rapsit

WHITEPAPER

MLOps

What is it
How to get started

WHITEPAPER

Content

Introduction 3

MLOps lifecycle 8

MLOps platform requirements 14

Getting started with MLOps 17

Choosing a platform 21

Why Rappit 22

Further reading 23

Introduction

Machine Learning Operations, better known as MLOps, is the discipline of delivering machine learning (ML) models through repeatable and efficient workflows. It advocates formalizing and automating critical steps of Machine Learning system construction, deployment and optimization.

This whitepaper addresses the need for MLOps in organizations that embark on their Machine Learning journey. We will set the stage by focusing on the challenges of operationalizing Artificial Intelligence (AI). Next, we explain the need for standardized Machine Learning engineering practices, and we will provide an overview of continuous delivery and automation of Machine Learning. The final part of this paper provides guidance in getting started with MLOps in your organization.

The challenges of operationalizing AI

While AI and Machine Learning are considered as crucial pillars of digital transformation, *recent research* shows that only about 50% of organizations has moved beyond a proof of concept or pilot stages. *Successful deployments* and *effective operations* are the *biggest bottlenecks* for organizations to get value from AI.

Typical reasons for failed Enterprise AI Initiatives:

- Teams engaging in a high degree of manual and one-off work.
- Lack of reusable or reproducible components.
- Difficulties in handoffs between data scientists and IT.
- Challenges in deployment, scaling, and versioning.
- Lack of talent with domain expertise.
- Lack of change-management processes.
- Lack of strong governance models.
- Unrealistic expectations from leadership.

Need for best practices

Given the challenges in operationalizing AI, Machine Learning systems cannot be built in an ad hoc manner, isolated from other IT initiatives. Sound engineering principles need to be adopted and applied to make Machine Learning successful.

Best practices:

1. Preparing and maintaining high-quality data for training Machine Learning models.
2. Tracking models in production to detect performance degradation.
3. Performing ongoing experimentation of new data sources, Machine Learning algorithms, and hyperparameters, and then tracking these experiments.
4. Maintaining the veracity of models by continuously retraining them on fresh data.
5. Avoiding training-serving skews due to data inconsistencies and runtime dependencies between training environments and serving environments.

Organizations need an **automated & streamlined** Machine Learning process. This helps the organization to **successfully deploy** *Machine Learning models* in production. It also helps **manage risk** when organizations scale the number of Machine Learning use cases in changing environments ensure.

MLOps for best practice Machine Learning engineering

MLOps is the best practice methodology for Machine Learning engineering, which unifies Machine Learning system development with Machine Learning system operations. It advocates formalizing and automating critical steps of Machine Learning system construction.

MLOps provides a set of standardized processes and technology capabilities for building, deploying, and operationalizing Machine Learning systems rapidly and reliably.

MLOps practices can result in the following benefits:

- Shorter development cycles, shorter time to market.
- Better collaboration between teams.
- Increased reliability, performance, scalability, and security.
- Streamlined operational and governance processes.

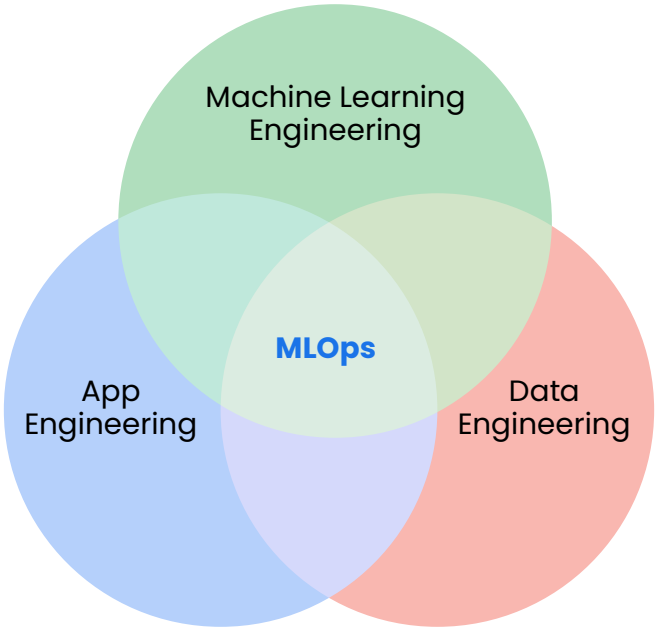
MLOps versus DevOps, DataOps

Machine Learning models typically do not operate in a vacuum. Their purpose is to generate insight and intelligence, often directly embedded into other applications. The process of building an ML enabled system usually involves three distinct focus areas:

1. **Data engineering** involves ingesting, integrating, curating, and refining data to facilitate a broad spectrum of operational tasks, data analytics tasks, and Machine Learning tasks. Data engineering can be crucial to the success of analytics and Machine Learning initiatives.

