Strengthen your AI/ML Coding Skills with the

MLOps Coding Course

MLflow Ambassador - 2024-06-05

Médéric HURIER & Matthieu JIMENEZ

About us

Dr. Médéric HURIER

Freelance MLOps Engineer

Working for Decathlon Digital

PhD in Computer Security and AI

https://www.fmind.dev/

Dr. Matthieu JIMENEZ

Research Associate

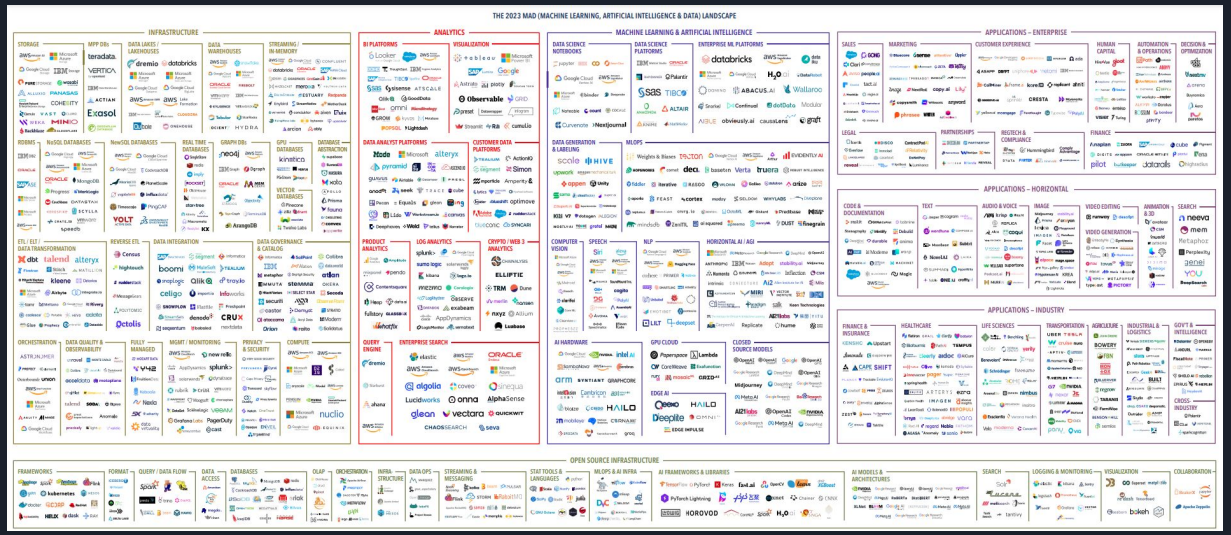
Working for University of Luxembourg

PhD in Computer Security and AI

https://website.jimenez.lu/

Where to begin ?

Many things to know in Machine Learning
(Frameworks, Solutions, Theories, Data ...)



[Mad Landscape 2023]

But what about code related ones ?

Many Software Engineering concepts and technologies apply to ML, especially for production.

One solution is to ask ML engineer, but this result in bottleneck and is hindered by communication issues.

MLOps Python Package

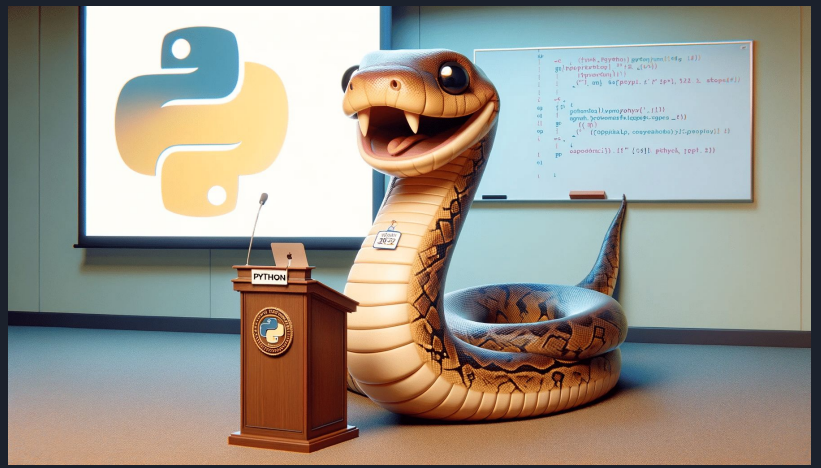
Where it all really started

Created based on an actual MLOps template

Include many OSS Libraries useful for MLOps

Have design patterns integrated out of the box.

