Machine Learning Operations

Chuks Okoli

Last Updated: May 22, 2024

Introduction

Imagine you have a magic cookbook that can teach you how to make the best cookies ever. But it's not just any cookbook, it's a special one that learns and gets better every time you make cookies.

Now, making cookies is a bit like solving a puzzle. You need the right ingredients (like flour, sugar, and chocolate chips) and you need to mix them together in just the right way. Sometimes, you might make a mistake and your cookies don't turn out quite right.

That's where MLOps comes in! MLOps is like having a team of helpful fairies who watch over you while you're baking. They make sure you're using the right ingredients, they help you mix everything together perfectly, and they even give you tips on how to make your cookies even better next time.

So, with MLOps, you can keep making cookies and your magic cookbook will keep learning from each batch. And before you know it, you'll be making the most delicious cookies anyone has ever tasted!

— Chuks Okoli

Hello there, and welcome to Machine Learning Operations. This work is a culmination of hours of effort to create my reference for machine learning operations. All of the explanations are in my own words but majority of the content are based on Alexey Grigorev's DataTalksClub MLOps Zoomcamp course.

1.1 What is MLOps?

> *MLOps, also known as DevOps for machine learning, is an umbrella term that encompasses philosophies, practices, and technologies that are related to implementing machine learning lifecycles in a production environment.*
>
> — Microsoft Blog

Machine Learning Operations (MLOps) is a set of best practices for putting machine learning models into production. The process for a machine learning project involves:

- Design - define if machine learning is the right tool for solving the problem
- Train - train the model to find the best possible model
- Operate - deploy the model, and monitor degradation or quality of the model

MLOps is a set of practices for automating everything and working together as a team on a machine learning project.

Fun Fact: MLOps Principles

As machine learning and AI become more common in software, it's important to create guidelines and tools for testing, deploying, managing, and monitoring ML models in real-world use. This is where MLOps comes in. It helps prevent "technical debt" in machine learning projects by ensuring smooth operation and maintenance of models throughout their lifecycle.

MLOps helps to manage and orchestrate the end-to-end machine learning lifecycle by ensuring models are consistently accessible, reproducible, and scalable. It focuses on automating deployment and monitoring of ML pipelines while optimizing the model development process.

The three-phase approach to implementing machine learning (ML) solutions are:

- **Business Understanding and Design**: This phase involves identifying user needs, designing ML solutions to address them, and assessing project development. Prioritizing ML use cases and defining data requirements are key steps. The architecture of the ML application, serving strategy, and testing suite are designed based on functional and non-functional requirements.
- **ML Experimentation and Development**: This phase focuses on verifying

the applicability of ML by implementing Proof-of-Concept models. It involves iteratively refining ML algorithms, data engineering, and model engineering to deliver a stable, high-quality ML model for production.

- **ML Operations**: Here, the emphasis is on deploying the developed ML model into production using established DevOps practices. Testing, versioning, continuous delivery, and monitoring are essential aspects of this phase.

These phases are interconnected, with design decisions influencing experimentation and deployment options.