2. **Machine Learning** engineering involves building and deploying Machine Learning models in production using curated data usually created by the data engineering team.

3. **App engineering:** Machine Learning models do not operate in silos; they are typically components of one or more application systems. Integrating Machine Learning models into applications is a critical task in app engineering (DevOps).



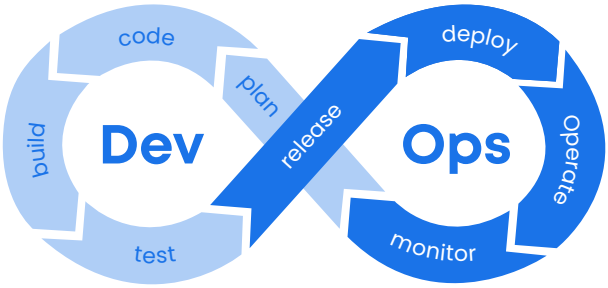
MLOps sits at the intersection of App, Data and Machine Learning engineering. It focuses on all aspects related to the development, deployment and operation of Machine Learning models.

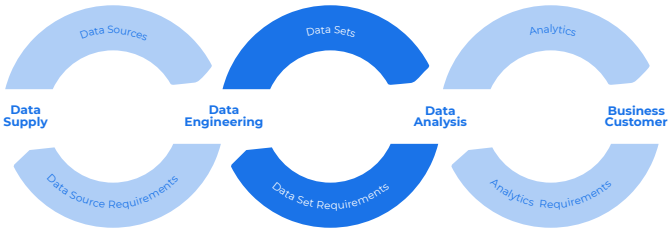
As a methodology for Machine Learning engineering, MLOps has many similarities to how DevOps supports application engineering and DataOps supports data engineering (analytics).

DevOps

DevOps combines philosophies, practices, and tools to deliver applications and services at high velocity.

The goal of DevOps is to automate and speed up the testing and deployment of code changes.





DataOps

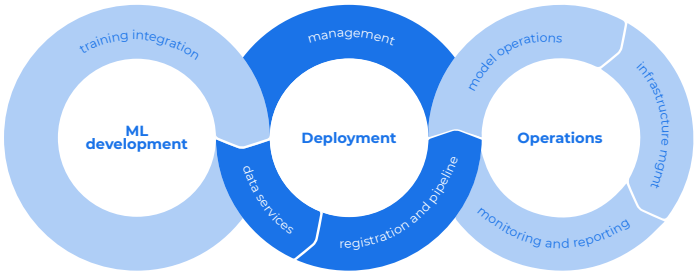
DataOps aims to reduce the life cycle time of data analytics while maintaining or improving its output quality.

The goal of DataOps is to ensure data is accessible and maintains integrity.

MLOps

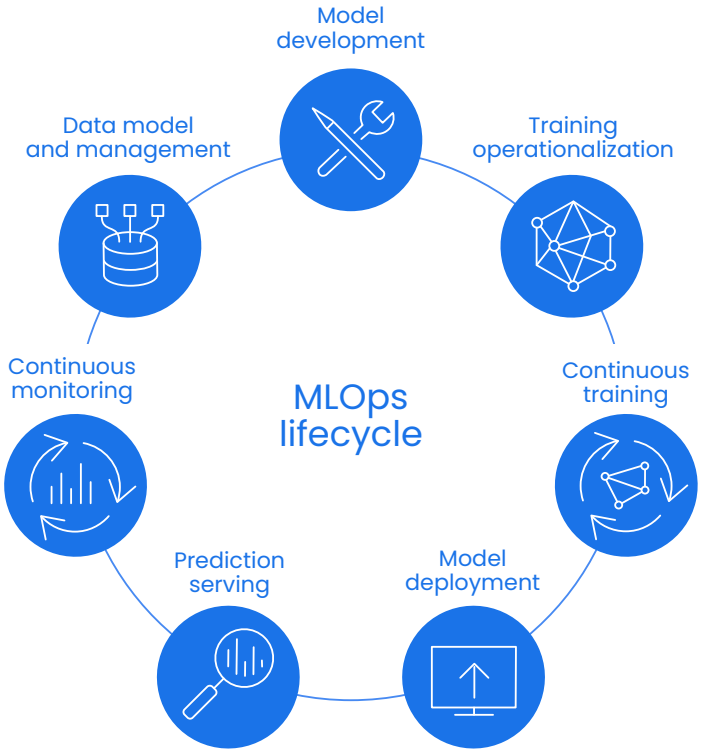
MLOps is the automation of critical steps in the deployment and monitoring of Machine Learning models, while satisfying both business and regulatory requirements.

