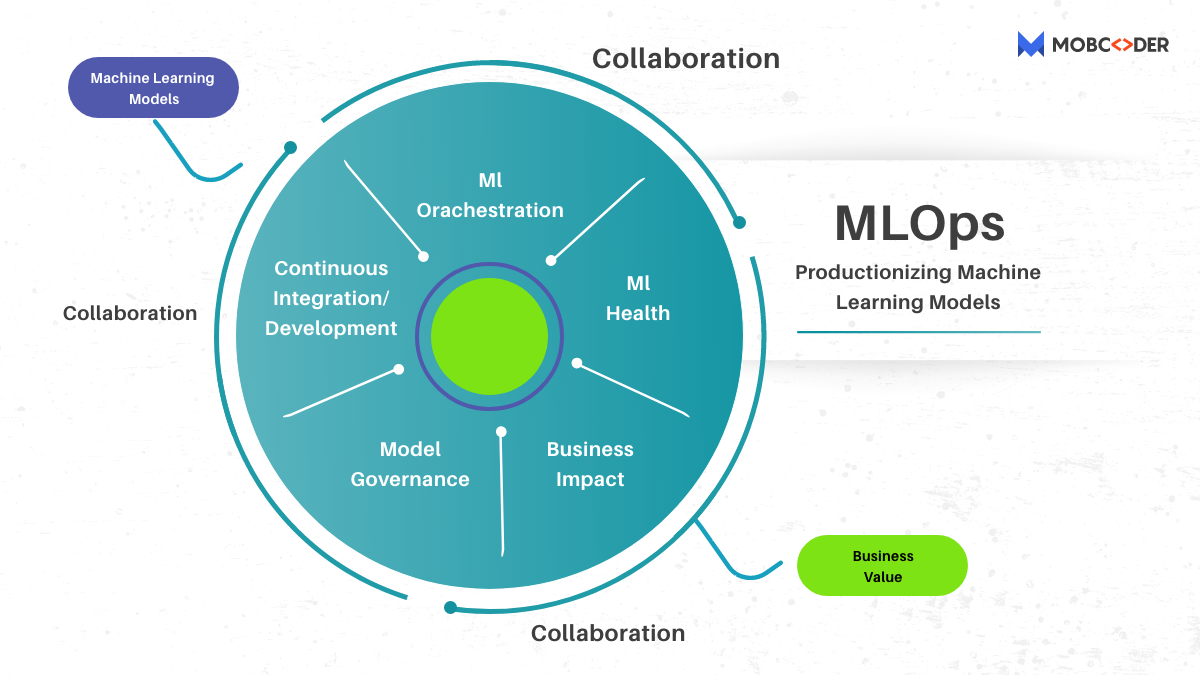
MLOps tools**help organizations apply DevOps practices to the process of creating and using AI and machine learning (ML) models**.

These tools are typically used by machine learning engineers, data scientists, and DevOps engineers.

**MLOps: Methods and Tools of DevOps for Machine Learning**



Machine Learning

During the first tech boom, Agile systems helped organizations operationalize the product lifecycle, paving the way for continuous innovation by clearing waste and automating processes for creation. DevOps further optimized the production lifecycle and introduced a new element, that of big data.

With more businesses now turning to machine learning insights, we’re on the cusp of another wave of operationalization. Welcome to MLOps.

When the term machine learning falls into the discussion, we tend to discuss data preparation or model building. Even most of the tech write-ups uncovering the technology fail to uncover its deployment part.

*According to a*[*recent study*](https://info.algorithmia.com/hubfs/2019/Whitepapers/The-State-of-Enterprise-ML-2020/Algorithmia_2020_State_of_Enterprise_ML.pdf)*, data scientists tend to dedicate up to 25 percent of their time to the infrastructure stage instead of focusing on deployment.*

The same study also put a heavy influence on the fact that putting a model from a research environment into production (where it eventually starts to add some business value) could take from 15 to 90 days on average.

