09.24 : MLOps Concepts

What is MLOps?

MLOps : Machine Learning Operation

set of practices to design, deploy and maintain machine learning in production continuously, reliably, and efficiently.

실제 business process에 이용되는 머신러닝 과정.

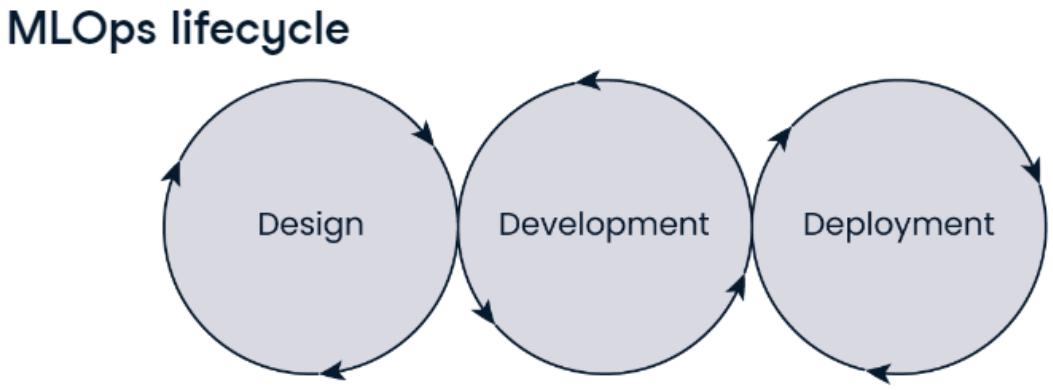
High quality data와 feature engineering으로 training.

model performance 측정.

continuous monitoring after deployment

Machine learning과 operation team간의 gap을 줄이는 것이 MLOps의 목적!

deploy와 model performance 측정함.야



Design phase

- Clarify problem context and assess ML value.
- Gather business requirements to define success.

- Establish key metrics to track progress.
- Ensure high-quality data processing for model robustness.
- Engage stakeholders to evaluate project viability.

Development phase

- Dive into creating the machine learning model.
- Experiment with data, algorithms, and hyperparameters.
- Train multiple models and evaluate performance.
- Iterate on designs based on results.
- Aim for a well-tuned model ready for real-world deployment.

Deployment phase

- Integrate the model into existing business processes.
- Ensure seamless operation within the larger system.
- Build a microservice for easy access and scalability.
- Set up monitoring to track performance and detect data drift.
- Monitor predictions for degradation and set alerts.
- Ensure long-term effectiveness and value delivery.

Roles in MLOps

business stakeholder

- The business stakeholder or product owner manages budget decisions.
- Ensures project aligns with the company’s vision.
- Defines business requirements during the design phase.
- Assesses experiment results in the development phase.
- Verifies outcomes in the deployment phase.

subject matter expert

- The subject matter expert (SME) has domain knowledge about the problem.

- Involved throughout the lifecycle to assist with data and results interpretation.
- Supports technical roles at each step of the process.

Data scientist

- The data scientist handles data analysis, model training, and evaluation.
- Responsible for monitoring the model post-deployment for valid predictions.
- Involved in all phases but primarily active during the development phase.

data engineer

- The data engineer collects, stores, and processes data.
- Ensures data quality and implements tests to maintain it.
- Involved before model training, during training, and in production.

ML engineer

- The machine learning engineer is a versatile role covering the entire ML lifecycle.
- Cross-functional and overlaps with other technical roles.
- Involved in all phases of the lifecycle.
- Skilled in data extraction, storage, model development, and deployment.

Additional roles involved in ML

data analysts, developers, software engineers, and backend engineers. and more versatile in startup.

Machine learning design

Added value estimation

- Predict whether a customer will churn

100K customers

$10 per month

80% accuracy predicting churn

1,000 customers churn

50% decrease of churn

1,000 customers x 80% x 50% = 400 customers p/m

400 x discounted subscription $8 = $3200 per month

Business requirements

- End user
  - Speed
  - Accuracy
  - Transparency
- Compliance and regulations
- Budget
- Team size

Key metrics

The data scientist looks at the accuracy of a model, how many times the algorithm is correct.

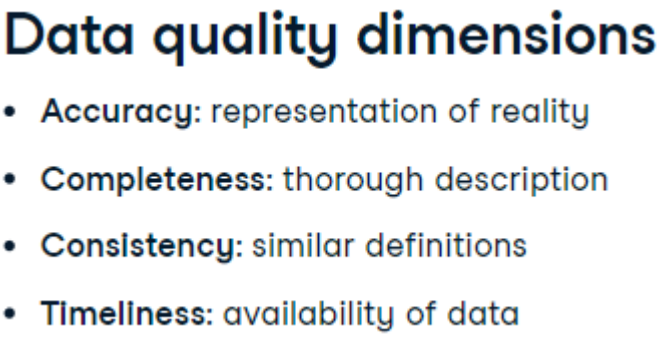
The subject matter expert is interested in the model's impact on the business. They are primarily interested in domain-specific metrics.

The business stakeholder focuses on the monetary value of the model.

Align metrics to ensure all stakeholders are on the same page for maximum value from ML.

Data quality and ingestion

- ML model quality heavily depends on data quality.
- Data is central to the machine learning model's success.
- Poor data quality negatively impacts model performance.
- Improving data quality is often the first step to enhancing model performance.



In the design phase, we plan how to extract and process data.

- Automated data pipelines are used for this process.
- ETL (Extract, Transform, Load) is a common data ingestion method.
- Data is extracted, transformed into the needed format, and loaded into a database.
- Automated checks (e.g., temperature column contains only numbers) can be included.
- These checks prevent faulty data and speed up the development and deployment phases.

Feature engineering and the feature store

- Feature engineering involves selecting, manipulating, and transforming raw data into features.
- A feature is a variable, like a column in a table.
- Creating features from data is crucial in ML development.

- Features can be used as-is from raw data or custom-built for specific needs.

