WOMEN WHO CODE
/emerging-tech

Source: ML-Ops.org (by INNOQ Data and AI); [MLOps Principles]

Machine Learning Engineering

A foundation for building scalable ML systems

Bella Nicholson • April 17th, 2024

Who am I?

Bella Nicholson

Machine Learning Engineer

EDUCATION

UNIVERSITY OF AMSTERDAM — MSc Artificial Intelligence

EXPERIENCE

BRENNTAG

crunchr
people analytics for people

Amsterdam UMC
University Medical Centers

Deloitte.

CUBELIZER

Table of contents

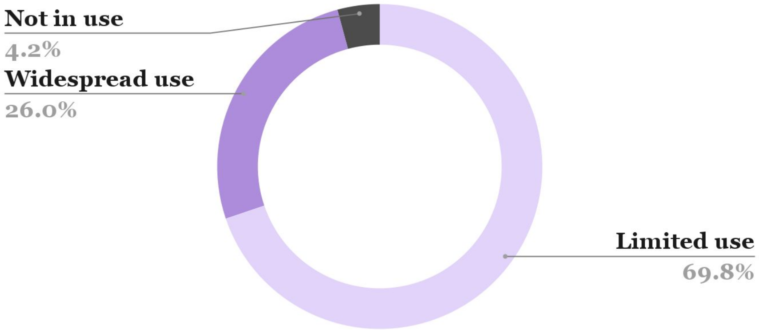
1

Introduction

What is machine learning engineering (MLE)?

The AI Implementation Gap

Global AI Adoption in 2022



New Vantage Partners; [Data and AI Leadership Executive Survey 2022]

42%

Companies are **exploring** AI

34%

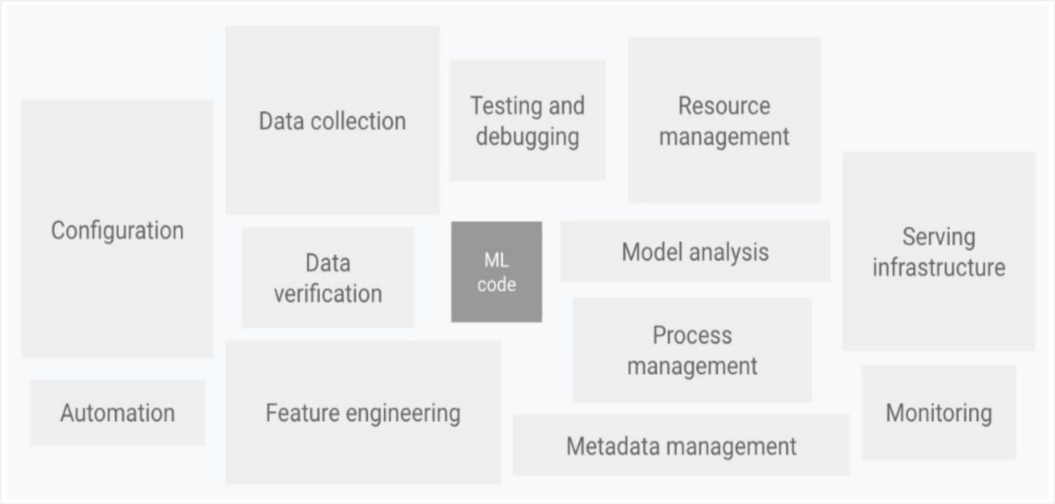
Companies are **deploying** AI

IBM; [Global AI Adoption Index 2022]

What causes this implementation gap?

"Only a **small fraction** of a real-world ML system is composed of the **ML code**"

— Google Cloud Architecture Center; [MLOps Overview]



Challenges of productionalizing ML

1 The real world keeps changing

- Model performance decays over time
- Models need to be monitored & retrained

2 Costs compound quickly

- Cloud services are "pay per use"
- Model retraining and serving at scale causes small inefficiencies to compound

3 The cost of maintainability

- ML products can run for many years
- Effort required to manage legacy code, update Python versions; etc.

Source: Google Cloud Architecture Center; [MLOps Overview]

More challenges come from classical software

Code is fragile

Tiny mistakes can cause major defects, or catastrophic failure

Complexity compounds

How do we separate accidental from essential complexity?

Collaboration has a cost

What's the most effective way for developers to collaborate?

