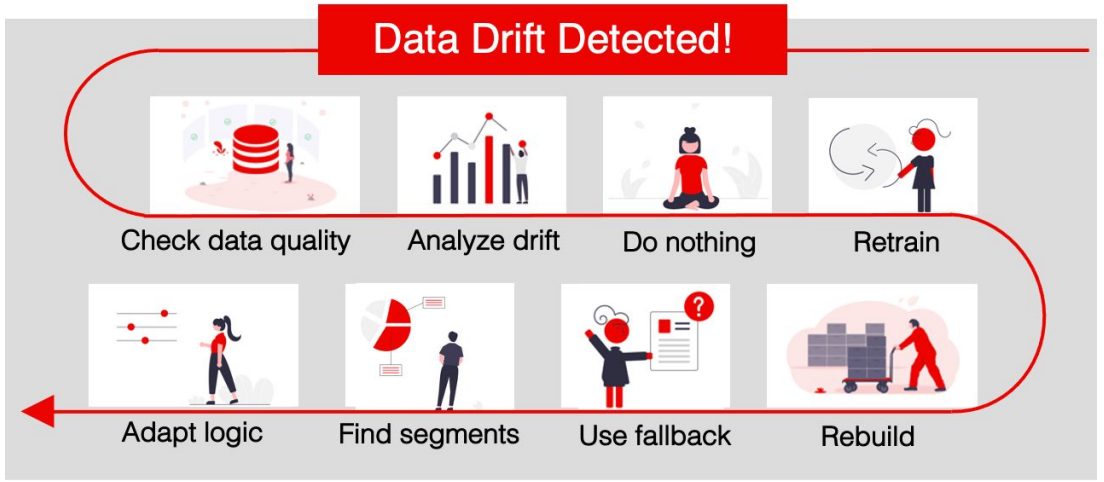
The similarity: Both data drift and concept drift can result in a decline in model quality and often coincide. In monitoring, data distribution drift can be a symptom of concept drift.

Techniques to address drift

Model Retraining: Periodically retrain models with new data to adapt to changes.

Drift Detection: Algorithms like the Kolmogorov-Smirnov Test or Population Stability Index (PSI) can monitor data distribution changes and trigger model retraining.

Online Learning: In this approach, models learn incrementally, allowing for real-time updates based on new data.



Exercise

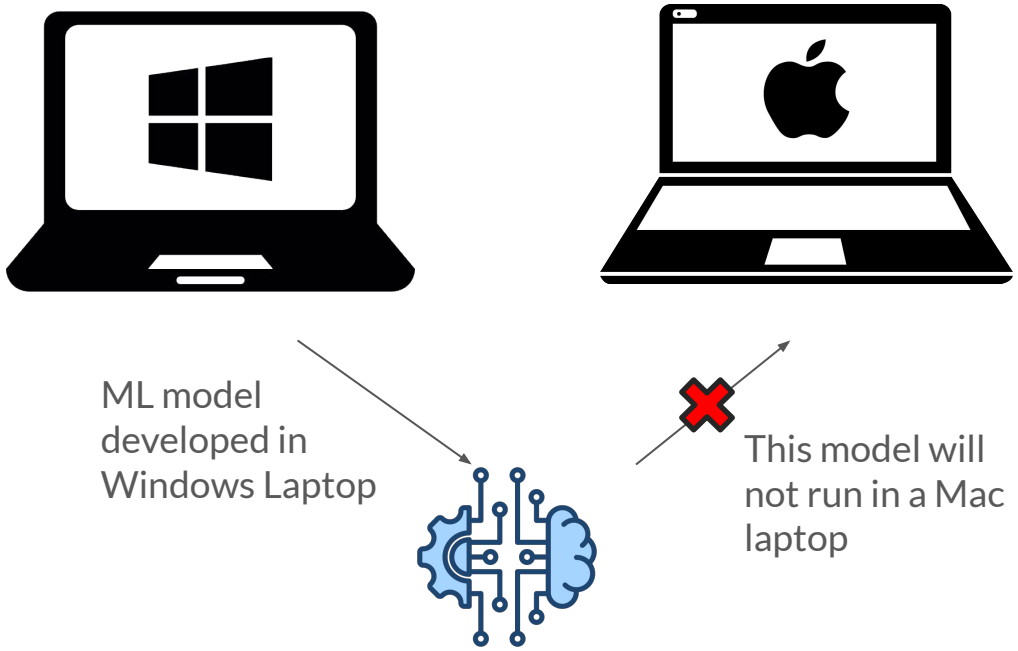
Train models and detect drift

Contents

Why MLOps is messy without the right tools

- Imagine you've built a cool machine learning model on your laptop, and it works perfectly.
- But when you share it with others or deploy it to a server, it *breaks*!
- Why? 🤔 Because the software, settings, and environment are all different.