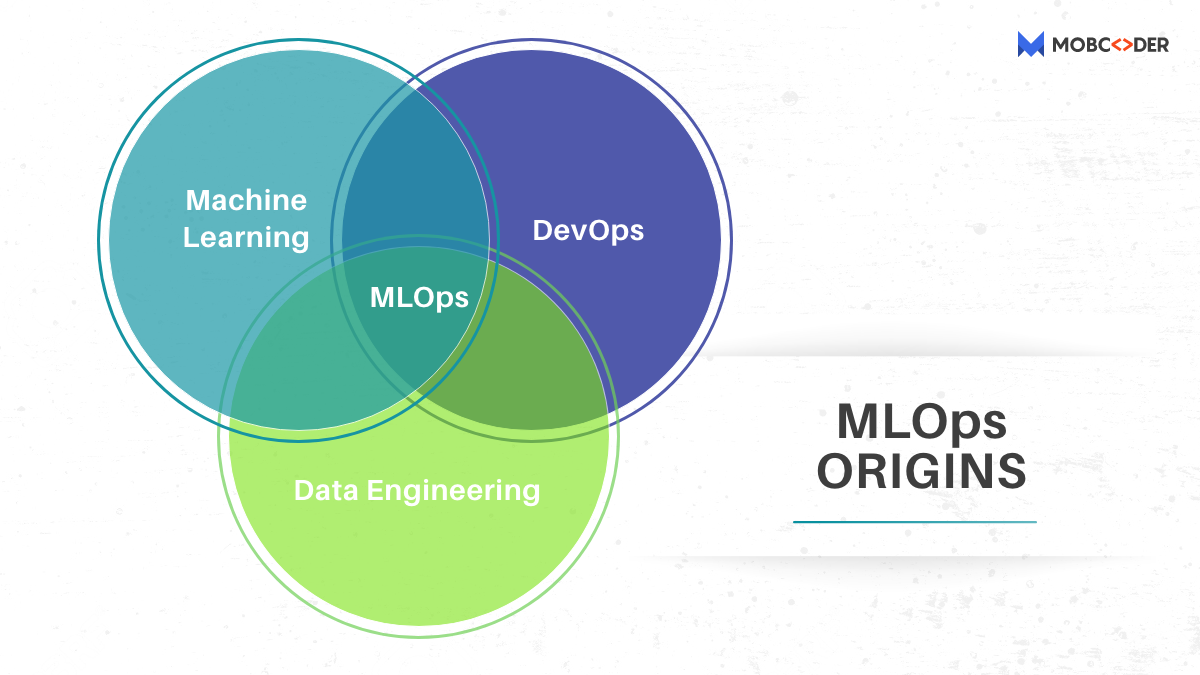
If we consider the worst-case scenario here, up to 75 percent of ML projects never got beyond the experimental phase.

As a logical reaction to this persistent problem, a new trend- ML Ops- has emerged. And as far as this document is concerned, this will incircle:

* Outlines the prime concepts and potential attributes of MLOps,
* explain how it is different from other popular Ops frameworks,
* acts as a guide through key MLOps phases, and
* Introduces all available tools and platforms to automate MLOps steps

Now without further ado, let’s get ahead to formalize the concept first and understand its meaning.

**What is MLOps and how it can drive & ensure business success?**



The term “MLOps” is a combination of machine learning and operations. It is a set of methods used to automate the lifecycle of machine learning algorithm in production—from initial model training to deployment to retraining against new data.

This methodology helps data science team and IT professionals to work together and combine their skills, techniques, and tools used in data engineering, machine learning, and DevOps — a predecessor of MLOps in the world of software development.

However, adoption of MLOps has numerous benefits but here are some most prominent as per enterprise standard:

More time for development for new models: “Untill now, data engineers or data scientists have been deploying models in production by themselves. But with the advent of MLOps, the production environment is the sole responsibility of professionals related to operations, while data scientists can solely focus on the core issues.

