MSDS 626 - Case Studies?

Robert Clements

MSDS Program

University of San Francisco

Case Studies…in MLOps

- In this course you will learn about MLOps
  - Sometimes called ModelOps.

- Most important aspects of the end-to-end ML pipeline will be covered
  - Introduce open source tools to get some hands-on practice.
  - Note: some steps require more *theory* than others (e.g. concept drift).
  - Note: some aspects & tools covered in previous courses (e.g. version control with git; docker; orchestration with airflow).
  - Note: different companies use different stacks – we can’t learn every tool available!

In This Course…

We will be talking about:

- MLOps pipeline
- Experiment tracking
- Data versioning and quality
- Registering artifacts
- Model serving
- Model monitoring
- Model explainability or Foundation Models (time-dependent)

We won't be talking about:

- Business problems
- How to train ML models
- Data storage/warehouses
- Data pipelines in detail
- Model re-training
- Ethics/Responsible AI
- Compliance
- Experimentation (in the A/B testing sense)
- ALL of the tools and steps of MLOps

How We Will Learn

- Lecture and Discussion
  - Bring your Practicum/previous job experiences to class
- In-class Demos and Labs
- Grade is based on:
  - Participation (**40%**)
  - Final Project (**60%**)
- Disclaimer: MLOps is a software engineering-heavy field
  - I am not a software engineer

Final Project & Participation

Project

- Group project – POCs for *most* stages of an MLOps pipeline
- Each lab will help you to complete part of the project
- There will be in-class time to complete the rest
- 5-minute presentations to be given during last class session or during finals week
- More details in Canvas

Participation

- Peer review

Course Resources

- Office Hours: T/Th 3:00-4:00 in Room 605, or by appt
- Canvas: for sharing lectures, demos and labs
- Piazza: for discussion, Q&A, and Project check-ins
- GCP Credits will be provided for labs

Day	Date	Topic	Assignment
T	3/21	Course Intro, Dev Env (demo); Experiment Tracking	Lab – Dev Environment; Group details deliverable due
Th	3/23	Experiment Tracking; Model Registry (demo)	Lab – Experiment and artifact tracking
T	3/28	Code and Data Versioning (demo)	Experiment and artifact tracking deliverable due
Th	3/30	Guest lecture: Nir Barazida - DagsHub	Lab – Data versioning
T	4/4	Data Quality (demo) and Feature Stores	Data versioning deliverable due
Th	4/6	Data Quality and Feature Stores	Lab – Data quality
T	4/11	Pipeline Orchestration & CI/CD (demo)	Lab – Containerization, orchestration, IaC
Th	4/13	Pipeline Orchestration & CI/CD	Lab – ML pipeline orchestration
T	4/18	Model Deployment Patterns (demo)	Lab– CI/CD; ML pipeline deliverable due
Th	4/20	Model Deployment Patterns	Lab – Model deployment
T	4/25	Model Deployment as a Webapp (demo)	Lab – Code quality; CI/CD and others deliverable due
Th	4/27	Model Monitoring	Lab – Webapp
T	5/2	Model Monitoring (demo)	Lab – Model monitoring; Webapp deliverable due
Th	5/4	Model Monitoring – A/B Testing and Retraining	Lab – free time
T	5/9	Foundation Models (demo)	Lab – free time; Model monitoring deliverable due
Th	5/11	No class session	Project presentations
Finals		Final MLOps Project Due	Project presentations

Path of Least Resistance

- Let’s not get bogged down by technology – plenty of time for spinning your wheels when you start working at a company.
- We will use simple, easy to set up, tools (for the most part).
- Meant to be a fun interactive class – if you want to explore other tools, or playing more with GCP, you are free to do so, but is not always required.

Resources for Learning

- Slides and demo notebooks
- Two books in our library related to MLOps:
  - Effective Data Science Infrastructure *by Ville Tuulos*
  - Designing Machine Learning Systems *by Chip Huyen*
- Other resources I will give you during each lecture

What is MLOps?

What is MLOps?

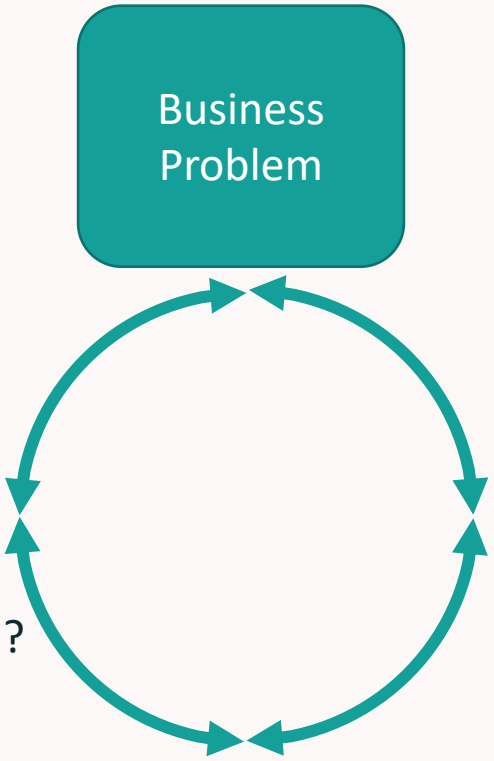
- Develop, deploy and maintain ML applications - https://madewithml.com/#mlops

- Set of best practices for putting machine learning models into production - https://github.com/DataTalksClub/mlops-zoomcamp

- DevOps (software apps) and MLOps (ML apps) are both about streamlining processes
  - MLOps is more complex than DevOps, though.