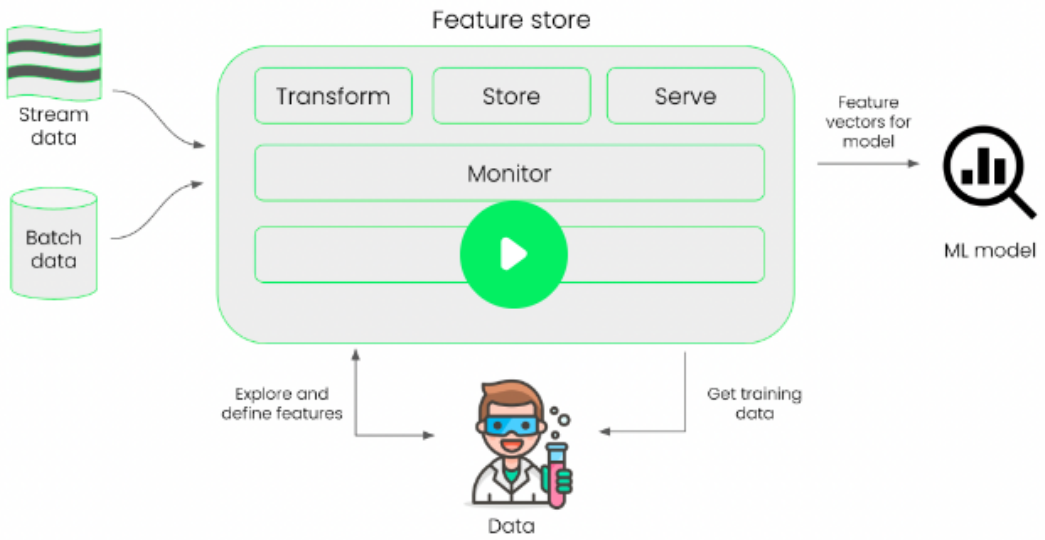
Feature engineering weigh-off

More features can

- produce a very accurate model
- achieve more stability
- be more expensive due to additional pre-processing steps
- require more maintenance
- lead to noise, or over-engineering

- A data scientist may initially perform feature engineering once for a single project.
- As the number of ML projects and models grows, managing features becomes more complex.
- Storing features centrally can speed up ML model development.
- A feature store is an important MLOps tool for storing and managing commonly used features.

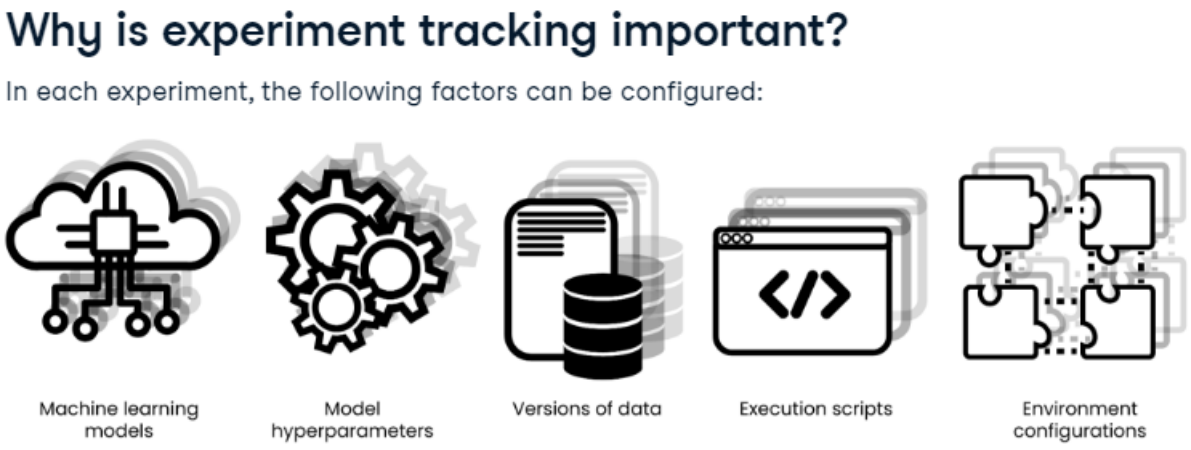
The feature store

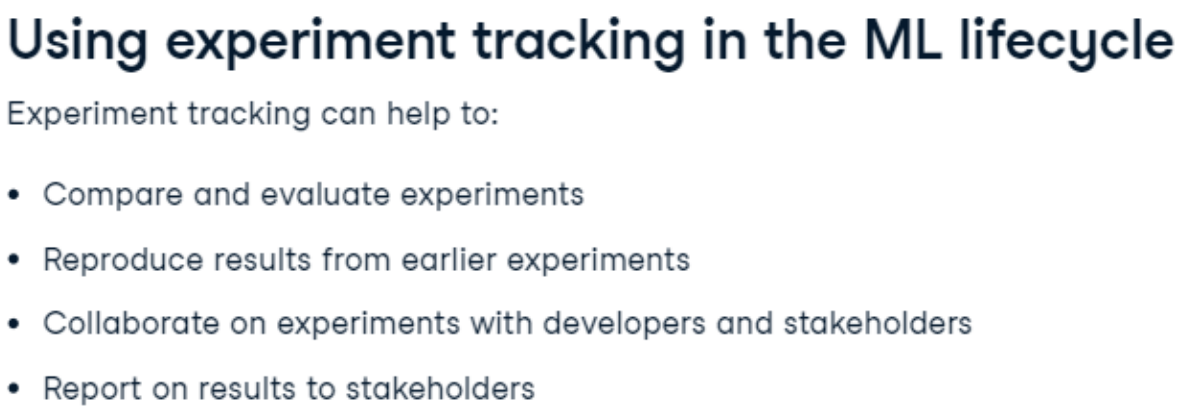


When to use a feature store?

- A feature store isn't always necessary for ML model development.
- Consider the computational cost of features before using a feature store.
- Some features may be ready for the model without further processing.
- The number of projects also influences the decision to use a feature store.
- These factors help determine if a feature store will benefit the current ML development.