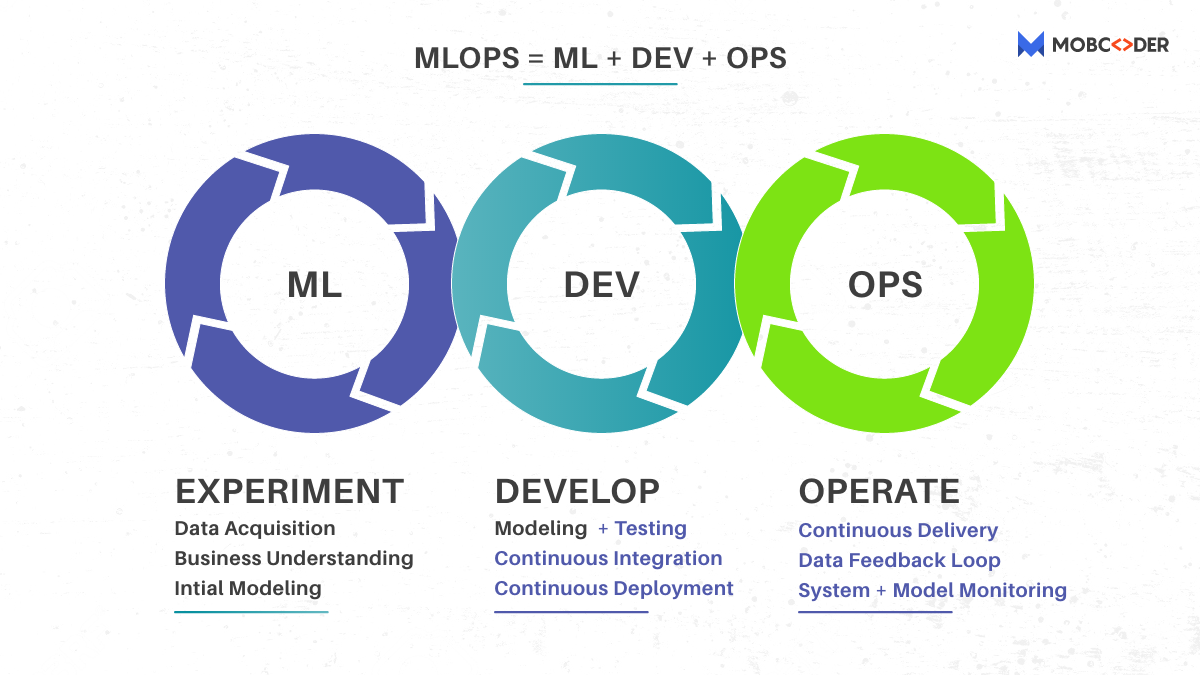
**Shorter time to market of ML models**

As devised for a purpose, MLOps brings automation to model training and retaining processes. It also establishes continuous integration & delivery (CI/CD) practices for deploying and update machine learning pipelines. As a result, ML-based solutions get into production faster.

Enhanced User Experience: Advanced MLOPs practices like continuous training and model monitoring ensures timely udates and improved customer satisfaction for the AI-powered apps.

Higher quality of predictions: MLOps is well designed methodology that takes care of data and model validation, evaluation of ts performance in production, and retraining against fresh datasets. These steps fend-off the risks of false insights and ensure reliable results produced by your algorithms in the course of decision making.

**MLOps and other Ops: what is the difference?**



The phenomenon called ‘Machine Learning’ defines the outline of MLOps that differentiate it from other “Ops” like DevOps, DataOps, and AIOps. So to avoid any further confusion, the further write-up will shed the light on core distinction between them.

**MLOps vs DevOps**

MLOps is frequently referred to as DevOps for machine learning. In a true sense, MLOps inherits a lot of principles from DevOps.

However, there are multiple similarities between DevOps and MLOps. But that doesn’t mean DevOps tools can apply to ML models to operationalize. These are the reasons behind this fact:

1. **Apart from the task of code versioning:** A dedicated place to save data and model version is quintessential as machine learning involves a lot of experimenting. Data scientists train models with various datasets, which leads to different outputs. That’s why in addition to code version control utllized in DevOps, MLOps requires specific instruments for saving data and model version to te reused.
2. **Models requires regular monitoring as they degrade over time unlike code**: Logically, after a trained model reaches production, it starts producing insignts from real time data.  However, there is no decline in the accuracy in the stable environment. But as live data changes frequently, predictive performance decreases over time. So to deliver real-time & accurate predictive analysis, we need continuous model monitoring which is not typical for DevOps practices.
3. **Upgradation is the key**: Upgradation never ends. Once any degradation in performance spotted, the model must be upgraded with the fresh data and validated before rolling out into the production again. As a result, in MLOps, continuous training and validation replace continuous testing, performed in DevOps.

**MLOps vs DataOps**

DataOps or Data operations came into existence almost simultaneously with MLOps, and inherit a lot of principles from DevOps. But the prime sphere of application is data analytics.

When it comes to DataOps, it encircles all the steps of the data lifecycle; right from collection to analyzing and reporting, and automates them where possible. It aims to improve the quality and reliability of data while mitigating the required time to deliver an analytical solution.

The approach is beneficial for organizations that work with large datasets and complex data pipelines. DataOps facilitate [ML projects](https://mobcoder.com/blog/3-ways-machine-learning-in-edtech-is-changing-the-education-industry/) at times, but only to a minor extent as it doesn’t have the capability to manage a model lifecycle. So it is better to consider MLOps a well-defined extension of DataOps.