The MLOps Pipeline/Lifecycle

always begins with the question: do we even need to use ML?



What is our goal?

What does the customer really need?

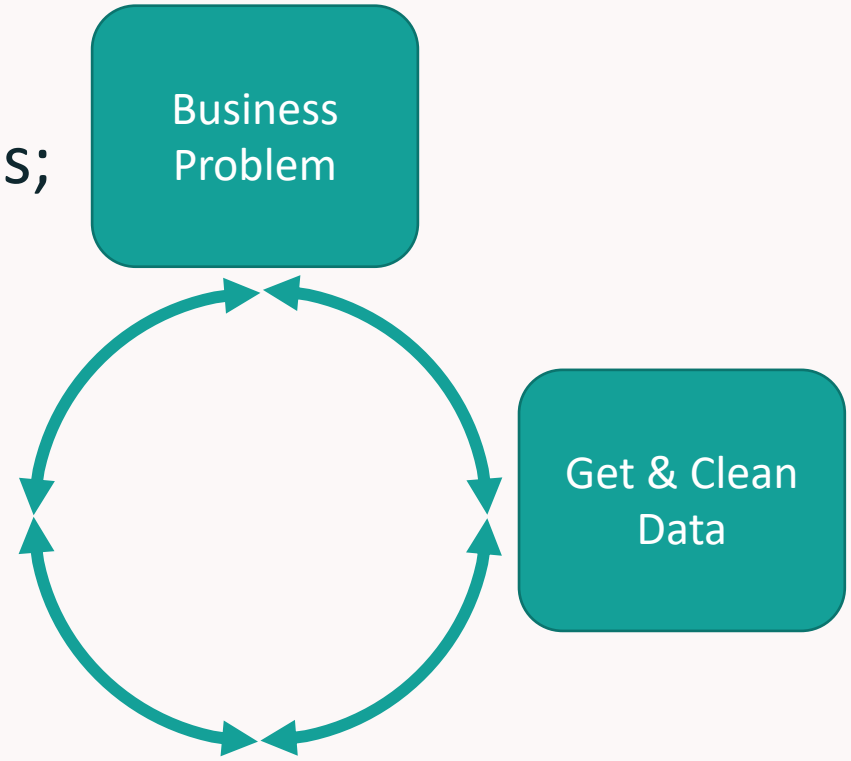
Is there at least some chance of success?

Is there a non-ML solution that will work?

Is there tangible value?

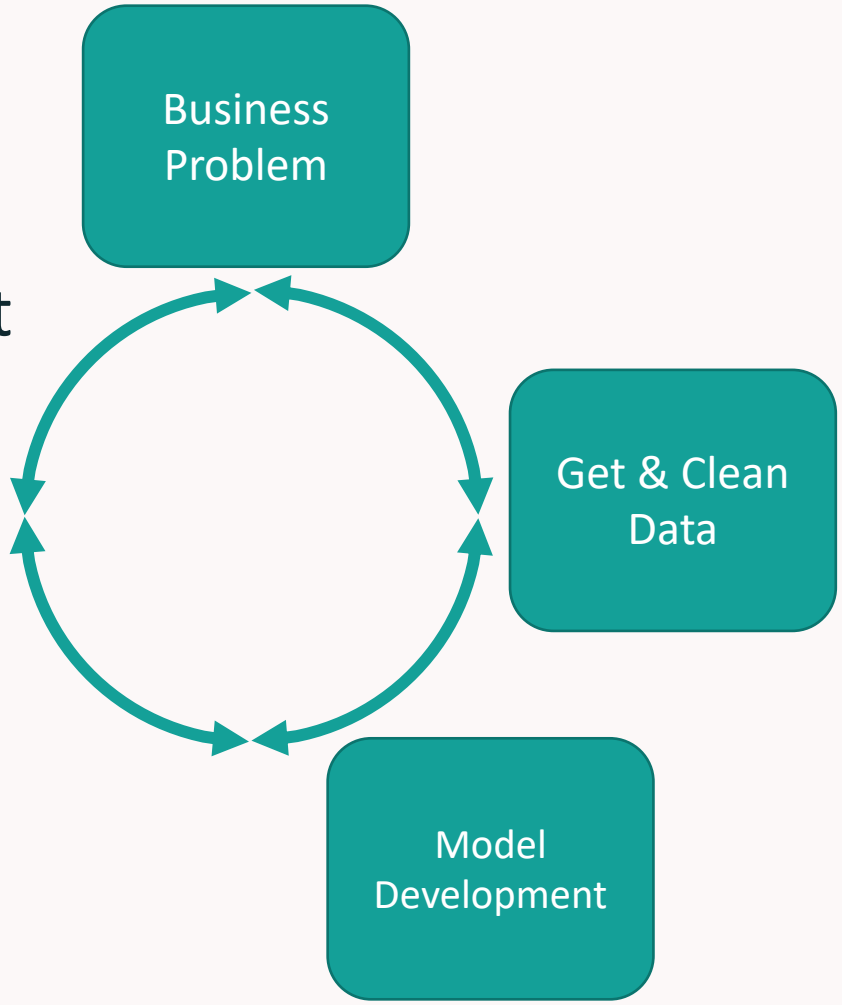
The MLOps Pipeline/Lifecycle

- Do we have data?
- Where is the data: flat files; database; datalake; lakehouse?
- Is it labeled?
- How clean/accurate is it?



