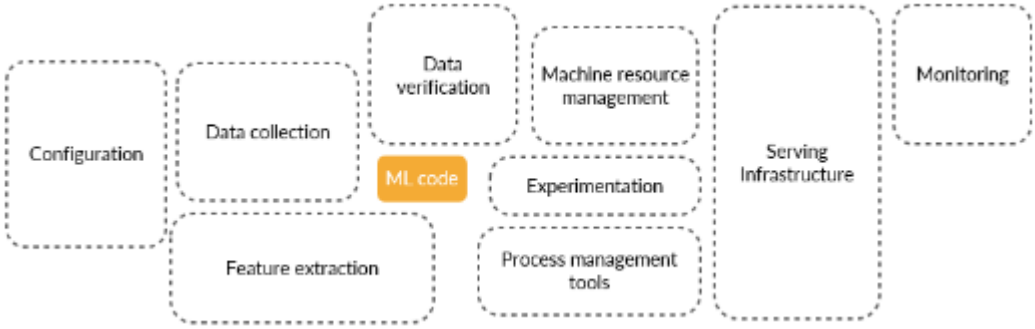
upGrad

# Building Blocks of an MLOps Stack

Without MLOps practices, taking any ML model from ideation to the deployment phase is extremely time-consuming. But with MLOps practices, this time can be reduced. Next, focussing on the MLOps life cycle as a process is more important than knowing the tools used in the different life cycle aspects. For example, several tools are used in the ML model training process, such as PyCaret, Airflow, and MLflow, among others. To choose the best tool for your model, you must be well versed in the development process and infrastructure. Also, understanding each aspect of the process will help you choose the appropriate tool.




The image given below shows the different parts of the MLOps life cycle. It is taken from “Practitioners guide to MLOps: A framework for continuous delivery and automation of machine learning”. It is highly recommended to read this paper to get a deeper understanding of MLOps. You can find the paper here.

upGrad

# ML Development


You have done model development in your jupyter notebook environment, but for industrial ML solutions, we have to do it in a rapid manner. We have many steps to do exploratory data analysis, which can be done by one line of code by using **Pandas Profiling**.

Similarly, model development and hyperparameter tuning can be automated using tools like **PyCaret** and **Optuna** that facilitate rapid experimentation.

A few tools for performing rapid experimentation are shown in the image given below. Some of these tools allow the process to be automated with just a single line of code. You will learn about these in the upcoming modules.

To aid the experiment tracking process, there are tools that help in tracking different experiments that you perform in your project. You can log the metrics and compare and choose the best model for deployment. Tools used in experiment tracking are shown in the image provided below.


1. **neptune.ai:** Neptune helps you automate and standardise things when your team grows or modelling gets to scale. It lets you log, organise, visualise, register, and share all your models and experiments in a single place.
2. **MLflow:** It is an open-source platform to manage the ML lifecycle, including experimentation, reproducibility, deployment, and a central model registry
3. **Valohai:** Valohai is an MLOps platform that automates everything from data extraction to model deployment.
4. **Weights & Biases:** It is an MLOps platform to build better models faster with experiment tracking, dataset versioning, and model management
5. **ClearML:** Easily Develop, Orchestrate, and Automate ML Workflows at Scale

There are multiple tools available other than the ones mentioned above. With all these steps, you come up with a baseline model**.** As you can see, all the tools help with more or less similar kinds of things, but there are some differences which you can explore later when you are choosing the tool apt for your use case. We'll be using MLflow for experiment tracking in future modules.

# Training Operationalization & Continuous Training

Training operationalisation focusses on automating training by converting Notebooks to scripts, following the best practices around modular coding and testing, among others.

Jupyter Notebooks are not a good choice for industrial solutions, and hence, you converted the Notebooks to Python scripts for the heart disease use case.Therefore, it is crucial to figure out

the lines of code that will be used to build the solution. Figuring out the code and then converting Jupyter Notebooks to modularised Python scripts using recommended software engineering practices, such as **code modularisation, code optimisation, exceptional handling** and **code documentation**, helps in the easy deployment of scripts in production systems.

Now that you have converted your notebooks into scripts, it is time to automate the training process where these scripts will be used. To automate the process, you have an automated process for continuous training.