Source: *"Modern Software Engineering: Doing What Works to Build Better Software Faster" by David Farley*

EXAMPLE

Mariner 1

OCCURRED 62 YEARS AGO

TYPE **Flyby**	LAUNCH **July 22, 1962**
TARGET **Venus**	RESULTS **Unsuccessful**

Used R instead of $\overline{R}$ in an equation

What was Mariner 1?

America's first attempt to explore Venus up close was lost to a software glitch. Investigators found a typo caused a fault in the launch vehicle's guidance software. The spacecraft and booster were destroyed shortly after launch for safety.

Costed approx. $18 million,
$170 million in today's money

Sources: *"Mariner 1" by NASA*; *NASA Space Science Data Coordinated Archive*

"The **real challenge isn't building an ML model**; the challenge is building an integrated ML system and operating it in production continuously."

— *Google Inc; [Machine Learning: The High Interest Credit Card of Technical Debt]*

ML Engineering as the Solution

Improved Quality

Prevents bugs and reduce the scope of their effects

Productivity Boost

Empowers teams to develop new features rather than bug fixes

Performance

Optimizes ML systems for speed, efficiency, reliability

Reduced Costs

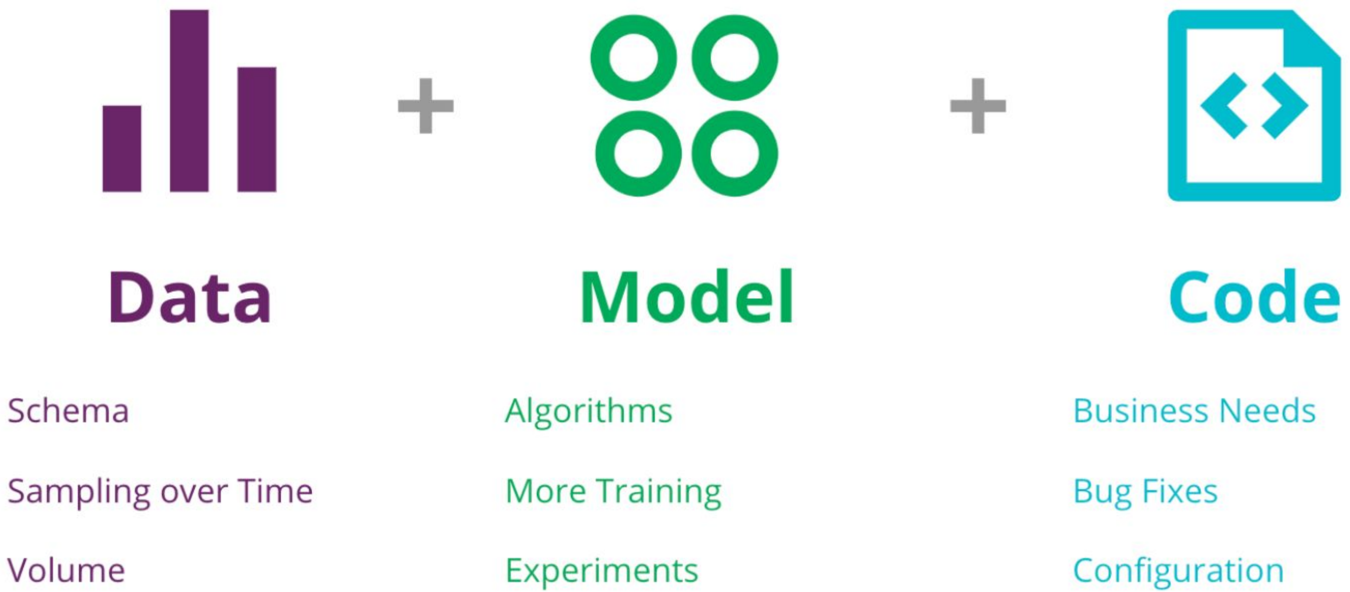
Lowers cloud bills through optimized ML system performance

2

ML Systems

Foundations, development, and maturity

What is a ML system?