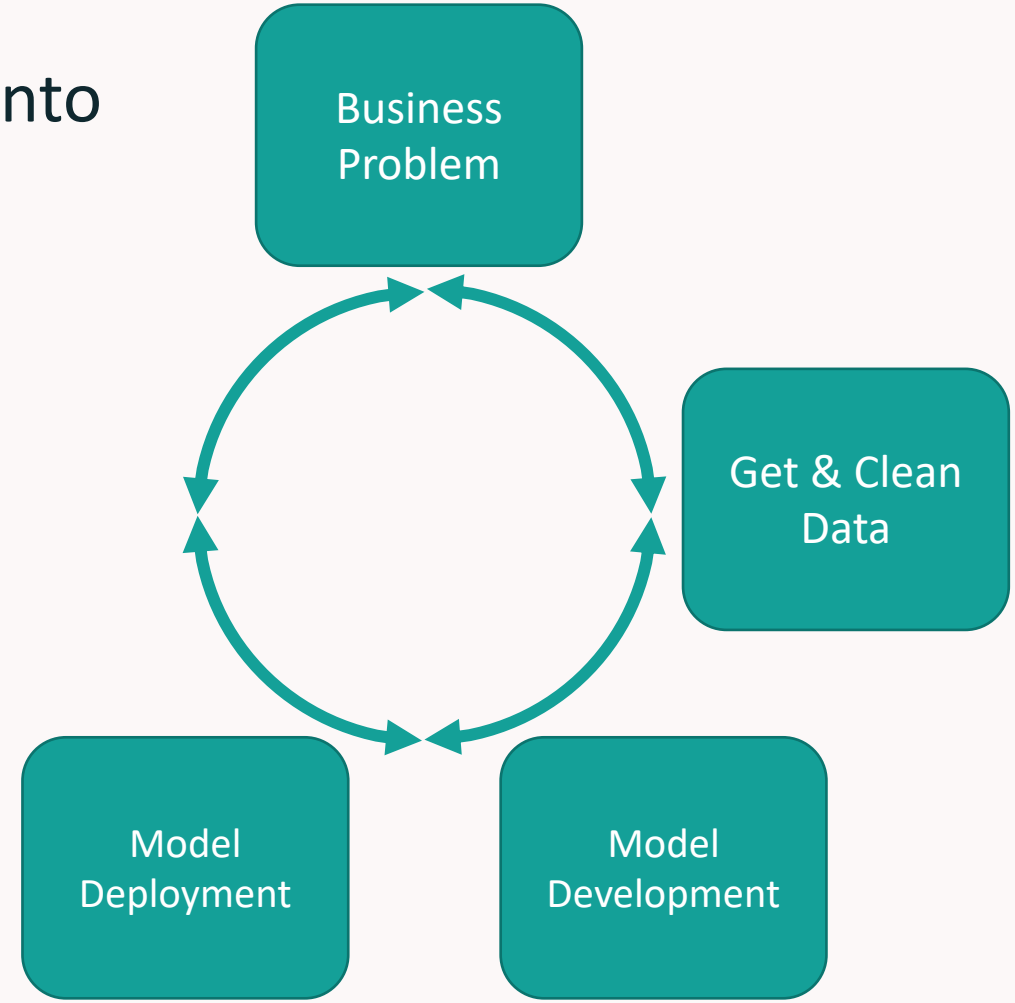
The MLOps Pipeline/Lifecycle

- Do we have infra to train models?
- Do we have development environment?
- Can we keep track of our experiments?