Problem how to explain all the notion behind ?



© datascientest

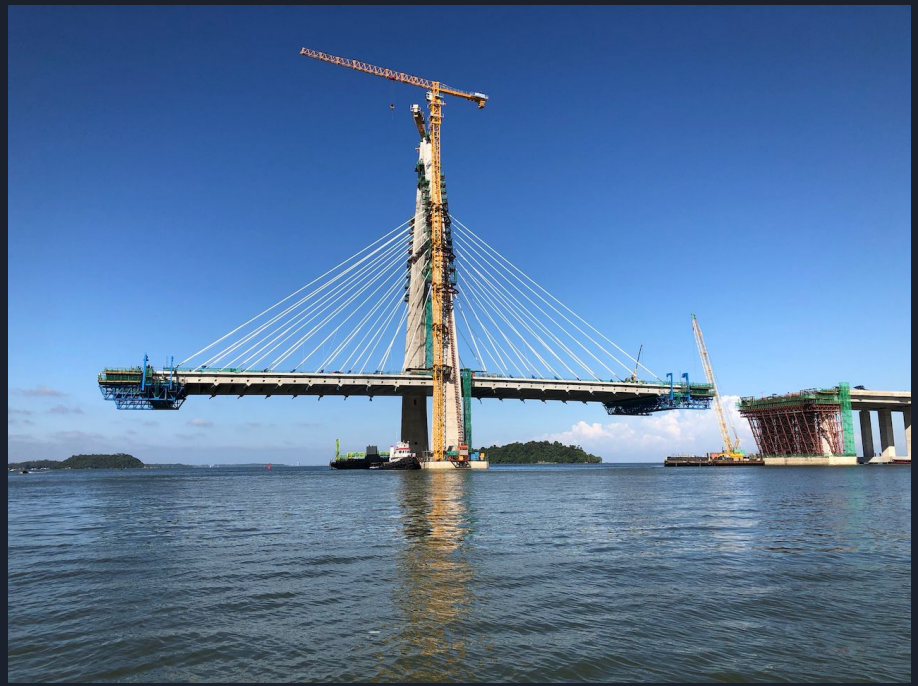
https://github.com/fmind/mlops-python-package

The Idea

Create a course to bridge the knowledge gap

Should be **platform agnostic**
(already many vendor solution courses exist)

Design to be applicable right away (e.g., for Mederic's colleagues at Decathlon)



© wikicommons

Introducing the MLOps Coding Course

Release on 22nd May 2024

7 chapters covering various MLOps topics from the coding perspective

Chapters follow the natural path of a project

Organized around a Q&A system, with additional resources and examples

Coupled to the MLOps Python Package

https://mlops-coding-course.fmind.dev/

Now the course !