The goal of MLOps is to provide a methodology for deploying and operationalizing Machine Learning models.



MLOps lifecycle

The MLOps lifecycle typically consists of 7 key, iterative stages. The overall success of implementing and operationalizing a Machine Learning system depends on each of these stages being completed successfully.



Stage 1: Model development

Experimentation is the core activity in the development of a Machine Learning model. It aims to arrive at an effective prototype model for the Machine Learning use case at hand. The primary source of development data is the *dataset and feature repository.* In addition to experimentation, data scientists need to *formalize their Machine Learning training procedures.* They do this by implementing an end-to-end pipeline, so that the procedures can be operationalized and run in production.

The key success aspects for this process are *experiment tracking, reproducibility, and collaboration.* Suppose there is a need to retrain models; in that case, the output of this process is not the model but the implementation of the continuous training pipeline to be deployed to the target environment.

Stage 2: Training operationalization

Training operationalization is the process of *building and testing a repeatable Machine Learning training pipeline* and then *deploying it to a target execution environment.*

For MLOps, Machine Learning engineers should be able to use configurations to deploy the Machine Learning pipelines. The configurations specify variables like the target deployment environment (development, test, staging, and so on), the data sources to access during execution in each environment, and the service account to use for running compute workloads.

A pipeline typically goes through a series of testing and staging environments before it is released to production.

The specifics of the pipeline deployment process depend on the technology used to implement the pipeline. Data scientists and Machine Learning engineers typically avoid the use of no-code solutions for pipeline implementation. Code-first technologies have more flexibility and control over the Machine Learning pipelines. Consequently, Machine Learning engineers can deploy the pipeline using standard CI/CD processes and tools.

Stage 3: Continuous training

The continuous training process is about orchestrating and automating the execution of training pipelines. These continuous training pipeline runs are based on a *retraining trigger.*

When the pipeline starts, it extracts a fresh training dataset from the data and feature repository, executes the steps in the Machine Learning workflow, and submits a trained model to the model registry. All the run information and artifacts produced throughout the pipeline run are *tracked in the metadata and artifact repository.*

An orchestrated and automated training pipeline mirrors the typical data science process that runs in the Machine Learning development phase. However, the automated pipeline has some differences. *Data validation* and *model validation* play a critical role in an automated pipeline as these steps are gatekeepers for the overall Machine Learning training process.

An important aspect of continuous training, therefore, is *tracking*. Pipeline runs must track generated metadata and artifacts in a way that enables debugging, reproducibility, and lineage analysis. When a pipeline run has finished and once the model has been validated, the pipeline can register a model candidate in the model registry.

Stage 4: Model deployment

After a model has been trained, validated, and added to the model registry, it is ready for deployment. The *model is packaged, tested, and deployed to a target serving environment* during the model deployment process..

When using a no-code/low-code platform, the model deployment process can be abstracted from the Machine Learning engineers. However, standard CI/CD routines can optionally be used when you want more control over the deployment process.

The model might need to go through a model governance process before being deployed to a target environment.

Stage 5: Prediction serving

After a model is deployed to its target environment, the model service starts to accept prediction requests by ingesting data and serving responses with predictions. Predictions are based on learning from the relationship between input (feature values) and output. This learning process is called inference, and can take the following forms:

- **Online inference** in near real time for high-frequency singleton requests using interfaces like REST or gRPC.
- **Streaming inference** in near real time, such as through an event-processing pipeline.
- **Offline batch inference** for bulk data scoring, usually integrated with extract, transform, load (ETL) processes.