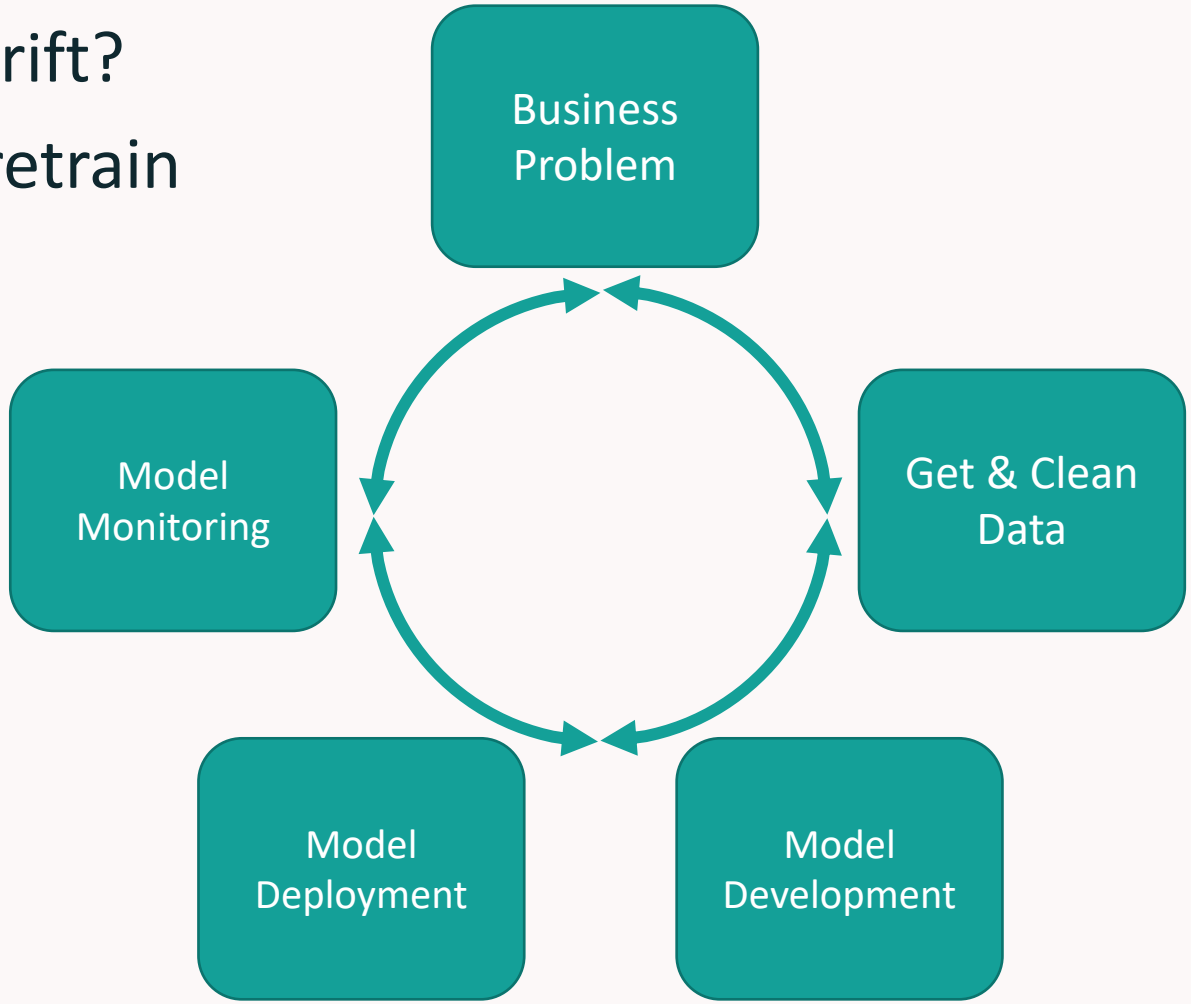
The MLOps Pipeline/Lifecycle

- Can we put model into production?
- Is it scalable?
- Is it secure?
- Is it fault tolerant?



The MLOps Pipeline/Lifecycle

- Does our data drift?
- Do we need to retrain or redevelop?
- Can we explain the predictions?



Why Study MLOps?