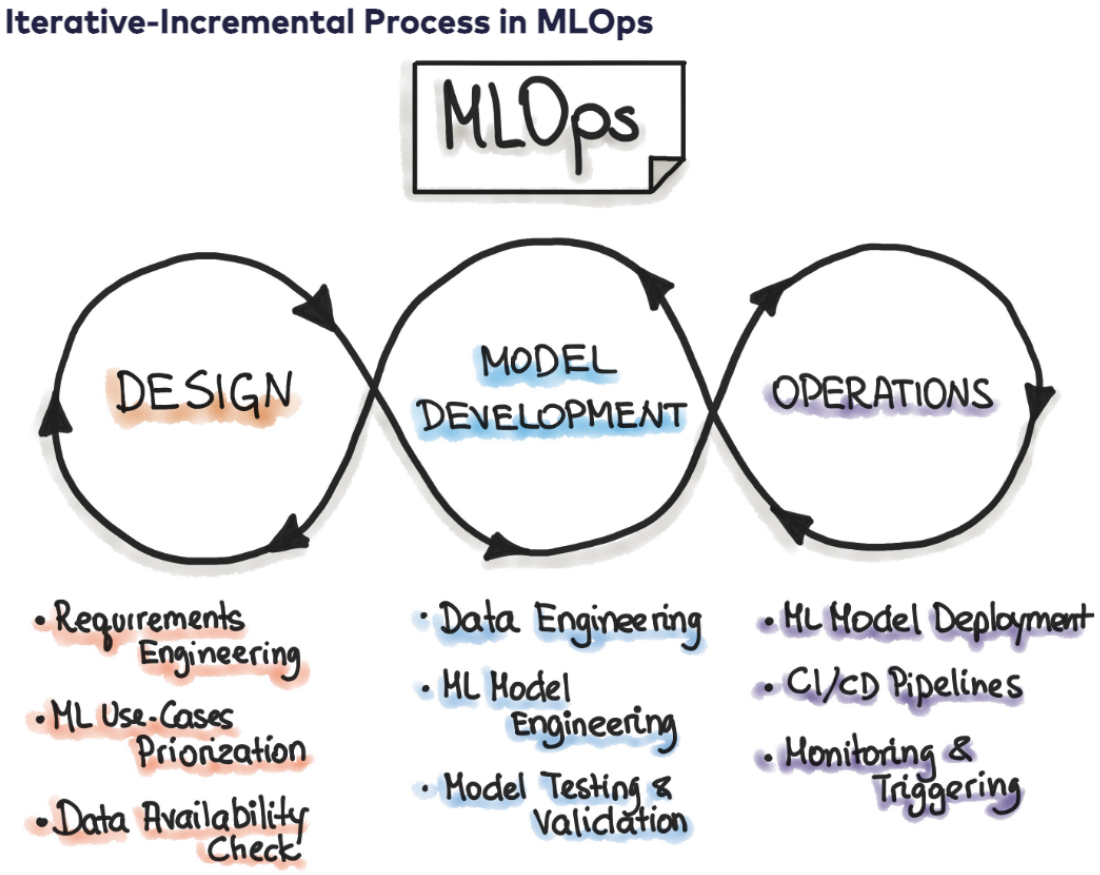

Figure 1.1: The complete MLOps process includes three broad phases of "Designing the ML-powered application", "ML Experimentation and Development", and "ML Operations".

MLOps leverages various tools to simplify the machine learning lifecycle.

– **Machine learning frameworks** like Kubernetes, TensorFlow and PyTorch for model development and training.

– **Version control systems** like Git for code and model version tracking.

- **CI/CD tools** such as Jenkins or GitLab CI/CD for automating model building, testing and deployment.
- **MLOps platforms** like Kubeflow and MLflow manages model lifecycles, deployment and monitoring.
- **Cloud computing platforms** like AWS, Azure and IBM Cloud provide scalable infrastructure for running and managing ML workloads.

1.2 Environment Preparation

1.2.1 Configuring Environment with GitHub Codespaces

To configure the environment using GitHub codespaces, first create a repository on GitHub, give the repository a name, add a "README" file and a `.gitignore` template, choose a license and create the repo. In the repo main page, click on `Create codespace on main` as shown in **Fig. 1.2**.

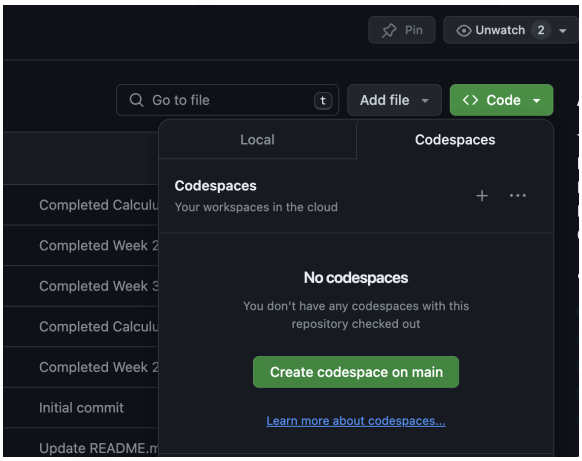

Figure 1.2: GitHub Codespaces setup

A connection to Visual Studio Code can be made through Codespaces. Start a new terminal, change directory to the workspaces and download and install Anaconda distribution of python in it using the step below.