- **Key problem:** Lack of consistency and portability across different systems leads to failures.

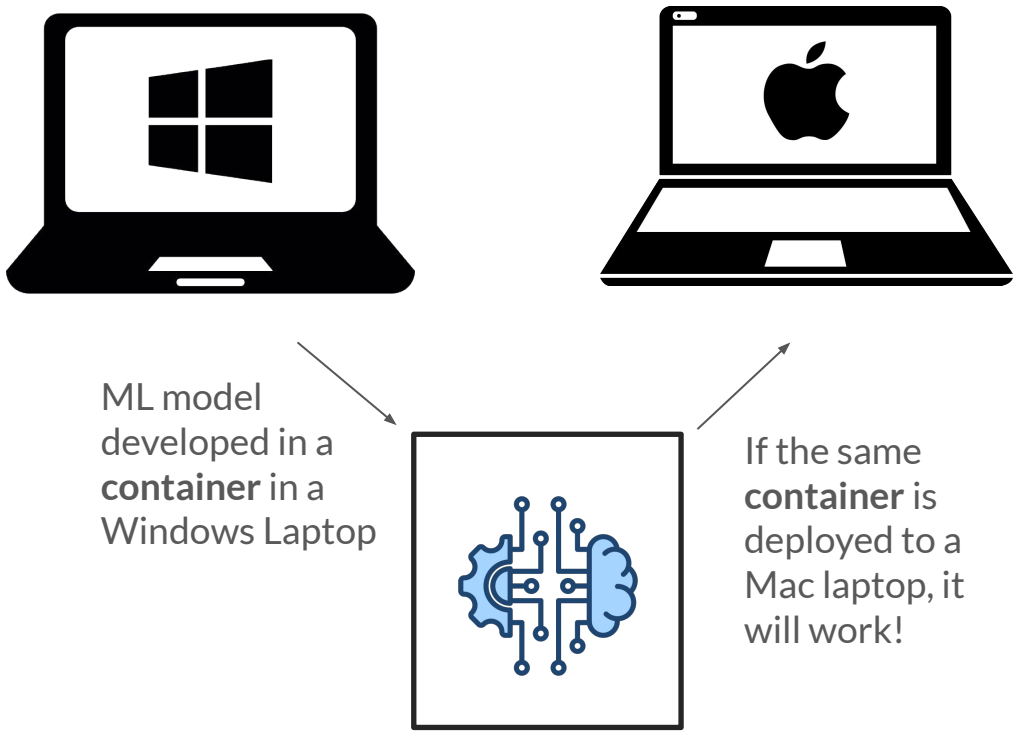


This is where containerization comes in

- Think of a container as a magic box 🧳 that holds your code, model, and all its settings (like Python version, libraries, etc.).
- Once your model is in a container, you can run it anywhere—on your laptop, a server, or the cloud—and it works the same way.