Infrastructure

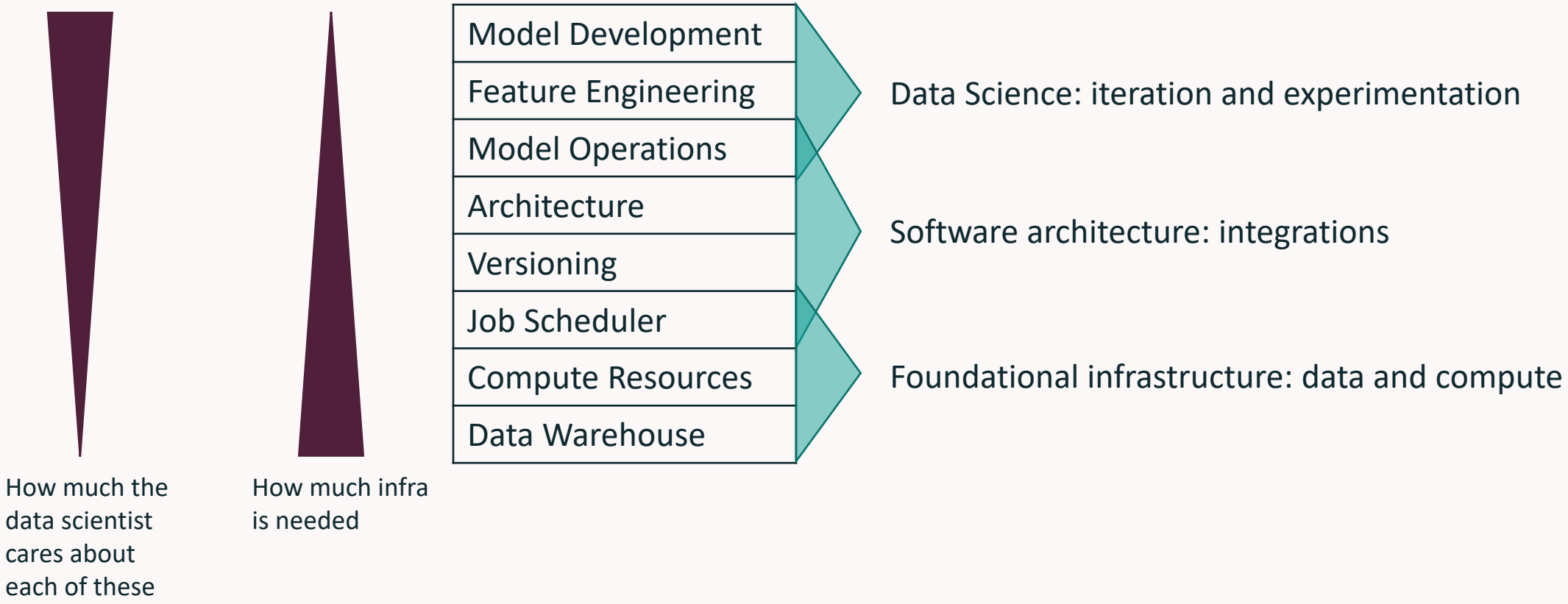
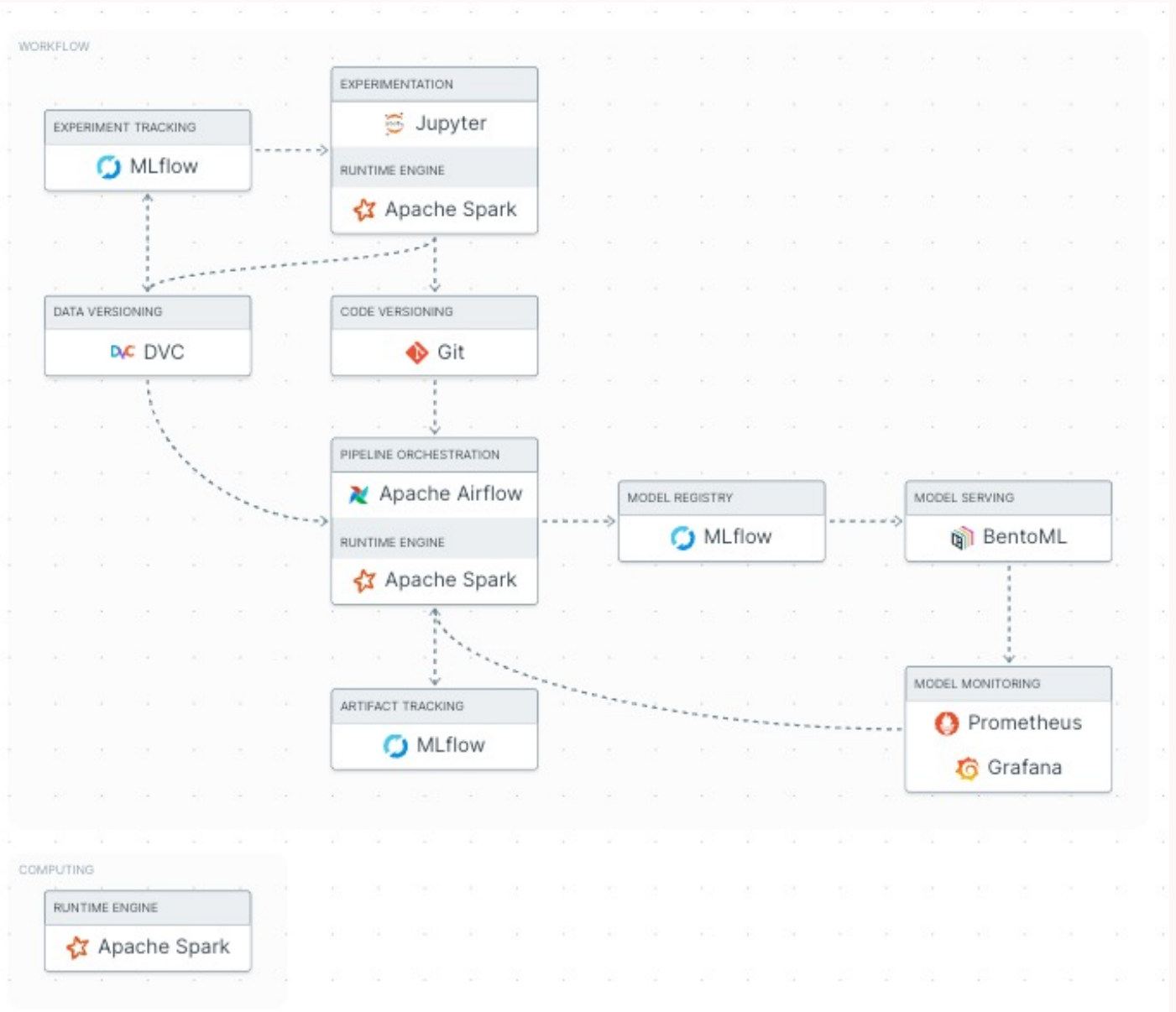