**MLOps vs AIOps**

The youngest of all the above-mentioned Ops, AIOps is often used in place of MLOps, which is, put simply, quite incorrect.

*According to*[*Gartner,*](https://www.gartner.com/en/information-technology/glossary/aiops-artificial-intelligence-operations)*who coined the term in 2017, AIOps — or Artificial Intelligence for IT Operations — “combines big data and machine learning to automate IT operations processes.”*

In totality, the aim of AIOps is to automatically spot issues in day-to-day IT operations and proactively react to them using AI. Gartner expects that by 2023 up to 30 percent of large enterprises will adopt AIOps tools to monitor their IT systems.

Now without further ado, it is high-time to get back to our core purpose and explore the entire MLOps cycle in more detail.

**MLOps Concepts and Workflow**

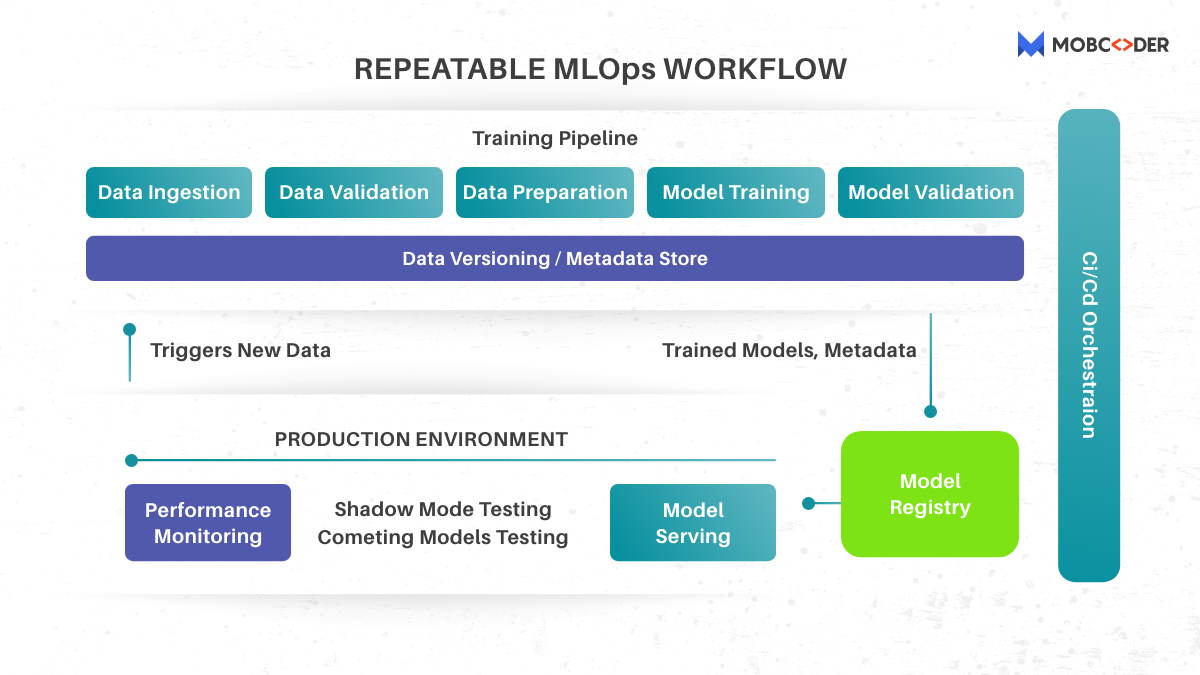
The end-to-end MLOps workflow is directed by continuous integration, delivery, and training methodologies that complement each other and pave the easiest way of AI solutions to customers.

**Continuous integration and continuous delivery (CI/CD**): MLOps follows a CI/CD framework advocated by DevOps as an optimal way to roll out quality code updates at regular or predefined intervals. However, machine learning expands the integration stage with data and model validation. Furthermore, it also addresses complexities of machine learning deployments. Currently, CI/CD brings all the components like data, model, and code components to release and refresh a predictive service.

**Continuous training (CT)**: A concept unique for MLOps, CT puts the automation part of the model retraining. It comprised of all the steps of model lifecycle from data ingestion to tracking its performance in production. The part ensures that your algorithm is updated with the first signs of decay or changes in the environment.