- **Containerization = Consistency + Portability**

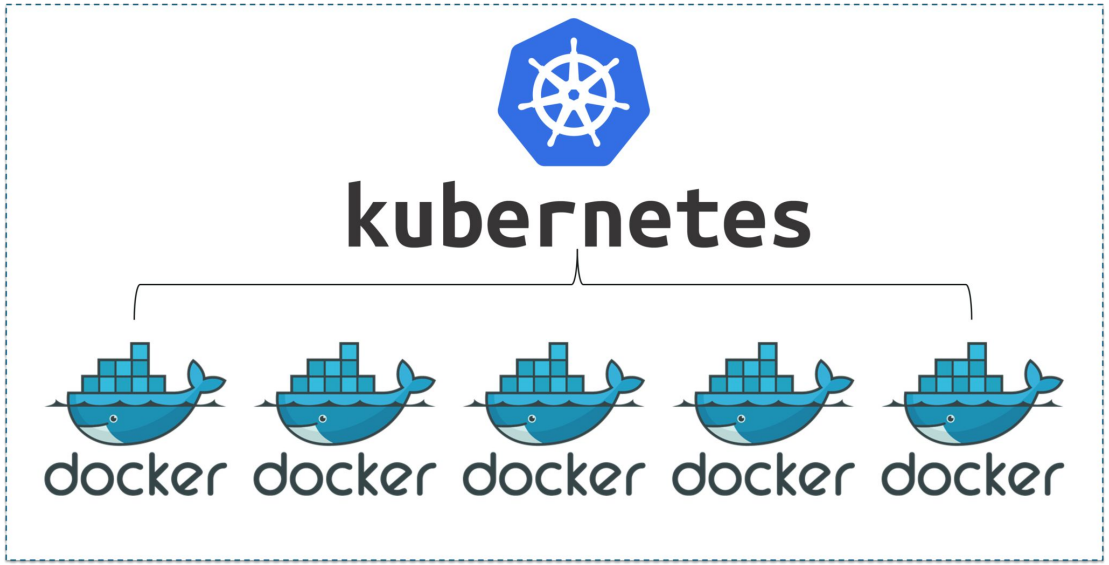
- It's like packing your stuff in a suitcase that works in any country. 🌍



How Docker & Kubernetes help

- **Docker** 🐳 – This is the tool that creates containers.
- It packages your model and its environment into one neat bundle.
- Now your model can run anywhere.

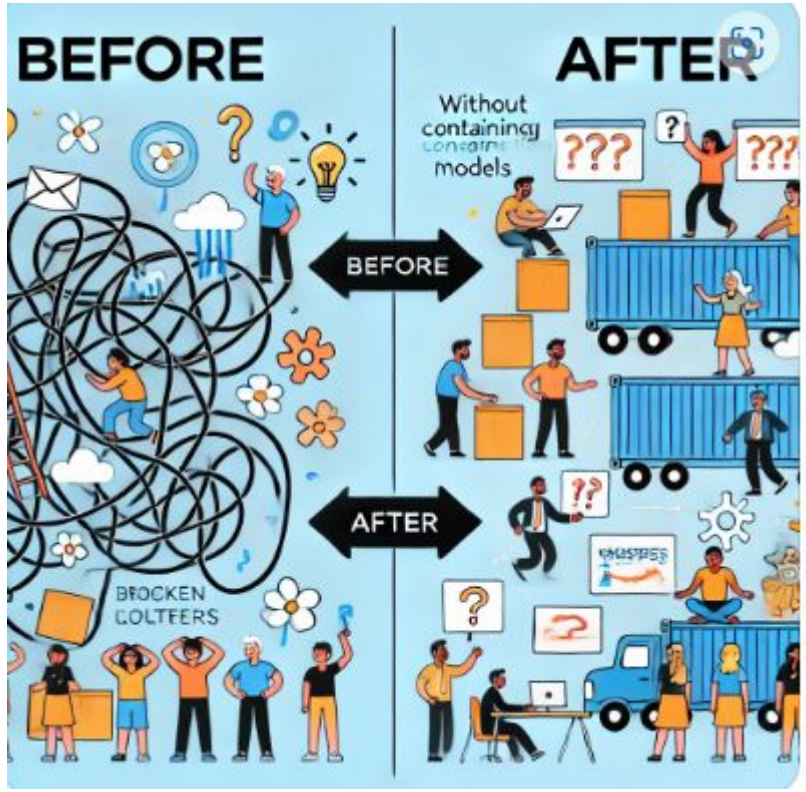
- **Kubernetes** ☸️ – This tool manages lots of containers at once.
- It helps deploy, scale, and monitor your containers across multiple machines or in the cloud.
- Think of it as the orchestra conductor for all your containers. 🎵



Containerization brings order to chaos

- The data science workflow is a chaotic experimentation process
- This is deliberate and it makes sense.