Image reproduced from Effective Data Science Infrastructure

The MLOps Stack

- https://mymlops.com/
  - Contains a non-exhaustive list, but useful to get an idea

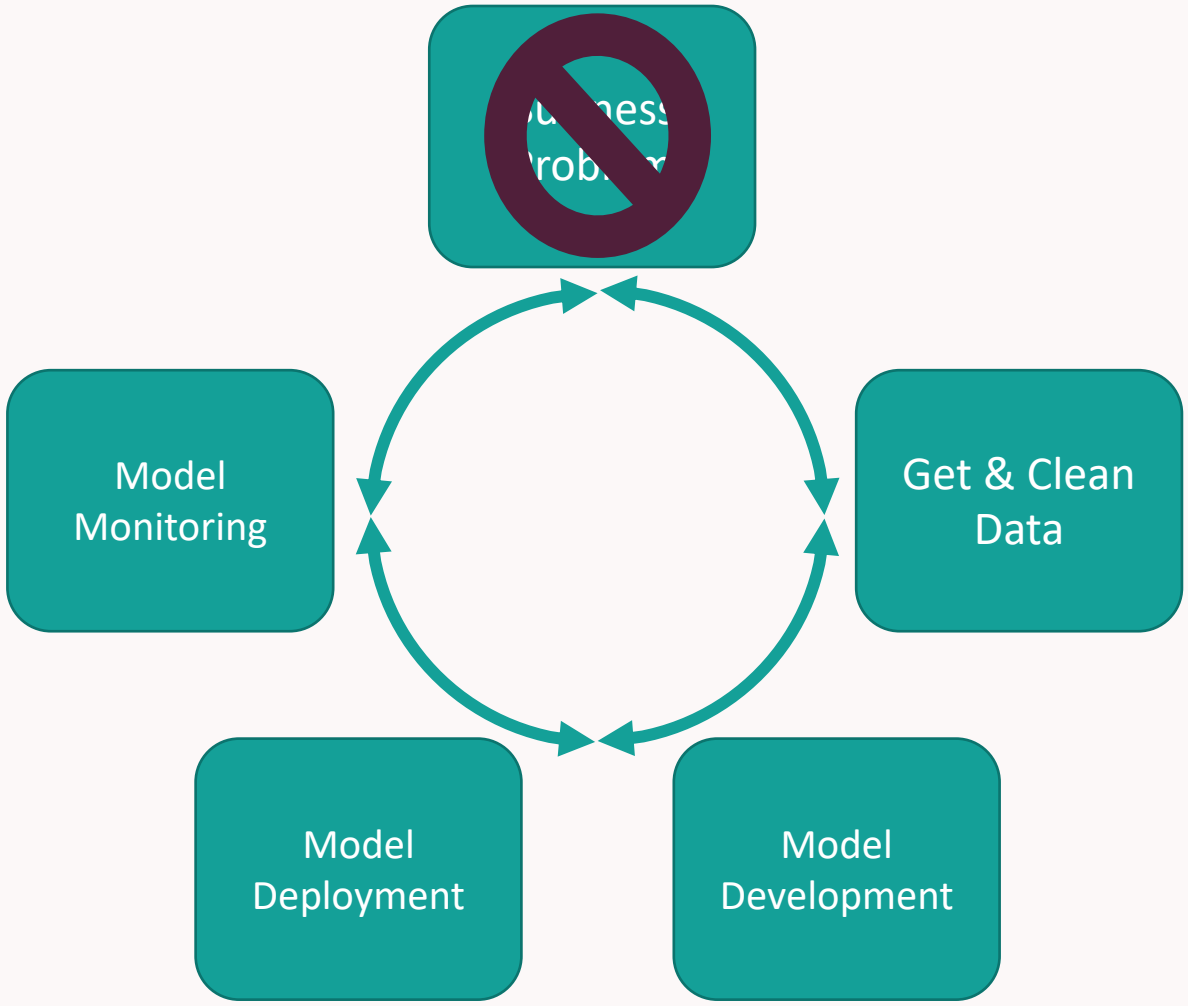


A Side Note about Titles

- Data Scientist – develops and prototypes models/solutions, looks for insights (analytics). A sometimes all-encompassing title for some below roles.
- ML Eng – implements the model in a scalable, production-ready way
- Data Eng – sets up data pipelines for input/output data
- DevOps Eng – deploys applications in prod and maintains them
- Infra/Platform Eng – provide general pieces of infra (data warehouse, compute platforms) for many applications to use
- Product/Program/Portfolio/Project managers, business owners, etc.