Experiment tracking



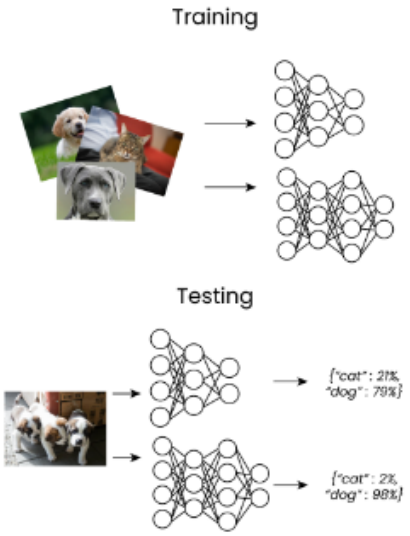


How to track experiments?

Tool	Pro	Con
Spreadsheet	Straightforward, easy to use	Require a lot of manual work
Proprietary platform	Custom solution specific for our process	Require time and effort
Experiment tracking tool	Specifically designed for experiments	Can be expensive

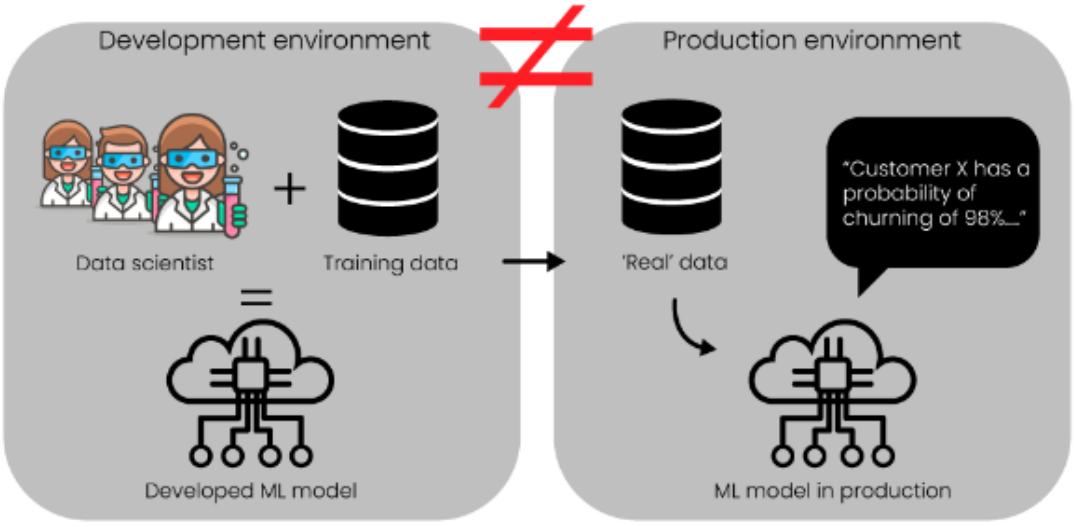
The experiment process

1. Formulate a hypothesis: *"We expect that..."*
2. Gather images and labels
3. Define experiments, e.g., types of models, hyperparameters, datasets
4. Setup experiment tracking
5. Train the machine learning model(s)
6. Test the models on a hold-out test set
7. Register the most suitable model
8. Visualize and report back to team and stakeholders, and determine next steps