Source: [Continuous Delivery for Machine Learning (martinfowler.com)]

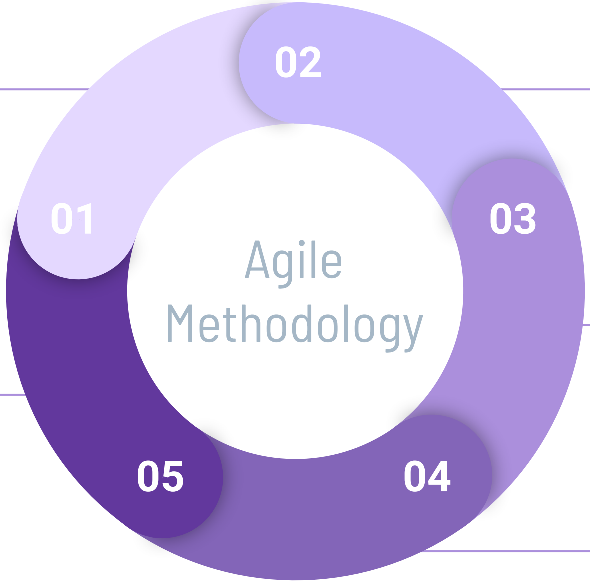
"ML packages have all the basic code complexity issues as normal code, but also have **a larger system-level complexity**"

— *Google Cloud Architecture; [MLOps Overview]*

Software engineering as a starting point

Plan
Gather user inputs, make product designs

Develop
Translate designs into software



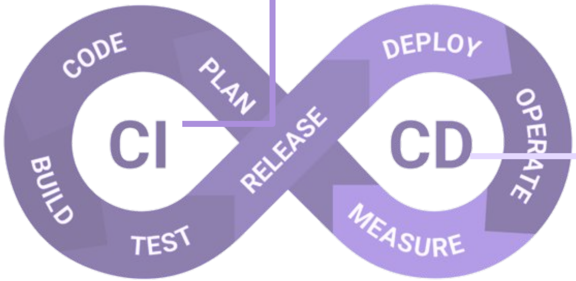
Test
Try to break your code first

Feedback
Observe user behavior, software metrics; etc.

Deploy
Release software

Source: [What is Agile methodology by Asana?]

Software engineering as a starting point



Continuous Integration

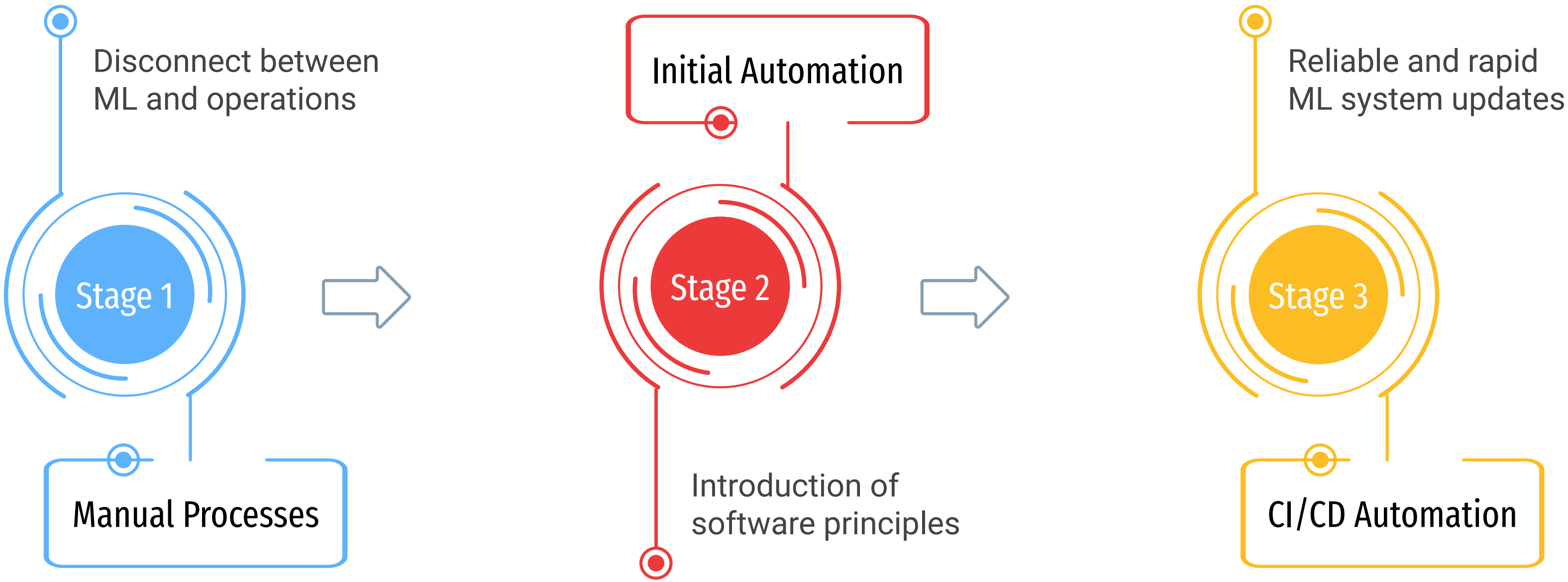
- Automatically **integrate** and package code from **multiple contributors**
- Frequently test code to **catch bugs quickly**