You saw that the continuous training pipeline consists of other pipelines, such as the continuous integration and continuous testing pipelines. The continuous integration pipeline helps integrate the code changes made to the data or the model pipelines due to issues detected at any other stage in the MLOps life cycle. The continuous testing pipeline ensures that the data and model pipelines, after integration, pass the required tests. The continuous training pipeline helps in retraining the model as and when required. Some tools that help create these pipelines are shown in the image given below.


# Model Deployment

You already know that you use the final trained model to make inferences or predictions on any new data. You have also learnt earlier that for deploying the trained model, you do not need to repeat the steps you have performed to train the model in the development phase. You just need to apply certain data transformations and run the model to make predictions on the unseen data. So, model deployment is nothing but the creation of an inference pipeline and then the continuous integration and deployment (CI/CD) of the same.

There are specific CI/CD tools for this task, which are shown in the image given below.


Similarly, there are tools, as shown below, that help in the deployment of solutions.


# Prediction Serving

Prediction serving is the process of providing model predictions to end users. This may vary depending upon who your end user is..

One of the most common methods for prediction serving is the creation of **APIs**. API helps in the creation of an endpoint through which you connect the model to the application. The results that the model generates can be relayed through the API to the application.

The image given below shows the tools used to create APIs to serve model results.





There are multiple ways to serve model results in an MLOps system, depending on the end user. You saw that in the case of song recommendation systems, the recommendations have to be generated in near real-time, and hence, prediction serving is done in real-time. Similarly, you must have seen suggestions of messages that you get on LinkedIn while interacting over chat with someone or while replying to a post. Here, the suggestions are quick and in real-time with your reply. Hence, the prediction serving is real-time.

However, you also saw that in the customer churn example, the end users are internal teams who plan to target churn predicted customers with some offers. Thus, prediction serving here is a batch process and can be done through an API, a JSON file or a CSV file, depending on what works best for the end user.

After successful delivery, the model needs to be monitored to detect anomalies, if any, and keep a check on the model's performance.

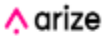
## Continuous Monitoring

In the continuous monitoring phase, we monitor the system's performance, the model's performance and certain elements specific to the ML solutions, like **data drift, concept drift** and model bias.

In a software system, we monitor the complete system to check for delays in responses, errors and exceptions. But in ML solutions, the focus is more on the data and the performance of the model.

Given below are some tools used in the industry for continuous monitoring.

upGrad








# Data And Model Management

Data and model management is essential for regulating ML artefacts and ensuring their integrity, traceability and compliance. Data and modelling management can help make ML assets more shareable, reusable and discoverable. Data and model management employs security management of different versions of data and models.

To maintain the security of a solution, it is necessary to understand the infrastructure of the ML system. This is done with the help of the following components:

**Compute:** This indicates the computational power of an ML model. Some models, like deep learning models, require higher computational power to generate predictions, whereas some simple models require low compute power. So, the compute power is decided on the basis of the algorithm used in the MLOps system.

**Storage:** To store different versions of the data and the model, the storage part of the infrastructure should be large. It should also ensure the security of the versions that are getting stored.

**Cloud, On-Prem or Edge:** There are different ways to deploy an ML model. Most ML models are deployed on cloud systems, as the cloud can be accessed from anywhere. Some ML models are deployed on on-prem systems that are located in specific premises. Unlike cloud deployment, these can be accessed only within the premises of the system and not remotely. Other than these, ML models can run on edge devices, such as cell phones, smart watches, Raspberry Pi and Arduino, which are portable and lightweight but with restricted storage and compute power.

The selection of the deployment method while building an MLOps system should be based on the type of problem and end users.

upGrad

# 10X Data Scientist

For building a successful data science career:

- It is important to understand software engineering to make a career in ML engineering to be able to integrate ML solutions with existing software systems.
- While writing code, it is important to document, review and refactor the code
- You should always ensure that you write clean code for others to understand the logic and the thought process behind writing the code. This also helps in the collaboration of large codebases.
- Try thinking holistically to come up with best solutions that help end users meet their goals.
- Finally, try keeping the solution simple, as complex solutions can confuse business stakeholders.