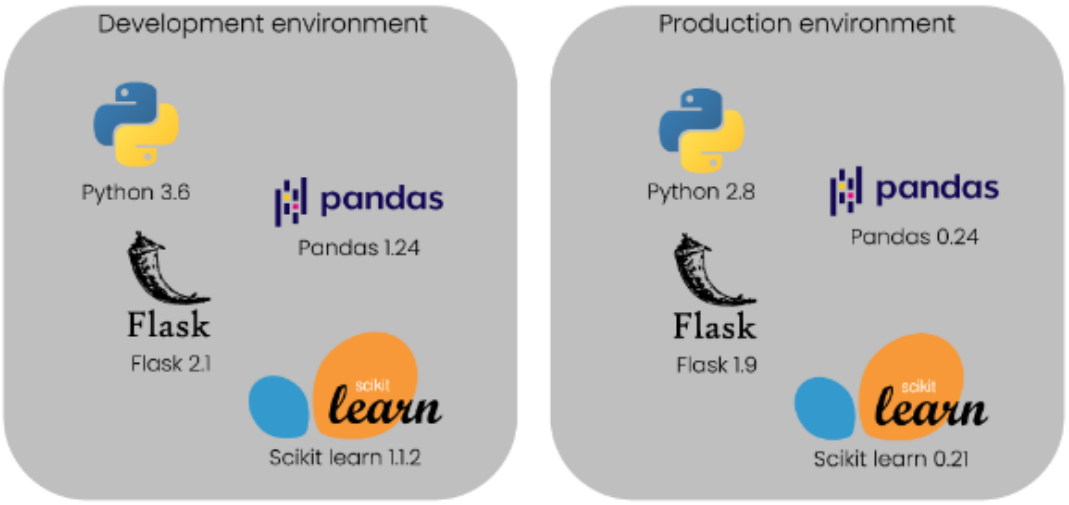


Preparing model for deployment

Development to deployment

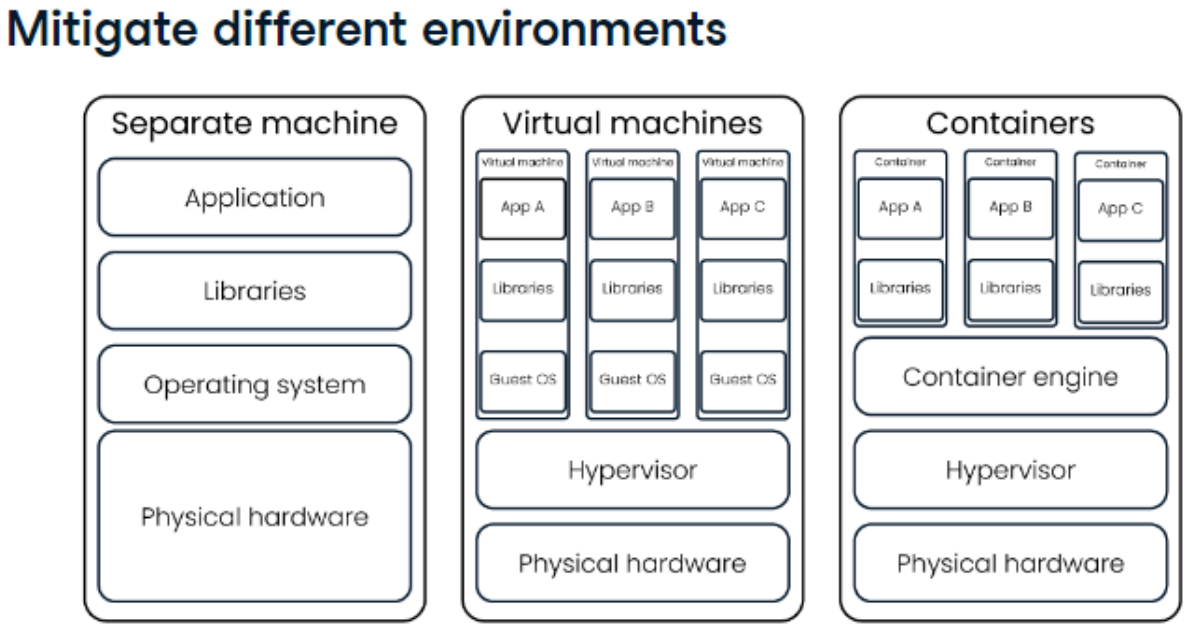


Runtime environments



<different version in two other environment>

Mitigate different environments



separate machine

- Using separate but identically set up machines can mitigate environment differences.

- Each machine includes hardware, OS, libraries, and the application (production environment and ML model).
- This solution is simple but difficult to maintain and not scalable.
- Every update requires updating the entire machine.

virtual machines

- Virtual machines (VMs) can be used on a single physical machine.
- Each VM is like a virtual version of a physical computer with its own OS, libraries, and application.
- A hypervisor manages resource distribution across VMs.
- Easier to maintain than physical machines but resource-intensive, as each application needs its own VM.

Containers

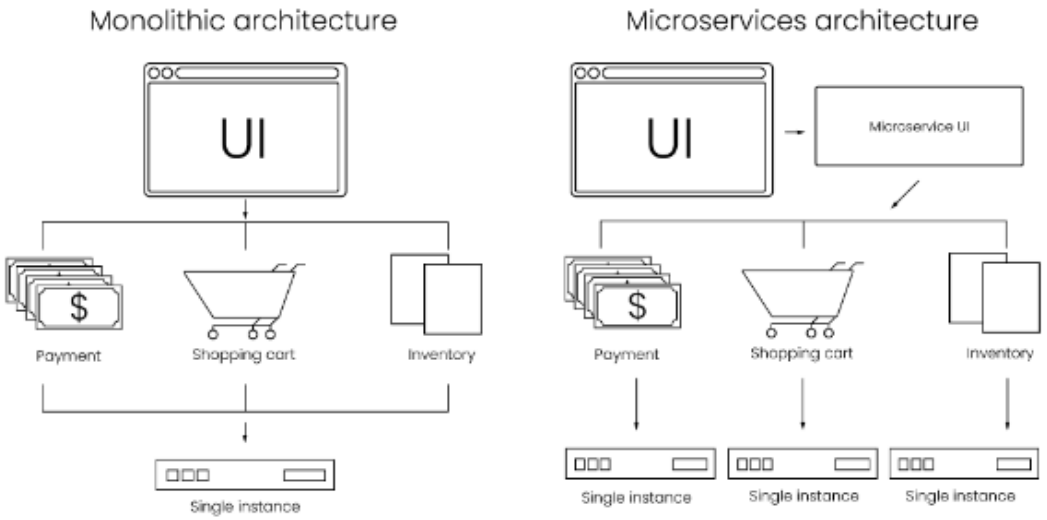
- Containers allow running multiple applications on one machine.
- Containers use fewer resources than virtual machines and are more portable.
- They are a lightweight alternative to virtual machines.
- Deploying ML models as containers is the current standard in MLOps.

Benefits of using containers

- Containers offer several benefits: easier maintenance, high portability, and fast startup.
- They are built once and can run anywhere.
- Containers only contain the necessary application, not a hypervisor or virtual OS.
- However, not every application needs to be containerized.
- If an application works well on a virtual machine without environment issues, containers may not be necessary.