Continuous Delivery

- Run **final tests** in a “mirror” of production environment
- **Automate code release** in a safe, controllable way
- **Monitor** systems once in production

Source: Atlassian; [Continuous integration vs. delivery vs. deployment]

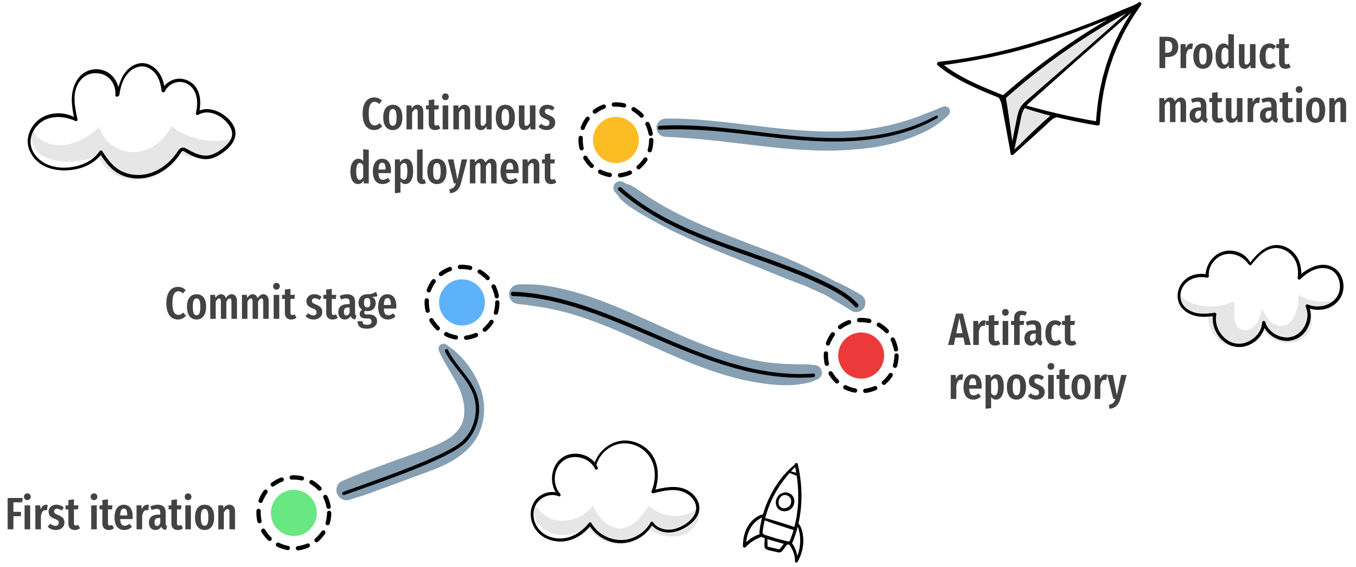
The levels of ML system maturity



Source: Google Cloud Architecture Center; [MLOps Overview]

As a product matures, so does the ML system

*ML system maturity is **a reflection of a product's maturity** – not a team's capabilities*