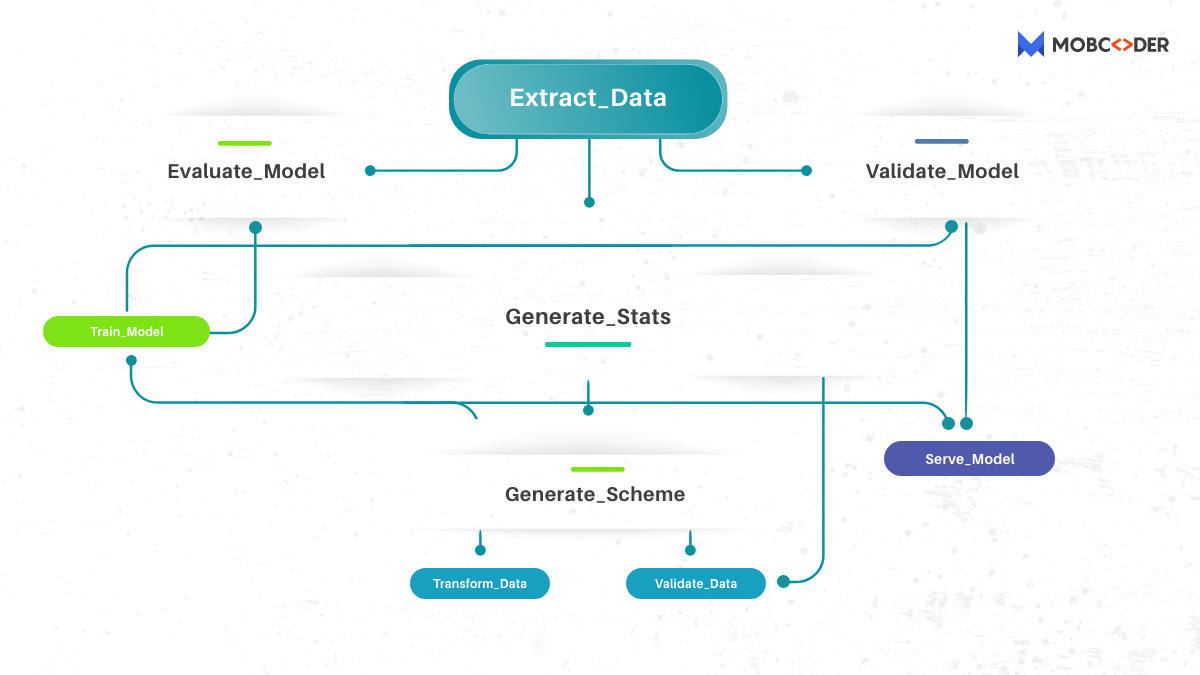
For the better understanding of how continuous integration, delivery, and training translate into practice and how the tasks are distributed between ML and operations specialists, let’s explore the key components of MLOps. This includes:

* Model training pipeline,
* Model registry,
* Model serving (deployment),
* Model monitoring, and
* CI/CD orchestration.



However, under some circumstances, these predefined steps and the entire workflow may vary and depend on the factors like — Project, Company size, Business tasks, Machine learning complexity, and many others. So, here we’ll describe the most common scenario and suggest available tools to automate repetitive tasks.

**Model Training Pipeline**



Here is the list of tools to build end-to-end model training pipelines**:** [TFX](https://www.tensorflow.org/tfx) (TensorFlow Extended), [MLflow](https://mlflow.org/), [Pachyderm](https://www.pachyderm.com/), [Kubeflow](https://www.kubeflow.org/)

**Who does what at this stage:**

* ML experts create training pipeline, engineer new features, monitor training process, fix problems
* Ops specialists test all the components of the pipeline and deploy them into a target environment.

If we talk about model training pipeline, it is a main component of the continuous training process and the entire MLOps workflow. It paves the way for frequent model training while data scientits can focus on developing new models for other business problems-rather than on fine tuning existing ones.

**Training cycle can be restarted, depends on the use case:**

* Manually,
* As per schedule (daily, weekly, monthly),
* Once the inflow of new data starts
* Once significant differences between training datasets and live datasets and live data are spotted
* Once model performance drops below the baselines

**These are the steps pipelines follows:**

**Data ingestion:** Every ML pipeline starts with data ingestion — to express in other words, acquiring new data from external repositories or feature stores, where data is saved as reusable “features”, designed for specific business cases. This step separates data into training and validation sets or combines different data streams into one, all-inclusive dataset.

**Data validation:**The aim of this step is to esnure that the ingested data meets all requirements. In case any anomaly is spotted, the pipeline can be automatically stopped until data engineers solve this issue. It also informs if your data changes over time, highlighting differences between training sets and live data your model uses in production.