In some scenarios of prediction serving, the serving engine might need to look up feature values related to the request. The serving engine in such cases fetches the values from feature repository and feeds them to the model to predict. The inference logs and other serving metrics are stored for continuous monitoring and analysis.

Stage 6: Continuous monitoring

Continuous monitoring is the process of monitoring the effectiveness and efficiency of a model in production, which is a crucial area of MLOps. Monitoring uses logs to identify anomalies, such as those caused by data drift and concept drift. Data drift is an unintended change in the input data stream, which could have any number of causes. Concept drift is caused by a change in the relationship between input and output data, typically caused by new external influencing factors.

It is essential to regularly and proactively verify that the model performance doesn't decay. Monitoring the process of model serving focuses on:

- **Model effectiveness:** by detecting data drift and concept drift
- **Resource utilization:** including CPUs, GPUs, and memory
- **Latency:** a key metric in online and streaming deployments to indicate model service health
- **Throughput:** a key metric in all deployments
- **Error rates**

Stage 7: Data and model management

One of the key challenges of data science and Machine Learning is creating, maintaining, and reusing high-quality data for training and analytics, across the organization. To mitigate this challenge:

- A **feature repository** helps to discover and reuse feature sets for entities, such as a product or a customer. It establishes a central definition of features and serves up-to-date values.
- **Dataset management** helps with maintaining scripts for creating datasets, reusing in various model implementations and environments and providing reproducibility.

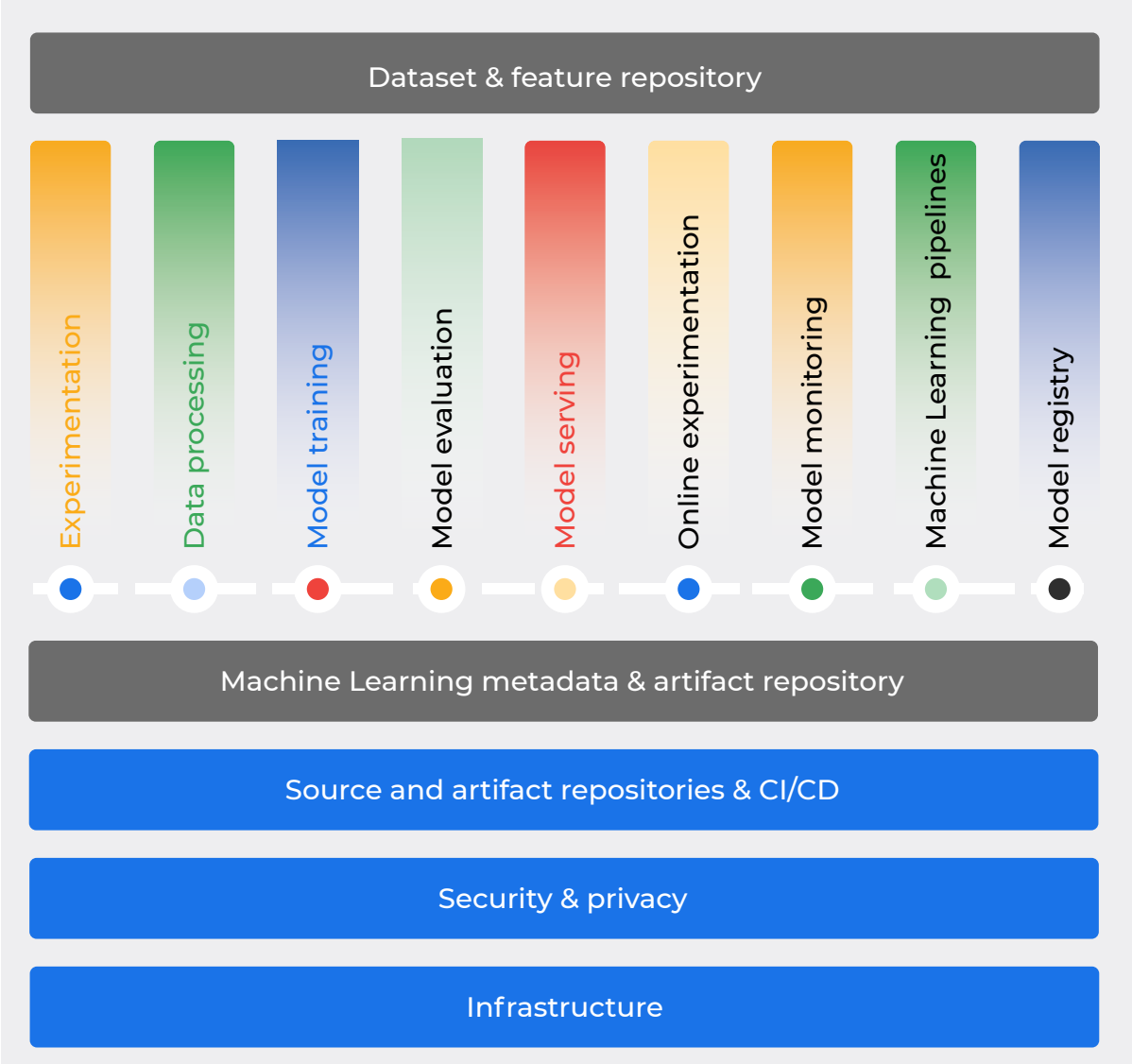
As organizations add to the number of models in production at scale, it becomes difficult to keep track of them manually. Organizations need controls to manage risk and implement Machine Learning models responsibility.