- **Step 1**: Download and install the Anaconda distribution of Python

```
wget https://repo.anaconda.com/archive/Anaconda3-2022.05-Linux-
    x86_64.sh
```

- **Step 2**: Run this command:

```
bash Anaconda3-2022.05-Linux-x86_64.sh
```

After installing Anaconda, initialize it. In a new terminal, confirm that Anaconda is running in the workspaces

```
(base) @your_username -> /workspaces/mlops-zoomcamp-2024 (main) $
   which python
/home/codespace/anaconda3/bin/python
```

- **Step 3**: Make sure to install `pyarrow` in order to download parquet file

```
!pip install pyarrow
```

- **Step 4**: Run jupyter notebook

```
jupyter notebook
```

1.2.2 Configuring Environment with Amazon Web Service (AWS)

To install using AWS, create an account in AWS, go to EC2 and create an instance. Select the OS to use e.g. Ubuntu with 64-bit(x86) architecture. Select the instance type that will be sufficient for your project. Configure the cloud resources and launch. Recommended development environment: Linux

- **Step 1**: Download and install the Anaconda distribution of Python

```
wget https://repo.anaconda.com/archive/Anaconda3-2022.05-Linux-
   x86_64.sh
bash Anaconda3-2022.05-Linux-x86_64.sh
```

- **Step 2**: Update existing packages

```
sudo apt update
```

- **Step 3**: Install Docker

```
sudo apt install docker.io
```

To run docker without sudo:

```
sudo groupadd docker
sudo usermod -aG docker \$USER
```

- **Step 4**: Install Docker Compose Install docker-compose in a separate directory

```
mkdir soft
cd soft
```

To get the latest release of Docker Compose, go to https://github.com/docker/compose and download the release for your OS.

```
wget https://github.com/docker/compose/releases/download/v2.5.0/
   docker-compose-linux-x86_64 -O docker-compose
```

Make it executable

```
chmod +x docker-compose
```

Add the soft directory to PATH. Open the .bashrc file with nano:

```
nano ~/.bashrc
```

In .bashrc, add the following line:

```
export PATH="${HOME}/soft:${PATH}"
```

Save it and run the following to make sure the changes are applied:

```
source ~/.bashrc
```

- **Step 5**: Run Docker

```
docker run hello-world
```

1.3 Model Development

1.3.1 Taxi trip prediction with machine learning

EXAMPLE 1.1: **TLC Trip Prediction**

In this section, we use machine learning to predict the trip duration for trips in New York. The data used were collected from the NYC Taxi and Limousine Commission (TLC). This dataset contains yellow and green taxi trip records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. The data is stored in the PARQUET format. We built a simple linear regression model and save the model as a pickle file for subsequent use later one. See the duration prediction notebook.

We can download parquet file using:

```
# download parquet data
!wget https://d37ci6vzurychx.cloudfront.net/trip-data/
   green_tripdata_2021-01.parquet
```

1.4 MLOps Maturity Model

The MLOps maturity model helps clarify the Development Operations (DevOps) principles and practices necessary to run a successful MLOps environment. The

maturity is the extent to which MLOps is implemented in a team. The MLOps maturity model encompasses five levels of technical capability.

Level	Description	Highlights	Technology
0	No MLOps	• Hard to manage lifecycle • Disparate teams • Systems as "black boxes"	• Manual builds/deployments • Manual testing • No centralized tracking
1	DevOps but no MLOps	• Less painful releases • Limited feedback • Hard to reproduce results	• Automated builds • Automated tests for application code
2	Automated Training	• Managed, traceable training • Easy to reproduce model • Low friction releases	• Automated model training • Centralized tracking of model training performance • Model management
3	Automated Deployment	• Releases are low friction and automatic • Full traceability from deployment back to original data • Entire environment managed: train > test > production	• Integrated A/B testing • Automated tests for all code • Centralized tracking
4	Full MLOps Automated Operations	• Fully automated, monitored • Systems provide improvement information • Zero-downtime	• Automated training/testing • Centralized metrics