Microservices architecture

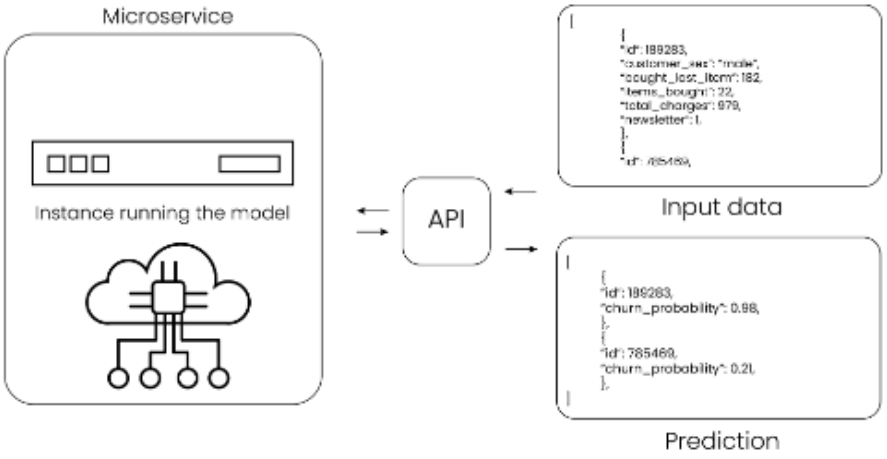
Monolith vs. microservice architecture



<적절히 세분화된 UI가 많은 service 운용에 도움을 준다.>

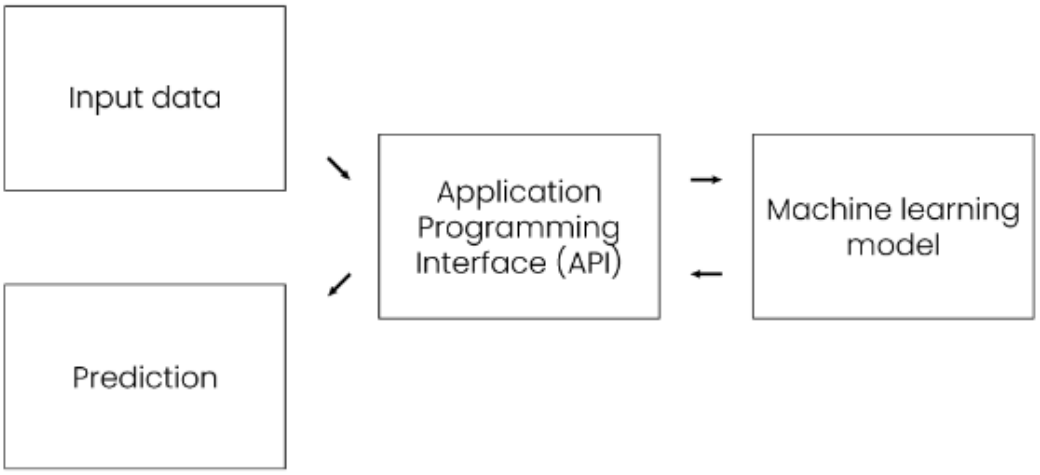
Inferencing

Inferencing is the process in which we send new input to the machine learning model and receive output from the model.



API

Application Programming Interface (API)



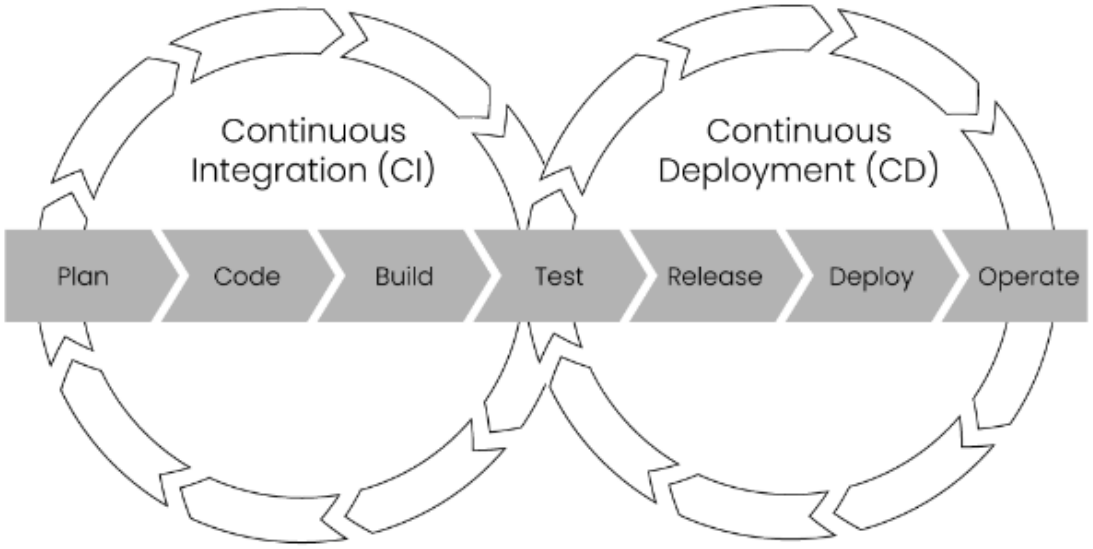
- An API enables communication between microservices.
- It defines input and output combinations for different services to interact.
- Comparable to a restaurant menu: input (order) goes to the kitchen (ML model), and the output (dish) is returned.
- Without an API, communication between services would be unclear, like not having a menu in a restaurant.

Integration

- After deploying the model as a microservice and setting up the API, the final step is integration into the business process.
- Integration typically involves connecting the API with the existing system.
- Before full production use, it's standard practice to test the model with a data sample to ensure everything functions as expected.

CI/CD and deployment strategy

CI/CD



- CI/CD is crucial in software development, automating code deployment.
- Originating from DevOps, it involves steps for developing, testing, and deploying code.
- A CI/CD pipeline allows for incremental changes to be easily pushed to production.
- These principles are also applied to developing and deploying machine learning models.

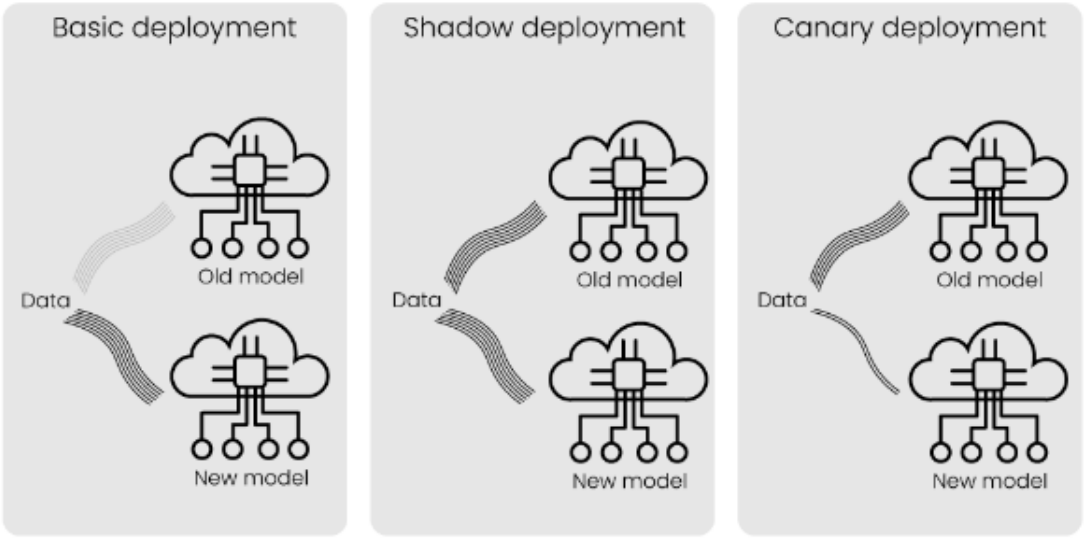
CI(continuous integration)