To help with this task, organizations need to establish robust model management, including Machine Learning metadata tracking and model governance.

- **Machine Learning metadata tracking** is generally integrated with different MLOps processes. The artifacts produced by the other processes are usually automatically stored in a Machine Learning artifact repository, along with the information about the process executions. Machine Learning metadata tracking enables data scientists and Machine Learning engineers to track experimentation parameters and pipeline configurations for reproducibility and lineage tracking. In addition, Machine Learning metadata tracking enables users to search, discover, and export existing Machine Learning models and artifacts.
- **Model governance** is about registering, reviewing, validating, and approving models for deployment. In addition, model governance should support reporting on the performance of deployed models.

MLOps platform requirements

The effective implementation of the key MLOps processes outlined in the previous chapter requires a set of core technical capabilities. A single integrated Machine Learning platform can provide these capabilities. Alternatively, they can be created by combining vendor tools best suited to particular tasks or created as a combination of these approaches. See the illustration below for an overview.



The technical capabilities generally required for MLOps can be categorized into three groups:

1. The first group covers foundational capabilities such as a reliable and secure infrastructure, source repositories, and CI/CD required to support any IT workload. Most organizations have already invested in these capabilities and can take advantage of them for Machine Learning workflows.

2. The second group is a set of core MLOps capabilities on top of the foundational capabilities. These include experimentation, data processing, model training, model evaluation, model serving, online experimentation, model monitoring, Machine Learning pipeline, and model registry.

Experimentation

Notebook environment integrated with Git, track experiments, analyze & visualize data and models, explore datasets and experiments.

Data processing

Provide a wide range of data connectors, efficient data transformations, feature engineering, scalable batch, and stream data processing.

Model training

Support common Machine Learning frameworks, large-scale distributed training, efficient hyperparameter tuning, and AutoML features.

Model evaluation

Perform batch scoring on evaluation datasets, visualize and compare the performance of different models, model behavior interpretation with explainable AI.

Model serving

Support for real-time and batch predictions, built-in support for common Machine Learning serving frameworks, logging prediction serving requests for analysis.

Online experimentation

Support canary deployments and A/B tests to understand the impact of a newly trained model. Results of experiments should be integrated with model registry.

Model monitoring

Measure model efficiency metrics like latency and serving-resource utilization, detect data skews including schema anomalies and data and concept shifts and drifts.

Machine Learning pipelines

Trigger pipelines on demand, on a schedule, or in response to specified events, integrate with the Machine Learning metadata tracking capability.

Model registry

Register, track, and version Machine Learning models, store model metadata and runtime dependencies, integrate with model evaluation capability.

3 Finally, two cross-cutting capabilities that enable integration and interaction are a Machine Learning metadata and artifact repository and a Machine Learning dataset and feature repository.

Machine Learning dataset and feature repository

Enable shareability, discoverability, reusability, and versioning of data assets, allow real-time ingestion and low-latency serving for event streaming and online prediction workloads, allow high-throughput batch ingestion and serving for ETL processes, model training, and for scoring workloads.