0

Overview

Course, Project, Plateforme

1 — 2 — 3 — 4 — 5 — 6

Initializing

Python, pip, pyenv, Poetry, git, Vscode

Prototyping

Notebook, Import, Config, Dataset, Analysis, Modeling

Productionizing

Package, Module, Paradigm, Entrypoint, Configuration, Documentation

Validating

Typing, Linting, Testing, Logging, Security, Debugging

Refining

Design Pattern, Task Automation, CI/CD, Container, Model Registry

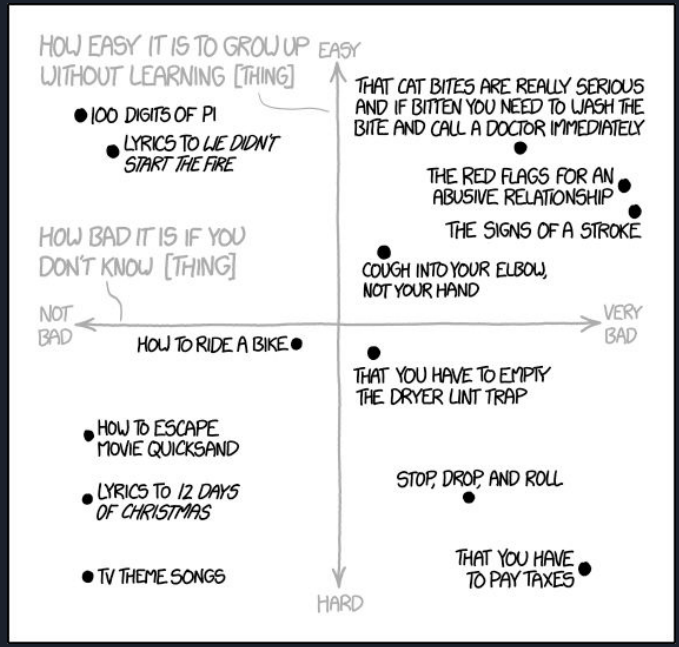
Sharing

Repository, License, Readme, Release, Template

0. Overview: an introduction to the course

Zero based indexing 0️⃣

- ➔ 0.0. Course: What is this course about?
- ➔ 0.1. Projects: Default or bring your own!
- ➔ 0.2. Datasets: On the (un)structure of data
- ➔ 0.3. Platforms: We are platform agnostic here!
- ➔ 0.4. Mentoring: 1-1 learning sessions
- ➔ 0.5. Assistants: A premium RAG for you
- ➔ 0.6. Resources: Going beyond the course

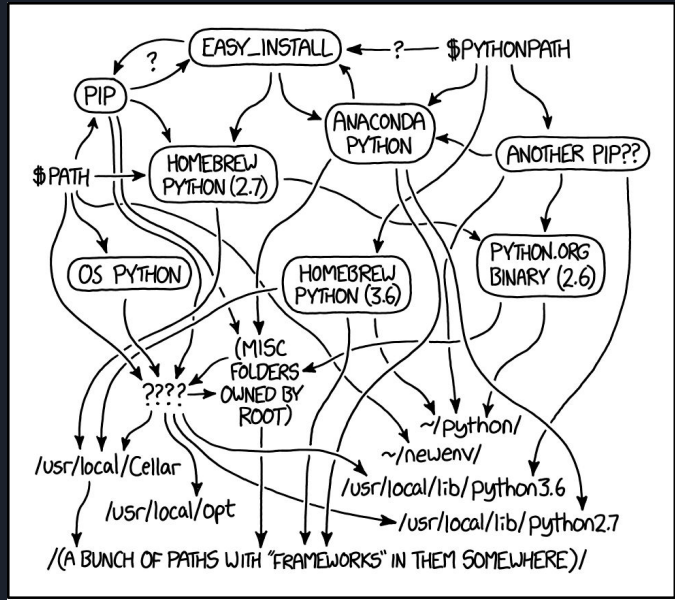


Things You Learn
https://xkcd.com/1775/

1. Initializing: prepare your dev. environment

Let's bring our snakes 🐍

- ➔ 1.0. System: You have my Operating System
- ➔ 1.1. Python: And my Python Interpreter
- ➔ 1.2. Pyenv: And my Python Environment
- ➔ 1.3. Poetry: And my Python Package Manager
- ➔ 1.4. Git: And my Version Control System
- ➔ 1.5. GitHub: And my Code Repository
- ➔ 1.6. VS Code: And my Code Editor