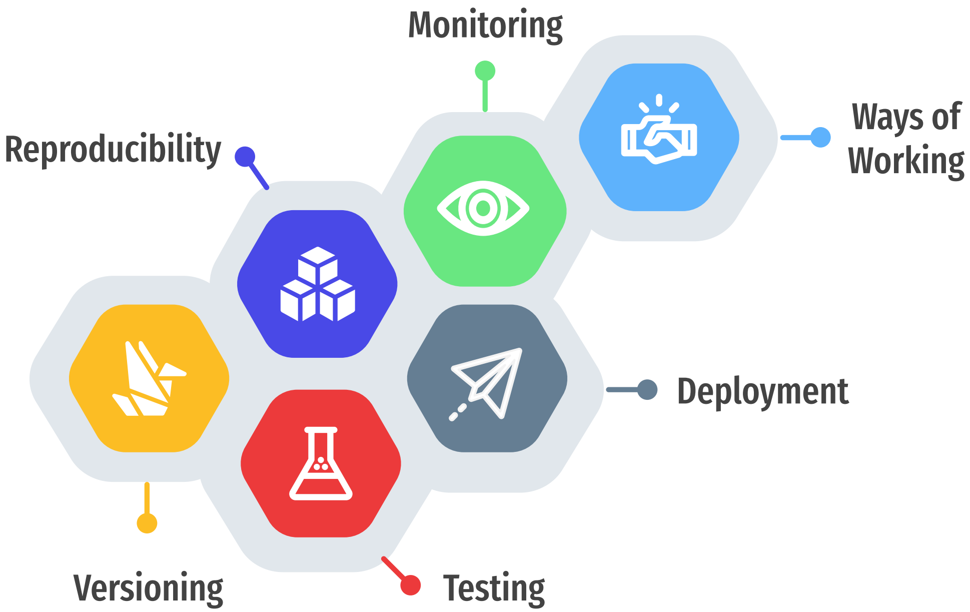
Source: *"Continuous Delivery Pipelines: How to Build Better Software Faster" by David Farley*

3

MLE Principles

6 core principles for understanding ML systems

Aspects of a ML system



Principles of ML Engineering

There's an implicit order to addressing each of these principles

Source: ML-Ops.org (by INNOQ Data and AI); [MLOps Principles]

The requirements for reproducibility

Versioning

Apply version control to:

- Datasets
- Code
- Models

Testing

- Test your software (code) for errors
- Test your data for quality issues
- Test your model to prevent "bad" behavior

Reproducibility

The ability to **recreate** the current state of your **entire ML system**

Source: ML-Ops.org (by INNOQ Data and AI); [MLOps Principles]

The requirements for monitoring

Reproducibility

The ability to **recreate** the current state of your **entire ML system**

Deployment

- Separate development from production code
- Release changes in a safe and controlled manner