Machine Learning metadata and artifact repository

Provide traceability and lineage tracking of Machine Learning artifacts, share and track experimentation and pipeline parameter configurations, store, access, investigate, visualize, download, and archive Machine Learning artifacts. Integrate with all other MLOps capabilities.

Getting started with MLOps

Selecting capabilities for use cases

Establishing a mature MLOps practice to build and operationalize Machine Learning systems can take years to get right. As you start your MLOps journey, you may not need to implement all processes and capabilities. A Machine Learning model may have different use cases, such as:

- Pilot
- Mission-critical
- Reusable & collaborative
- Ad Hoc retraining
- Frequent retraining
- Frequent implementation updates
- Batch serving
- Online serving

Some will have a higher priority than others, depending on the type of workload and business value they create for you. The table below shows the recommended capabilities based on the characteristics of each use case.

Characteristics

Capabilities	Pilot	Mission critical	Reusable & collaborative	Ad hoc retraining	Frequent retraining	Frequent Implementation updates	Batch serving	Online serving
Experimentation	✓		✓			✓		
Data processing	✓			✓	✓		✓	
Model training				✓	✓			
Model evaluation		✓		✓	✓	✓		
Model serving							✓	✓
Machine Learning pipelines					✓		✓	
Model registry		✓	✓		✓	✓	✓	
Model monitoring		✓		✓				
Online experimentation		✓				✓		✓
CI/CD						✓		
Metadata tracking	✓	✓	✓		✓	✓		
Artifact tracking	✓	✓	✓		✓	✓		
Dataset repository		✓	✓					✓
Feature repository		✓	✓					✓

Mapping use cases to capabilities

Pilot use case

In a pilot use case, the focus is typically on data preparation, feature engineering, model prototyping, and validation for testing a proof of concept.

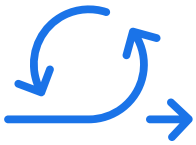
These tasks require experimentation and data processing capabilities. Data scientists want to set up experiments quickly and easily and track and compare them. They need the Machine Learning metadata and artifact tracking capability to debug, provide traceability and lineage, share and track experimentation configurations, and manage Machine Learning artifacts. For large-scale pilots, you might also require dedicated model training and evaluation capabilities.

Mission-critical use case

For mission-critical use cases, failure with the production model will have a significant negative impact on the business. Such use cases also need a robust model governance process to store, evaluate, check, release, report on models and protect against risks.

You can enable model governance by using the model registry and metadata and artifact tracking capabilities. The model evaluation capability is important to identify bias and fairness and provide explainability of the model. Monitoring is essential to assess the quality and production performance of the model. Online experimentation lets you test newly trained models against the one in production using a controlled environment before replacing the deployed model. Additionally, datasets and feature repositories provide you with high-quality data assets that are consistent and versioned.

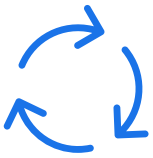
Reusable and collaborative use case



Reusable and collaborative assets allow your organization to share, discover, and reuse AI data, source code, and artifacts.

A *feature store* helps you standardize registering and storing processes, access features for training, and serve Machine Learning models. Once features are curated and stored, they can be discovered and reused by multiple data science teams. Having a feature store helps you avoid reengineering existing features and saves time on experimentation. Using Machine Learning metadata and artifacts tracking provides consistency, testability, security, and repeatability of the Machine Learning workflows.

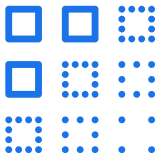
Frequent retraining use case



Use cases for frequent retraining are those where model performance relies on changes in the training data. The retraining might be based on time intervals (for example, daily or weekly), or it could be triggered based on when new training data becomes available.

For this scenario, you need Machine Learning pipelines to connect multiple steps like data extraction, pre-processing, and model training. You also need the model evaluation capability to ensure that the accuracy of the newly trained model meets your business requirements. As the number of models you train grows, both a model registry and metadata and artifact tracking help you keep track of the training jobs and model versions.

Batch serving use case



For batch predictions, there is no need to score in real-time. You pre- compute the scores and store them for later consumption, so latency is less of a concern than online serving.

However, because you process a large amount of data simultaneously, throughput is important Often batch serving is a step in a larger ETL workflow that extracts, pre-processes, scores, and stores data. Therefore, you need the data processing capability and Machine Learning pipelines for orchestration. In addition, a model registry can provide your batch serving process with the latest validated model to use for scoring.