MLOps Pipeline Revisited

1. Experiment tracking
2. Artifact/model tracking
3. Code versioning
4. Data versioning
5. Data quality
6. Feature stores
7. ML Orchestration
8. Model serving
9. Model monitoring
10. Model explainability
11. ML Applications



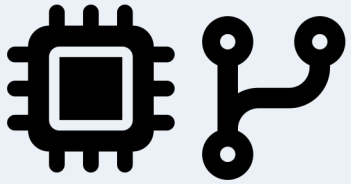
Current Trends

- ML development is getting easier:
  - CV and NLP has tons of pre-trained (**foundation**) models ready for use
    - Huggingface being the most popular example
  - Azure, GCP, AWS offer ml-as-a-service (MLaaS)
    - Though these tend to also be NLP and CV-based, there are some that do time series forecasting and anomaly detection
  - AutoML can do a lot of the tedious work for you
- MLOps has some notable toolkit offerings, some of which we'll be exploring in this course

Environments

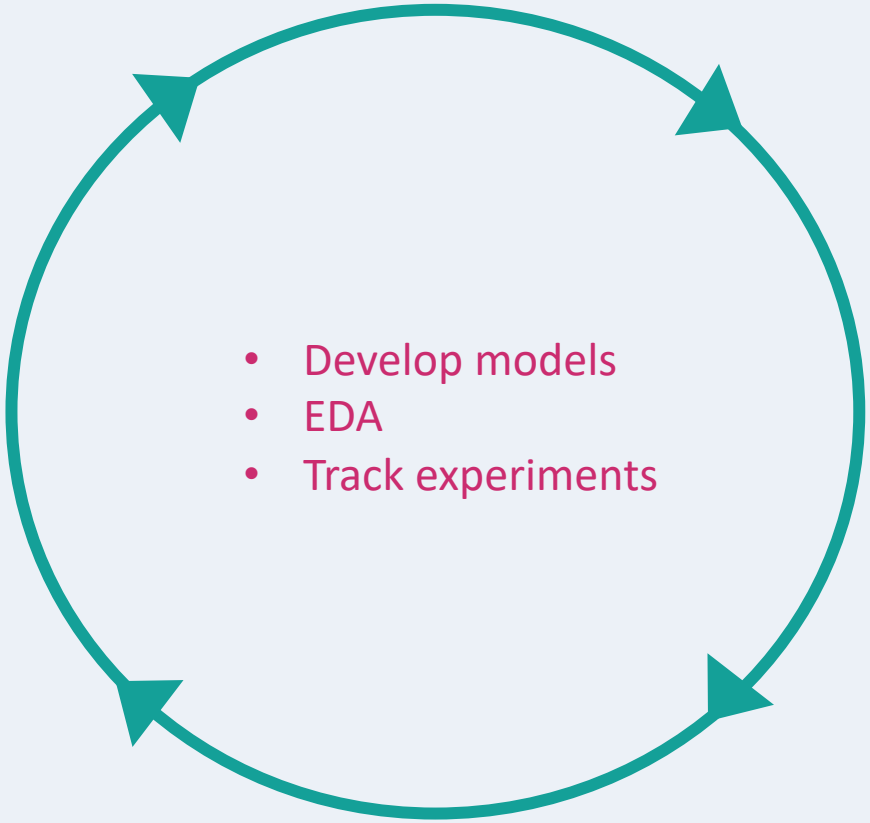
- Dev data & storage
  - Exploration
- Mirror of prod data
- Mix of formats
  - Flat files; S3; SQL/NoSQL; Hive; etc.

DEV



- Use dev branch
- Use dev cluster (compute)

Sandbox environment. Go crazy!

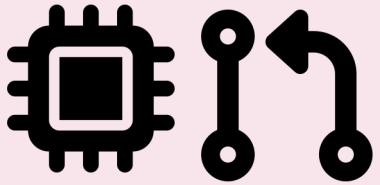


- Develop models
- EDA
- Track experiments

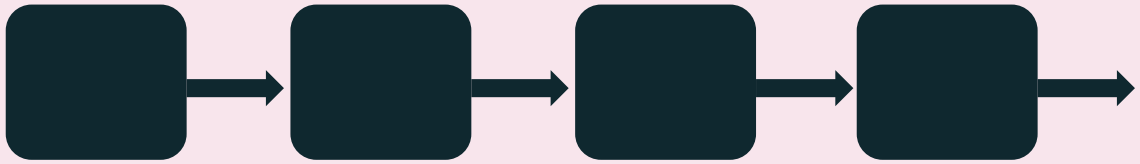
Environments

- Stage data & storage
  - Test ML pipeline
- Mirror of prod data

STAGE



- Merge with staging branch
  - Or alternatively, pull request
- Use stg cluster (compute)

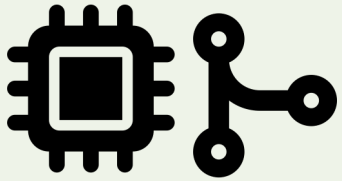


- Trigger ”build”
- Trigger unit/regression/integration tests

Environments

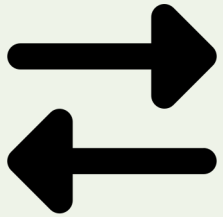
- Prod data & storage
  - “Live” data
- Feature store

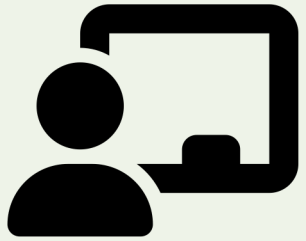
PROD



- Merge with main/release branch
- Use prod cluster (compute)