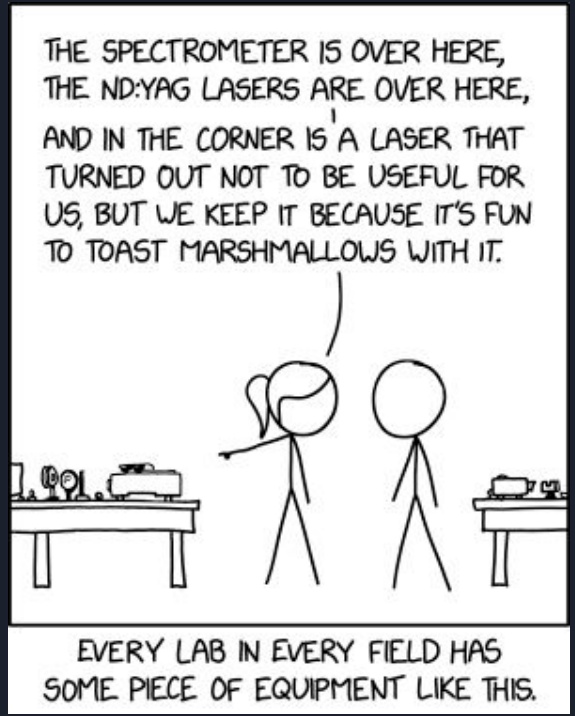


Python Environment
https://xkcd.com/1987/

2. Prototyping: explore your search space

To Jupyter notebooks and beyond 🪐

- ➔ 2.0. Notebooks: To book or not book?
- ➔ 2.1. Imports: By the power of Python ecosystem
- ➔ 2.2. Configs: Separate the code from the settings
- ➔ 2.3. Datasets: How to load and handle datasets
- ➔ 2.4. Analysis: Learning how to speak with data
- ➔ 2.5. Modeling: Create AI/ML models and pipelines
- ➔ 2.6. Evaluations: Put in the science in the modeling

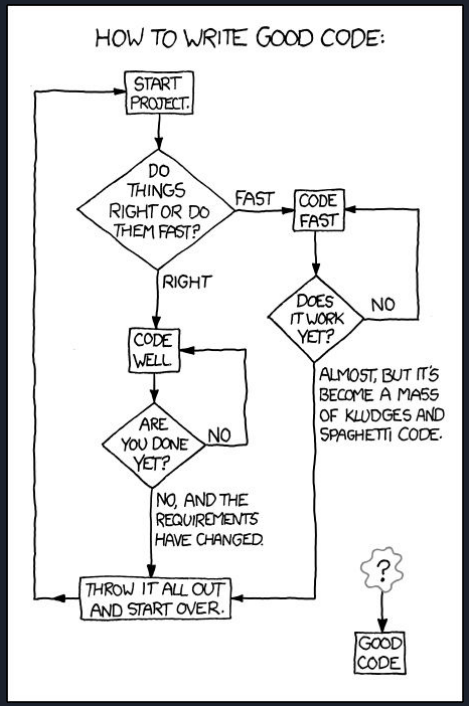


Lab Equipment
https://xkcd.com/2514/

3. Productionizing: go reach your audience

Entering the Industrial Period ⚙️

- ➔ 3.0. Package: Package = Metadata + Code
- ➔ 3.1. Modules: Divide in modules and conquer
- ➔ 3.2. Paradigms: The best programming philosophy
- ➔ 3.3. Entrypoints: Expose packages to other systems
- ➔ 3.4. Configurations: Edit your configs, not your code
- ➔ 3.5. Documentations: Good docs make good friends
- ➔ 3.6. VS Code Workspace: Organize your workspace



Good Code
https://xkcd.com/844/

3. Productionizing - Takeaways

- Building a Python package is key
- Modularity helps separate concerns
- Separate configurations from the code
- Exposed your solution to other systems