- CI involves integrating code changes quickly and frequently.
- Each code change is automatically tested upon commit and merge.
- CI helps identify errors and bugs early.
- It enables multiple developers to collaborate on the same code seamlessly.

CD(continuous deployment)

- CD automates the release of code validated during continuous integration (CI).
- The goal of CD is to always maintain production-ready code.
- CI and CD work together to streamline development and deployment processes.

Deployment strategies



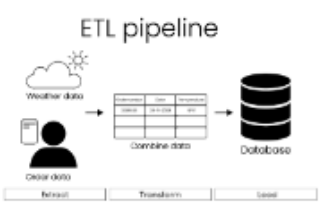
Deployment strategies

Strategy	Pros	Cons
Basic deployment	Straightforward, easy to implement, low resources	High risk if the model does not work as expected.
Shadow deployment	Easy to implement, no risk if model does not work as expected	Double resources.
Canary deployment	Slightly harder to implement, medium amount of resources	Small risk if model does not work as expected.

Automation and scaling

- The design phase is **crucial** in the ML lifecycle; without clear objectives and high-quality data, later phases may fail.
- ML is multidisciplinary, so alignment between all roles is key.
- The design phase remains largely manual, but can be templated for added value, business requirements, and key metrics.
- Templating makes the design phase more structured and aligns with MLOps practices.

Design phase



Project design

- Project design remains a manual process
- Use templates to automate and scale

Data acquisition

Development phase

Feature store

- Saves time building the same features
- Helps to scale

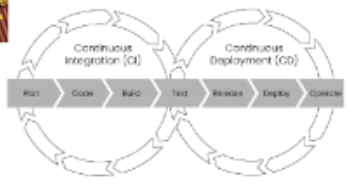
Experiment tracking

- Automates tracking
- Ensures reproducibility

Deployment phase

Containerization

CI/CD pipeline



Microservice architecture

Containerization

- Easy to start up copies of same application
- Improves scalability

CI/CD pipeline

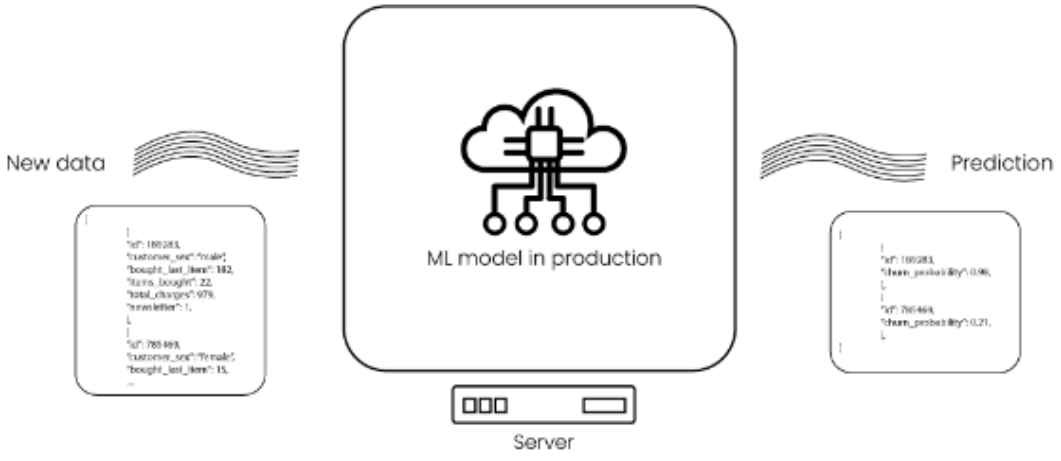
- Automates development and deployment
- Increases velocity of processes

Microservices architecture

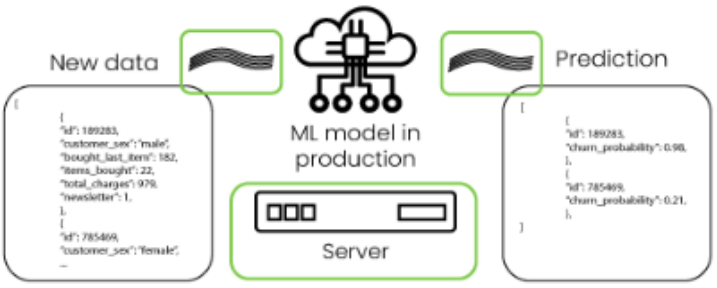
- Improves scalability
- Independent development and deployment

Monitoring machine learning models

Monitoring



Types of monitoring

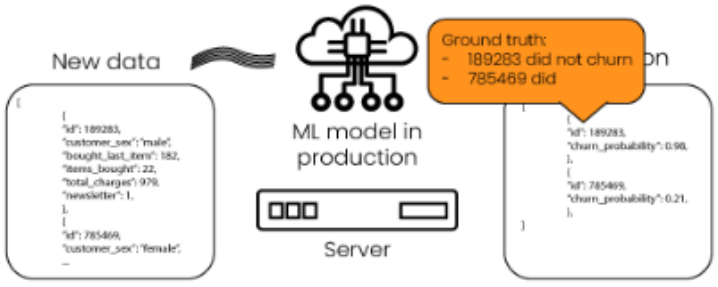


Computational monitoring: focuses on technical metrics

Examples: server CPU usage, number of incoming requests, number of predictions, downtime of server

- **Computational monitoring** is like checking if the kitchen appliances are running, gas and electricity are on, and people are working—focused on operational aspects, not the food.
- **Statistical monitoring** focuses on the kitchen's input and output: the quality of ingredients (data) and the taste of dishes (model predictions)—ensuring data quality and model performance.