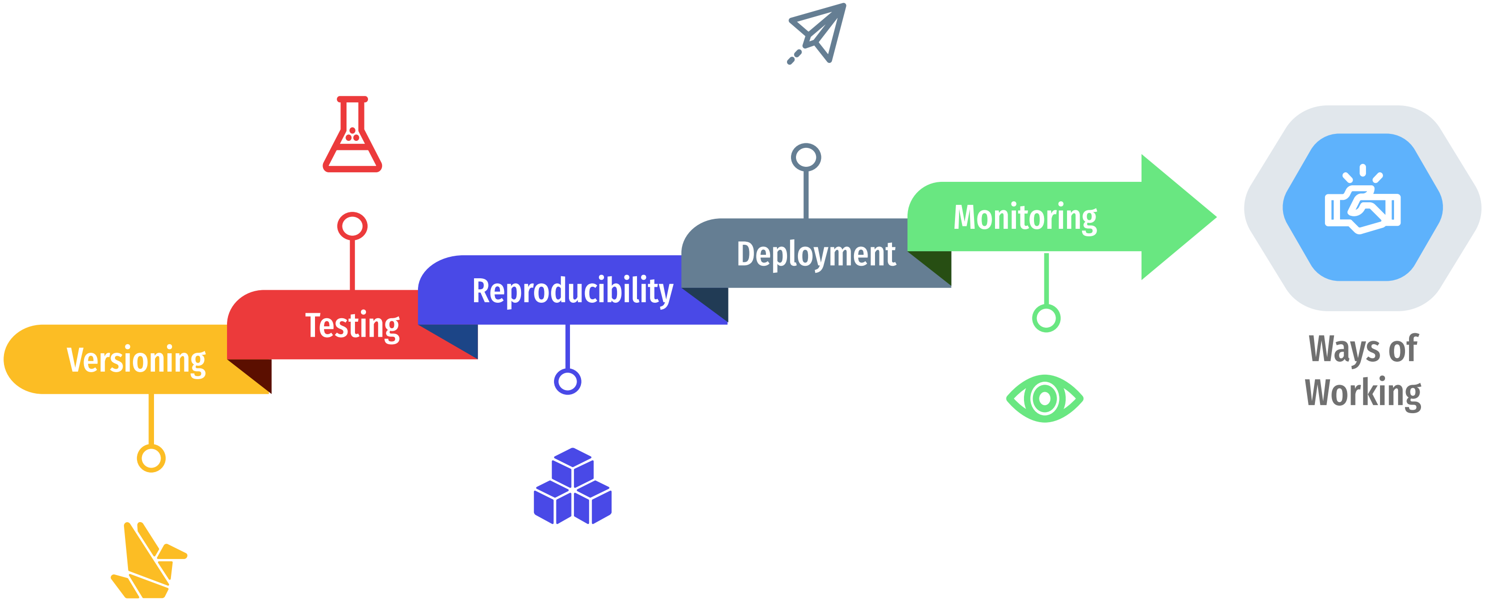
Monitoring

- **Track ML system health** once in production
- Receive **automatic alerts** when things go wrong

Source: ML-Ops.org (by INNOQ Data and AI); [MLOps Principles]

Ways of working

The way a ML Team works is a **reflection of the technical decisions** made with respects to the other 5 MLE principles



EXAMPLES

Business planning

Review process for pull requests

Writing documentation

Source: ML-Ops.org (by INNOQ Data and AI); [MLOps Principles]

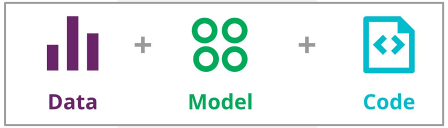
4

Bridging the Gap

Applying MLE theory to real-world systems

Covered ML engineering frameworks

01



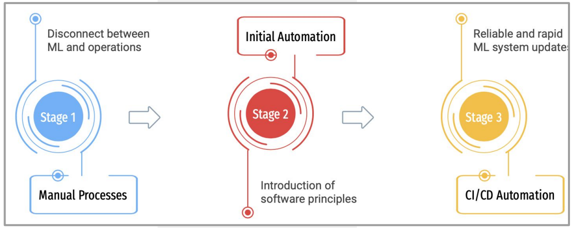
3 Components Framework

from Martin Fowler

+

by Google

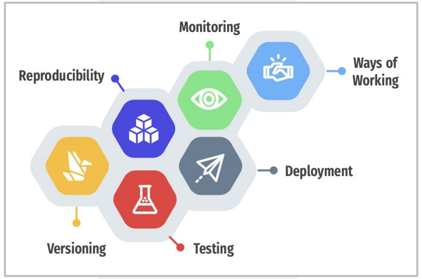
Levels of MLE Maturity



02

+

03



6 Principles of MLE

by ml-ops.org

A unified perspective

Main idea. Think of each framework as a "dimension" of ML system

- Every MLE principle has a code, model, and data aspect
- Each aspect of a MLE principle has maturity score of 1, 2, or 3

Level of maturity is not uniform through your system — or in a given MLE principle

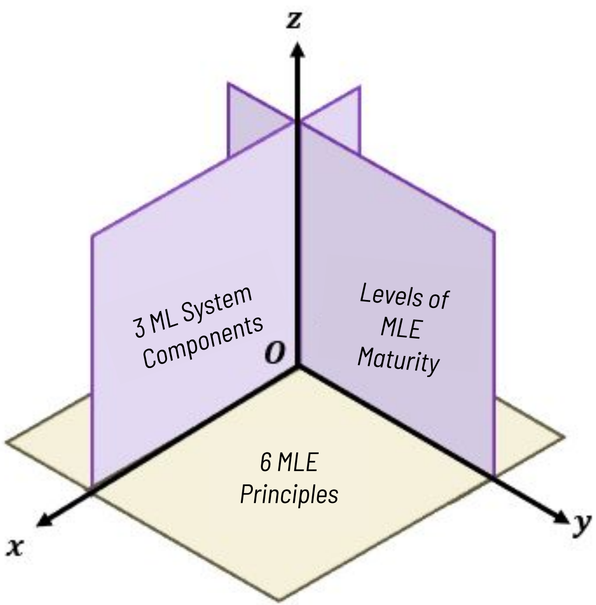


Figure: Adapted from Story of Mathematics

An example of real-life ML system analysis

Versioning

Apply version control to:

- Code
- Models
- Datasets

Code

Are you using a tool like Git to version (tag) the code producing your model?

git GitHub GitLab

Model

Are you saving and versioning your models?

mlflow DVC

Data

Are you versioning the data used to train your models?