**Data preparation:**Inthis step, raw data is cleaned up and gets mould into the right quality and format so your model can consume it easily. Meanwhile in this step, data scientists may come to combine raw data with domain knowledge and build some new innovative features, using capabilities of DataRobot, Featuretools, or other solutions for feature engineering.

**Model training.** At last, we come to the core of the entire pipeline. In the simplest scenario, the model is trained against freshly ingested and processed data or features. But you can launch several training runs in parallel or in sequence to identify the best parameters for a production model as well.

**Model validation.**This when we test the final model performance across the dataset it has never seen before to confirm its readiness for deployment.

**Data versioning.**This practice of saving data because similar to code versions in software development. The way to perform this is to use [DVC](https://dvc.org/), a lightweight CLI tool on top of GIT. You can also find similar functions in more complex solutions like MLflow or Pachyderm.

But why is this practice is quintessential for the entire MLOps lifecycle? In the course of training, model outputs differ significantly depending on the training dataset and parameters one opt for. Versioning tools store configurations used in a particular training run, which means you can reproduce the experiment with the same results wherever necessary. This helps you easily switch between datasets and models to look for the best combination.

**Model registry**

**Platforms serving as a model registry:** MLflow Model Registry, AI Hub

**Who does what at this stage:**ML specialists may share models and collaborate with Ops specialists to improve model management.

When an optimal option for production is found, it is pushed to a model registry — a centralized hub capturing all metadata for published models like:

* identifier,
* name,
* version,
* the date this version was added,
* the remote path to the serialized model,
* the model’s stage of deployment (development, production, archived, etc.),
* information on datasets used for training,
* runtime metrics,
* governance data for auditing goals in highly regulated industries (like healthcare or finance), and
* other additional metadata depending on the requirements of your system and business.

**Model Serving**

**Tools for model serving:**Seldon Core , MLflow Models, Algorithmia, Kubeflow

**Who does what at this stage:** Ops specialists control model deployment, while ML specialists may initiate testing in production.

**There are the three main ways to launch models in production:**

* on an IoT edge device,
* embedded in consumer application, and
* within a dedicated web service available via REST API or remote procedure call (RPC).

The latest approach called Model-as-a-Service is currently the most popular of all as it simplifies deployment, separating the machine learning part from software code. This means you can update a model version without redeployment of the application. Besides, in this case, the predictive service can be accessed by multiple consumer apps.

Tools like Kubeflow, TFX, or MLflow automatically package models as Docker images to be deployed on Kubernetes or special model servers like TensorFlow Serving and Clipper. What’s more, you can roll out more than one model for the same service to perform testing in production.

**Shadow mode testing.**Also called “dark launch” by Google, this method runs production traffic and live data through a fresh model version. Its results are not returned to end users, but only captured for further analysis. Meanwhile, the old version continues to produce predictions for customers.

**Testing competing models.**This technique involves simultaneous deployment of multiple models with similar outputs to find out which is better. The method is similar to A/B testing except that you can compare more than two models. This testing pattern brings additional complexity, as you must split the traffic between models and gather enough data for the final choice.

**Model monitoring**

**Tools for model monitoring:**MLWatcher, DbLue, Qualdo

**Who does what at this stage:** ML experts analyze metrics and capture system alerts.

Upon release to production, the model performance may be affected by numerous factors, from an initial mismatch in research and live data to changes in consumer behavior.

At some point, when the accuracy becomes unacceptable, the algorithm must be retrained. But it’s not easy to detect when exactly the so-called model drift happens. Usually, machine learning models don’t demonstrate mistakes immediately, but their predictions do affect the end results. Inaccurate insights can lead to bad business decisions and, consequently, financial losses. To avoid such troubles, companies should employ software solutions to automatically detect anomalies, make early alerts, and trigger machine learning pipelines for retraining.

**CI/CD Orchestration**

**Tools for streamlining CI/CD workflow**: GoCD, IBM Urban Code, AutoRABIT

**Who does what at this stage:** operations specialists unify multiple processes and automate the entire release process.

This allows you to tie together pipelines in a relatively simple manner. It’s worth noting that many MLOps solutions easily integrate with mainstream CI/CD tools as well as with Git.

**The future of MLOps**

Over the span of a few short years, MLOps has grown in unprecedented popularity and a number of Open Source frameworks have emerged. A move that signifies the importance of this practice, as [data and technology](https://mobcoder.com/)continue to expand and reach new heights, developing ML strong strategies now, will assist organizations of all kinds to manage and succeed in the future.