- Deploy model
  - Batch, streaming, on-demand





- Monitor
  - Input data
  - Output data
  - Model performance

Model Development Environment

Optimal Development Environment

- Optimizes for two things:
  - Rapid prototyping of models
  - Interaction with the production environment

- Laptop vs Cloud
  - Cloud can be an on-prem ”cloud” or true cloud instance (AWS, GCP, Azure, etc.)
  - Laptops are not: secure; scalable; like the production environment; easy to monitor and support

IDEs and Notebooks

- No shortage of options
- Jupyterlab
- **VS Code**, pycharm
- Amazon Sagemaker
- Databricks

Virtual Environments

Virtual Environments

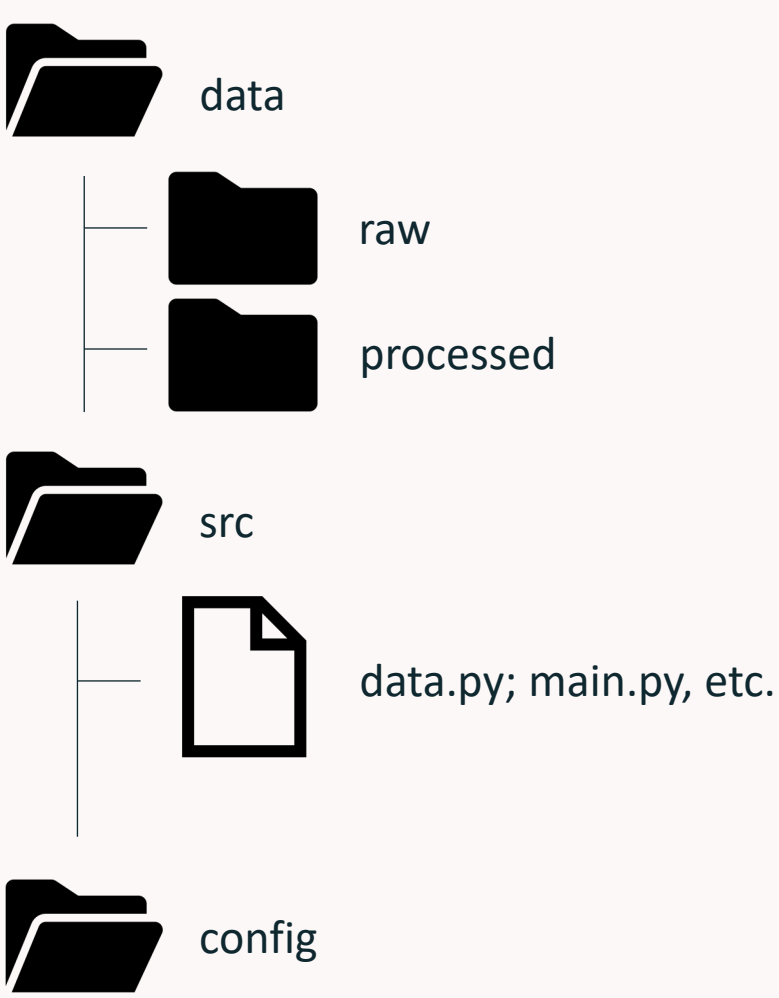
requirements.txt

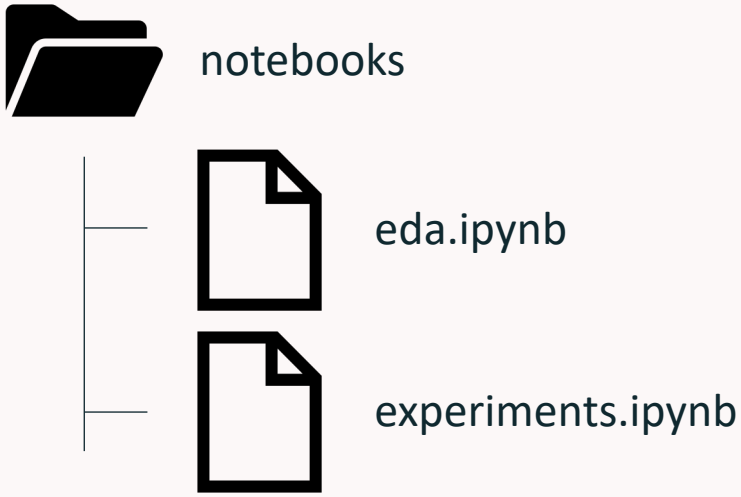
python libraries and versions

setup.py or pyproject.toml

Project metadata and instructions for setting up environment

Project Organization





Do what feels comfortable for you, I am not going to be prescriptive.

Project Documentation

Be explicit

```
def get_random_ingredients(kind: array=None) -> array:
"""
Return a list of random ingredients as strings.

:param kind: Optional "kind" of ingredients.
:type kind: list[str] or None
:raise lumache.InvalidKindError: If the kind is invalid.
:return: The ingredients list.
:rtype: list[str]
"""
        return ["shells", "gorgonzola", "parsley"]
```

Use descriptive doc strings
(use sphinx or mkdocs/mkdocstrings to generate a readthedocs style site).
OR
Create a dedicated site for your project with all the bells and whistles.

Dev Environment Demo and Lab