4.0 Validating: insurance & reinsurance

No sight of formal methods 🧐

- ➔ 4.0. Typing: Types are schemas
- ➔ 4.1. Linting: Linters are safeguards
- ➔ 4.2. Testing: Tests are executable specs
- ➔ 4.3. Logging: Logging is your intel service
- ➔ 4.4. Security: Security is solution bodyguard
- ➔ 4.5. Formatting: Code in style with formatters
- ➔ 4.6. Debugging: Debugging is the best bug killer

Academia vs. Business
https://xkcd.com/664/

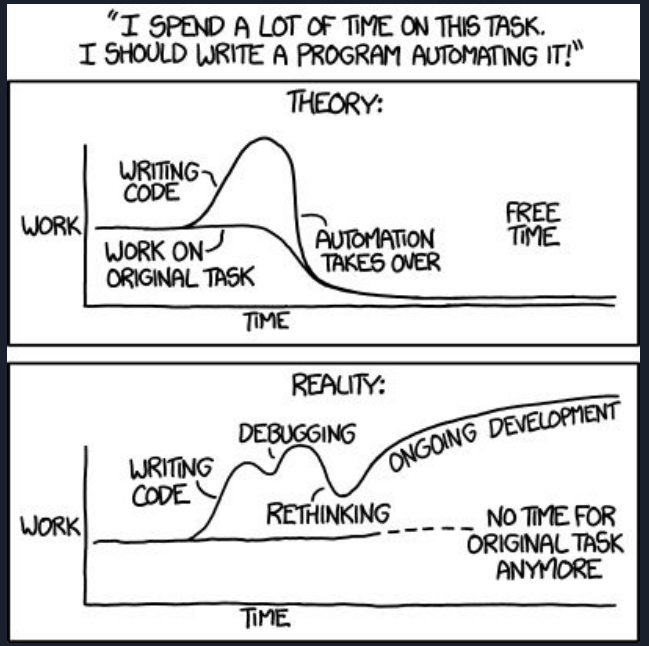
4.0 Validating - Takeaways

- Software validation is an investment ...
- But it has positive returns on the long run
- You don't want to be woken up by a silly bug
- Validations are the developer community rules

5. Refining: advanced concepts for engineers

Software Engineer 2.0 ✌

- ➔ 5.0. Design Patterns: Build SOLID code bases
- ➔ 5.1. Task Automation: Don't repeat yourself
- ➔ 5.2. Pre-Commit Hooks: Your local CI/CD
- ➔ 5.3. CI/CD Workflows: Your shared CI/CD
- ➔ 5.4. Software Containers: Keep It Reproducible
- ➔ 5.5. AI/ML Experiments: MLflow Tracking
- ➔ 5.6. Model Registries: MLflow Model Registry



Automation
https://xkcd.com/1319/

5. Refining - Takeaways

- Automation, automation, automation
- Design patterns are off-the-shelf solutions
- Containers are your go-to for reproducibility
- Tools like MLflow are ML engineer best friends

6. Sharing: build with others for others

Join us now and share the software ♫

- ➔ 6.0. Repository: Reference you project
- ➔ 6.1. License: Tell how people can use it
- ➔ 6.2. Readme: Tell what people can do with it
- ➔ 6.3. Releases: Tell people what have changed
- ➔ 6.4. Templates: Tell people how to build their own
- ➔ 6.5. Workstations: Help people execute your code
- ➔ 6.6. Contributions: Tell people how to contribute

Supported Features
https://xkcd.com/619/

Need a Hand ?

virtual or real, we got you covered !

- MLOPS coding chatbot assistant

  Gemini with the entire course as context, able to answer about any specific part of the class.

  Explain the choice of Q&A

- Mentoring

Demo time

Contributing and Suggestions

The course is Open Source on Github

Don’t hesitate to build upon it (just mention us ;))

If you have any comments, corrections or new content idea, get in touch!

I hope this course will help learners understand experiment/model management through **MLflow**

© DALL.E

Thanks for your attention!

https://mlops-coding-course.fmind.dev

*Join us on the **MLOps Community Slack***