Feedback loop

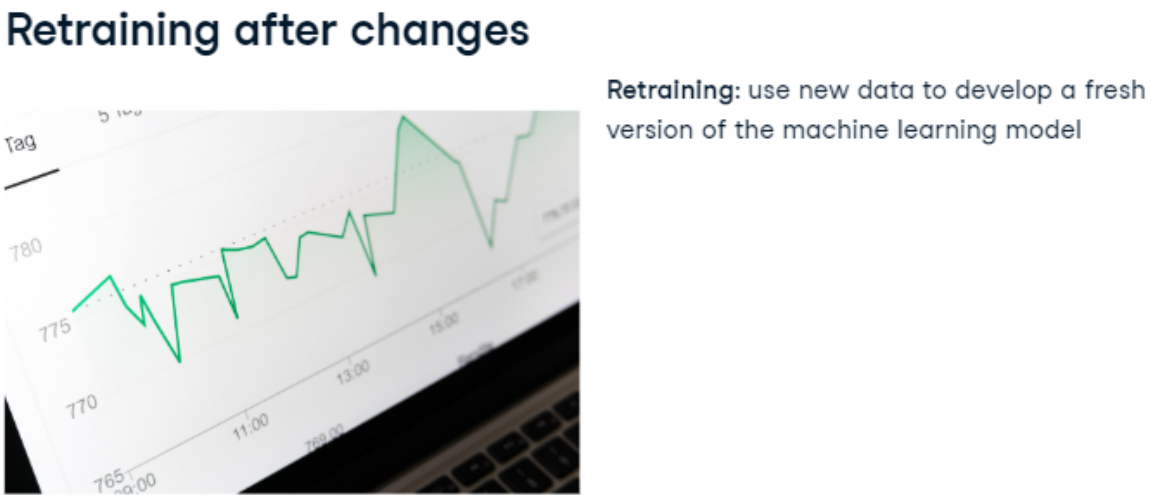


Feedback loop: the process through which the ground truth is used to improve the machine learning model

- The actual result is called the **ground truth**, used to evaluate model performance.
- Comparing model output to the ground truth forms a **feedback loop**.
- The feedback loop is essential for improving the model over time.

- It helps identify when and why the model was wrong, such as misclassifications for specific customer groups.

Retraining a machine learning model



Drift in data

- In ML problems, **input data** predicts the **target variable** (output data).
- For churn prediction, customer data is the input, and the target is whether they churn (0 = did not churn, 1 = churned).
- Two main changes can occur in the dataset: **data drift** and **concept drift**.
- **Data drift** occurs when there are changes in the input data over time.
- Examples include shifts in customer demographics, such as age or region.
- These changes can impact model performance, but not always, as data naturally evolves over time.
- **Concept drift** refers to changes in the relationship between input data and the target variable.
- It occurs when customer behavior changes, for example, when the same input data leads to no churn instead of churn.
- This shift in the input-output relationship can degrade model performance as the patterns the model was trained on no longer apply.

How often to retrain?

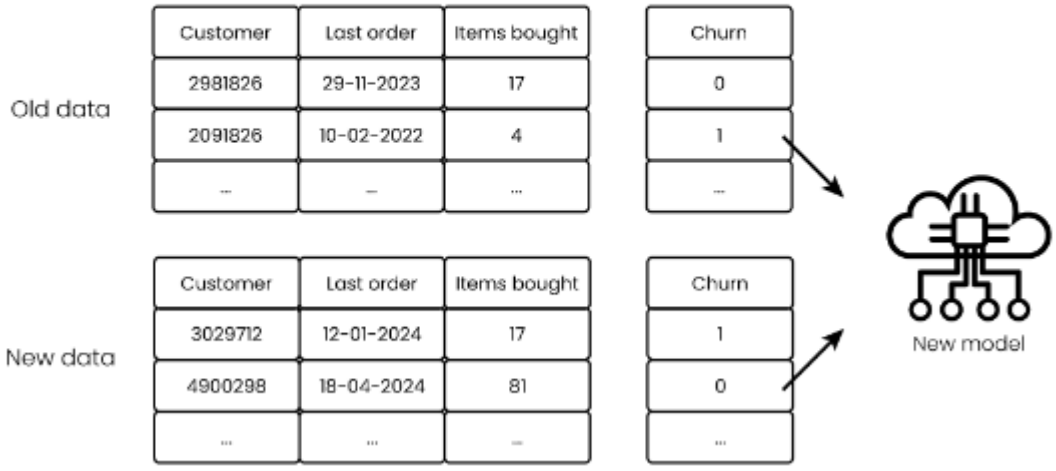
- Retraining frequency depends on several factors:

1. **Business environment**: Some environments change more frequently, and subject matter experts can help predict when changes might occur.
2. **Cost of retraining**: Retraining uses resources, and more complex models require more resources, thus increasing costs.
3. **Business requirements**: If a model must maintain a certain accuracy (e.g., >90%), retraining may be needed more often if accuracy drops below the threshold.

- The rate at which model accuracy declines is called **model degradation**.