FEAST DVC lakeFS

An example of real-life ML system analysis

Testing

- Test your software (code) for errors
- Test your data for quality issues
- Test your model to prevent "bad" behavior

Code

Are there automated tests? If so, what's test coverage percentage? Unit vs. integration testing

Model

Classical model performance evaluation metrics (accuracy, F1-score; etc), Bias and fairness in sensitive applications

Data

Are there data quality checks for raw data? What are the quality standards for your data?

5
Conclusion

Main points and where to learn more

Conclusion

- ML needs to make it into production to incur tangible business value

- MLE is about safely productionalizing code and managing system complexity

Conclusion

ML products are software products

- Require same considerations as "normal" software products
- There are also additional ML-specific considerations

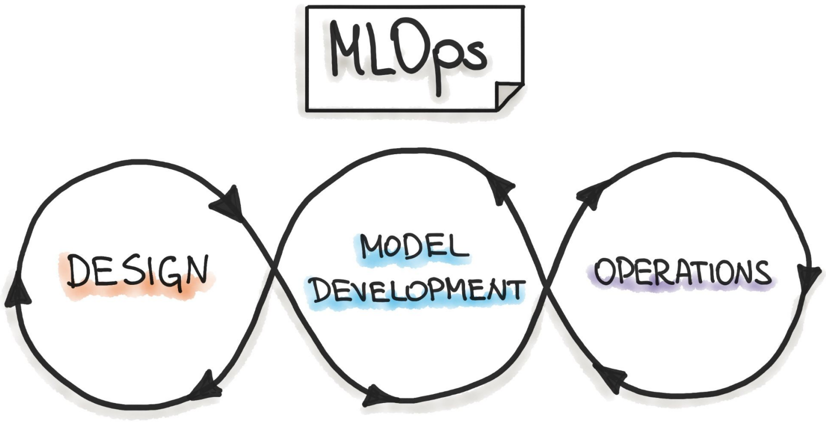


Figure: ML-Ops.org (by INNOQ Data and AI); [MLOps Principles]()

Want to learn more?

Software Engineering Fundamentals

- "Beyond the Basic Stuff with Python: Best Practices for Writing Clean Code" by Al Sweigart
- "Computer Science 101: Mastering the Theory Behind Programming" by Kurt Anderson
- "Software Development from A to Z – Beginner's Complete Guide" by Karoly Nyisztor
- "Software Engineering 101: Plan and Execute Better Software" by Kurt Anderson
- "Git Complete: The definitive, step-by-step guide to Git" by Jason Taylor
- "User Story Masterclass: Your Agile Guide to User Stories" by BA Guide
- "Building Maintainable Software: Ten Guidelines for Future-Proof Code" by Joost Visser
- "AWS Cloud Practitioner Essentials" by AWS

Want to learn more?

Software Engineering Deep Dive

- "Code Diagnosis Workshop" by ArjanCodes
- "SOLID principles Made Easy in Python" by ArjanCodes
- "GRASP Design Principles: Why They Matter (And How to Use Them)" by ArjanCodes
- "Modern Software Engineering: Doing What Works to Build Better Software Faster" by David Farley
- "Continuous Delivery Pipelines: How To Build Better Software Faster" by David Farley
- "Design Patterns" by Refactoring Guru

Machine Learning Fundamentals

- Machine Learning Specialization by DeepLearning.AI
- University of Amsterdam's Deep Learning Tutorials by Phillip Lippe
- Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications by Chip Huyen

MLOps

- The Big Book of MLOps by Databricks
- MLOps Overview by Google's Cloud Architecture Center
- Machine Learning Engineering for Production (MLOps) Specialization by DeepLearning.AI
- Rules of Machine Learning by Martin Zinkevich (Google Developers)
- Machine Learning: The High-Interest Credit Card of Debt by Google Research
- Continuous Delivery for Machine Learning by ThoughtWorks and Martin Fowler
- Guide to Evaluating MLOps Platforms by Thoughtworks

Legend

Website

Free

YouTube

Free

Udemy

€10 - €15 if on sale

Book

€10 - €45 for ebooks, depends on retailer

Coursera

USD 50 per month

Thanks!

Do you have any questions?

bellanich.software@gmail.com

bella-nicholson github.com/bellanich