Online serving use case

Online inference requires tooling and systems to meet latency requirements. The system often needs to retrieve features, perform inference, and return the results according to your serving configurations.

A feature repository lets you retrieve features in near real-time, and model serving allows you to deploy models as an endpoint easily. Additionally, online experiments help you test new models with a small sample of the serving traffic before you roll the model out to production.

Choosing a platform

Vertex AI

We see Google's Vertex AI as a powerfull and all-encompasing solution for managing all characteristics of the MLOps capabilities in one unified platform.
It offers a unified UI for the entire Machine Learning workflow, with end-to-end integration for data and AI. Some key elements are outlined below.

Vertex AI Feature Store: centralized repository for organizing, storing and serving Machine Learning features. Organizations can share, discover and reuse Machine Learning features at scale. Monitor feature serving with cloud monitoring.

Vertex AI Machine Learning Metadata: record the metadata and artifacts produced by the Machine Learning system and query to help analyze, debug, and audit the performance of your Machine Learning system.

Vertex AI Training & Prediction: train with AutoML or custom models and configure endpoint for prediction.

Vertex AI Pipeline: automate and monitor Machine Learning workflows (build, train & deploy).

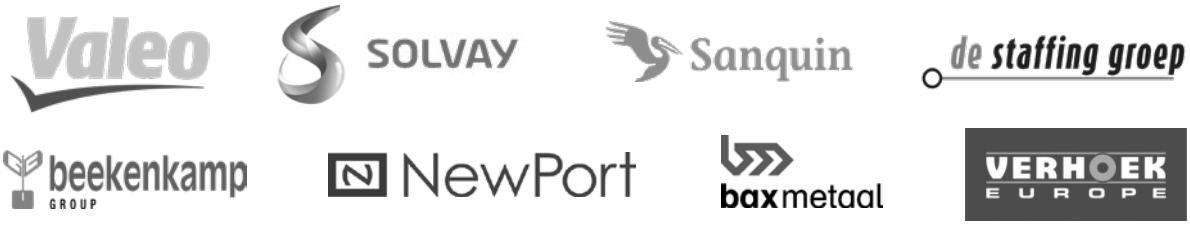
Vertex AI Model Monitoring: Vertex AI exports metrics to Cloud Monitoring. Monitor endpoints to understand model performance and
resource usage. Monitor feature store. Monitor training-serving skew and prediction drift on both categorical and numerical values.

Google's Data Catalog

In addition to Vertex AI, Google offers a fully managed and highly scalable data discovery and metadata management service with the following characteristics:

- A unified view of all data assets in the organization.
- Integration with BigQuery, PubSub and Cloud Storage and automatic tagging of data.
- Possibility to add technical and business tags
- Simplifies data discovery of scale using Google's search technology.

Selected customers:



Further reading

Practitioners Guide to MLOps
https://cloud.google.com/resources/mlops-whitepaper

Getting started with MLOps
https://cloud.google.com/blog/products/ai-machine-learning/select-the-right-mlops-capabilities-for-your-ml-use-case

MLOps: Continuous delivery and automation pipelines in Machine Learning
https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

Best practices for implementing Machine Learning on Google Cloud
https://cloud.google.com/architecture/ml-on-gcp-best-practices

Rappit is a leading independent software provider. Building on a proud heritage, we are experts in developing Cloud based Enterprise Software modernization solutions, that include Custom Made Applications, Data Analytics, Artificial Intelligence and Machine Learning. Based on Rappit, our application development suite consisting of amongst others Rappit Developer and Rappit Composer, we enable our customers to create unique business value.

Rappit serves a customer base consisting of multinationals and medium sized organizations, mainly in the Manufacturing, Supply Chain & Logistics and Retail & Wholesale industries. Our Rappit Suite also supports ISV's and System Integrators in building world-class applications.

Rappit headquarters
Vanenburgerallee 7
3882 RH Putten
The Netherlands

+ 31 (0) 341 74 59 99
info@rappit.io

Copyright © 2024 Vanenburg Software B.V. All rights reserved.

Copyright © 2024 Vanenburg Software B.V. All rights reserved. No part of this publication may be published, reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of Vanenburg Software