Retraining methods

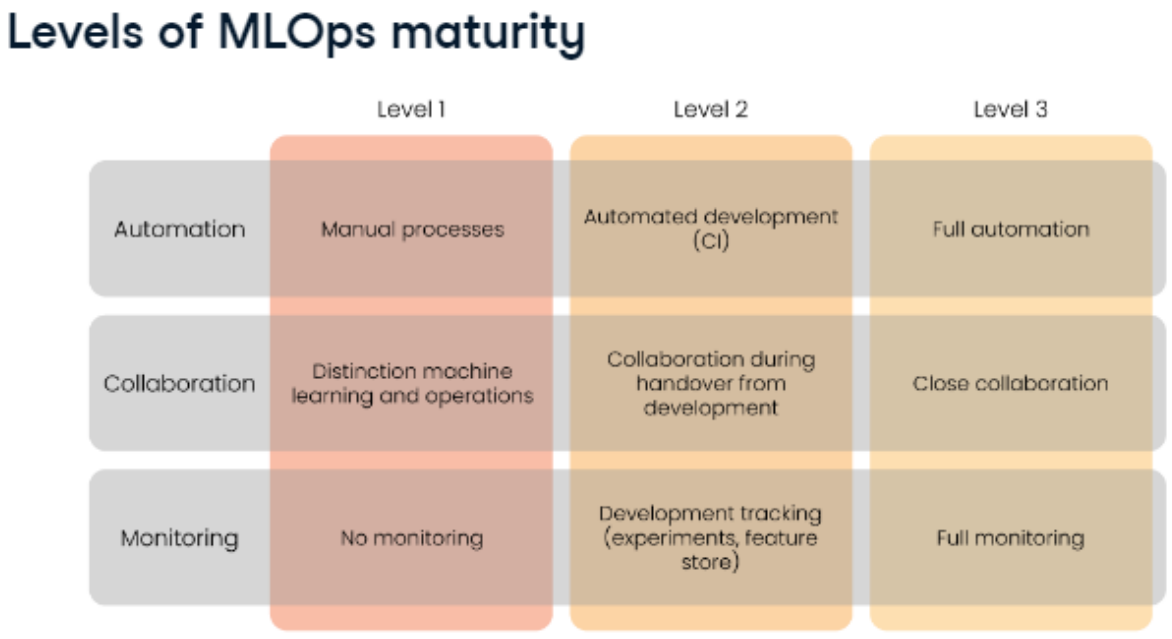
Retraining methods



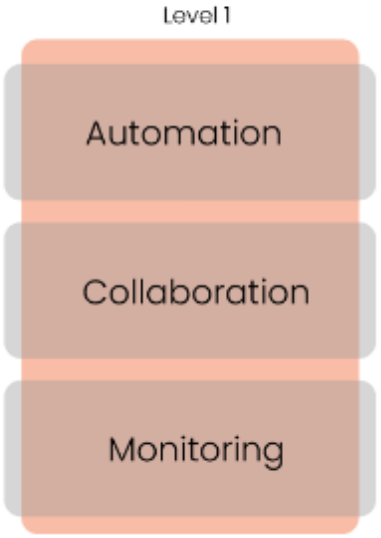
- Automatic retraining can be applied based on the company's ML maturity.
- Retraining can trigger when a certain amount of new data or **concept drift** is detected.
- For example, retraining could occur when detecting a change in the average age of customers.

Levels of MLOps maturity

- **MLOps maturity** reflects the automation, collaboration, and monitoring of ML and operations processes within a business.
- A higher level of MLOps maturity doesn't always mean it's better but indicates areas for potential improvement.
- MLOps maturity mainly applies to the **development** and **deployment** phases.
- The **design phase** requires human input, so it can't be fully automated, but templates can speed up and streamline the process.

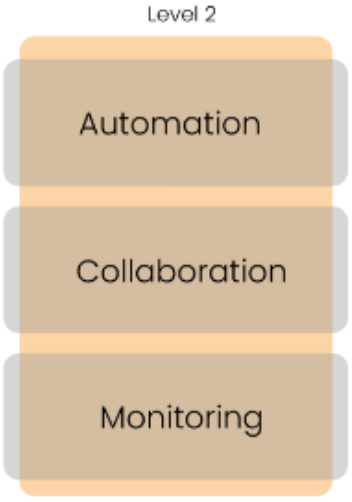


Level 1: Manual processes



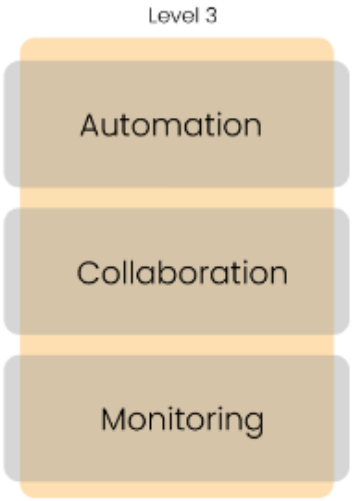
- Manual process for development
- Manual process for deployment
- No collaboration between ML and operations
- Teams work in isolation
- No tracking of development
- No monitoring after deployment

Level 2: Automated development



- Automated development pipeline (Continuous integration)
- Manual process for deployment
- After development teams will collaborate to deploy model
- Tracking of ML experiments and features
- Little monitoring after deployment

Level 3: Automated development and deployment



- Automated development pipeline (CI)
- Automated deployment pipeline (CD)
- Close collaboration between teams
- Monitoring of development and deployment
- Potentially automatically triggering retraining

MLOps tools

https://www.datacamp.com/blog/infographic-data-and-machine-learning-tools-landscape

Feature store

- Both open-source
- **Feast:** self-managed
- **Hopsworks:** part of larger platform

Experiment tracking

- **MLFlow and ClearML:** full machine learning lifecycle tools
- **Weights and Biases:** tracking and visualizing experiments

Containerization

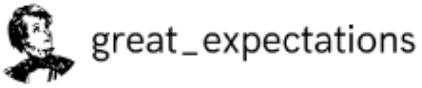
- **Docker:** containerizing applications
- **Kubernetes:** running containerized applications
- **Cloud providers:** provides Kubernetes-like services

CI/CD pipeline

- **Jenkins:** open-source continuous integration tool
- **GitLab:** code sharing and version control through repositories

Monitoring

- **Fiddler:** machine learning model monitoring
- **Great expectations:** data monitoring



MLOps platforms

Tools for full machine learning lifecycle

- AWS Sagemaker
- Azure Machine Learning
- Google Cloud AI platform