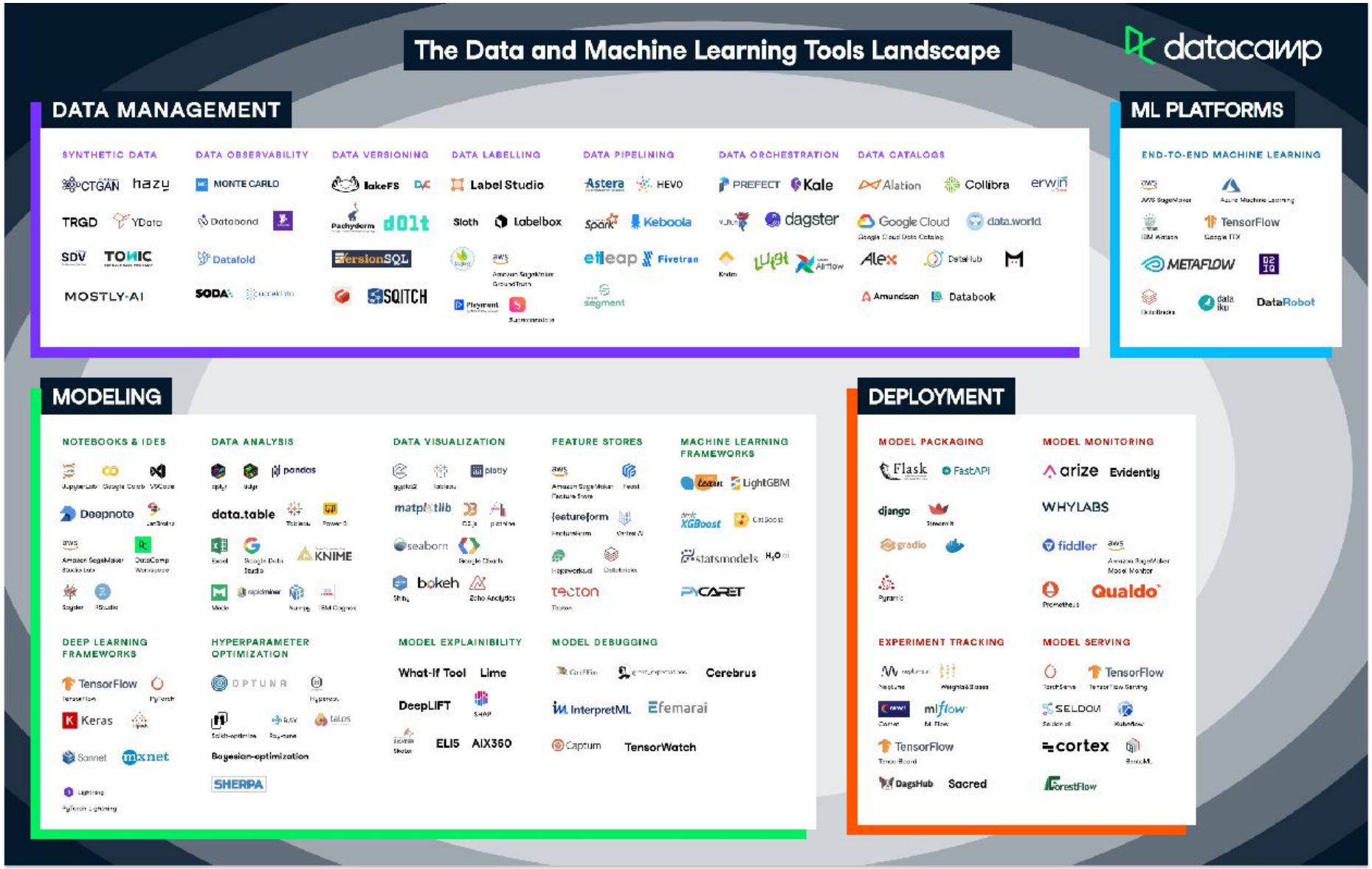
- But at some point, this chaos needs to have order and that is where Containerization comes in
- It is the heart of MLOps