Table 1.1: Machine Learning operations maturity model (culled from Microsoft Blog

Experiment Tracking and Model Management

2.1 Introduction to Experiment Tracking

Machine Learning experiment is the process of building an ML model. Experiment run represents each trial in an ML experiment. A run artifact is any file associated with an ML run. The experiment metadata is all the information related to the experiment. *Experiment tracking* is the process of keeping track of all the **relevant information** from an **ML experiment**, which includes:

- Source code
- Environment
- Data
- Models
- Hyperparameters
- Metrics

> **Why is Experiment Tracking so important**
>
> Experiment tracking is important because of **Reproducibility**, **Organization**, and **Optimization**

Experiment tracking is done using MLflow. MLflow is "an open source platform for the machine learning lifecycle". In practice, it's just a Python package that can be installed with pip, and it contains four main modules:

- Tracking
- Models
- Model Registry
- Projects

The MLflow Tracking module allows you to organize your experiments into runs, and to keep track of:

- Parameters
- Metrics
- Metadata
- Artifacts
- Models

Along with this information, MLflow automatically logs extra information about the run:

- Source code i.e., the name of the file that was used to run the experiment
- Version of the code (git commit)
- Start and end time
- Author

2.2 Experiment tracking with MLflow

To start experiment tracking, we need to create a conda environment for tracking experiment and activate it.

```
conda create -n exp-tracking-env python=3.9
```

2.3 On Colorful Boxes

There are lots of different boxes you can make:

- Like this one,

  which is wrapped in gray. I use it for notes....

- Or this one,

  which is wrapped in red. I use it for fun facts or other asides...

- Or this one,

  which is wrapped in blue and used for mathy stuff.

- Or this last one,

  which is wrapped in green. With a title, it's used for enumerated examples (see `\extitle` and `\excounter`). Observe:

Example 2.1: Test

This is an example. What's the answer to $2 + 2$?

Answer: Obviously 4, lol.

Example 2.2: Test Again

This one will increment the counter automatically, resetting for each chapter.

- For red and blue boxes, there are custom commands for titles, too:

One Title

Like this

Two Titles: A Subtitle

Or this

These styles also automatically apply to theorems and claims.

Theorem 2.1 (Pythagorean Theorem)**.** *For any right triangle with legs a, b and hypotenuse c:*

$$a^2 + b^2 = c^2 \tag{2.1}$$

Proof. This is left as an exercise to the reader. ■

Claim 2.1. *This is the greatest note template in the world.*

There are different ways to quote things, too, depending on how you want to emphasize:

> *This is a simple, indented quote with small letters and italics usually suitable*

for in-text quotations when you just want a block.

Alternatively, you can use the `\inspiration` command from the chapter heading, which leverages the `thickleftborder` frame internally, but adds a little more padding and styling (there's also just `leftborder` for a thinner variant):

> Hello there!

2.4 On Cross-Referencing

This is the standard way to include margin notes. There are also commands to link to source papers directly (see `\lesson`).

You can reference most things—see Theorem 2.1 or (2.1) or the Introduction chapter—directly and easily as long as you give them labels. These are "built-ins." However, you can also create a **custom term** that will be included in the index, then include references to it that link back to the original definition. Try clicking: custom term. Building the index is on you, though. You can also reference by using a different term for the text: like this. Sometimes it doesn't fit the **grammatical structure** of the sentence so you can define the term one way and visualize it another way (this creates a **grammar** entry in the index). There's also **math terms** and a way to reference them: math terms (clickable), but they do **not** show up in the index.

2.5 On Math

Most of the math stuff is just macros for specific things like the convolution operator, $\circledast$, probabilities, $\Pr[A \mid B = C]$, or big-O notation, $\mathcal{O}(n^2 \log n)$ but there's also a convenient way to include explanations on the side of an equation:

$$1 + 1 \stackrel{?}{=} 2 \qquad \text{first we do this}$$
$$2 \stackrel{?}{=} 2 \qquad \text{then we do this}$$
$$2 = 2 \quad \blacksquare$$

These are all in the `CustomCommands.sty` file.