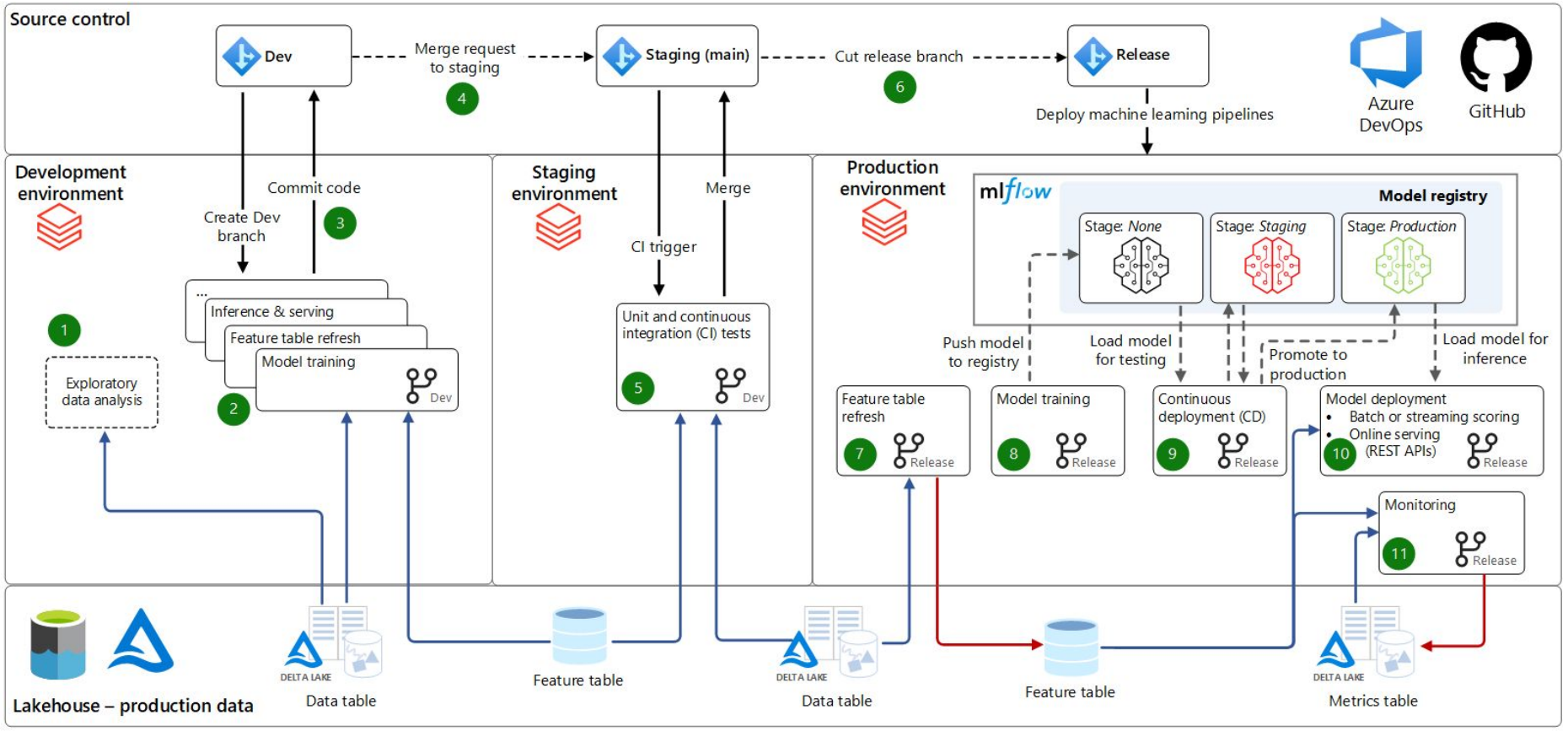
MLOps tools

MLOps tools overview by DataCamp:

https://www.datacamp.com/blog/infographic-data-and-machine-learning-tools-landscape



Example architecture in MS Azure



Microsoft Azure

Wrap up

1. Background & fundamentals of MLOps
    a. What is MLOps?
    b. Key concepts such as monitoring, model ops, data ops, drift, version control, etc.
    c. Tools used – MLFlow, Docker, Kubernetes, etc.
2. Retraining & version control
    a. How to control retraining & experimentation
    b. Using MLFlow in practice
    c. Exercise – MLFlow
3. Monitoring & drift detection
    a. How to monitor ML models
    b. What is data drift and how to detect it
    c. Exercise – data drift detection
4. Containerization
    a. What are containers?
    b. Deep dive into Docker & Kubernetes
    c. MLOps holistic architecture view