2022 • GIORGIA BERTACCHINI

MLOps

Machine Learning Model Operationalization Management

Part 1

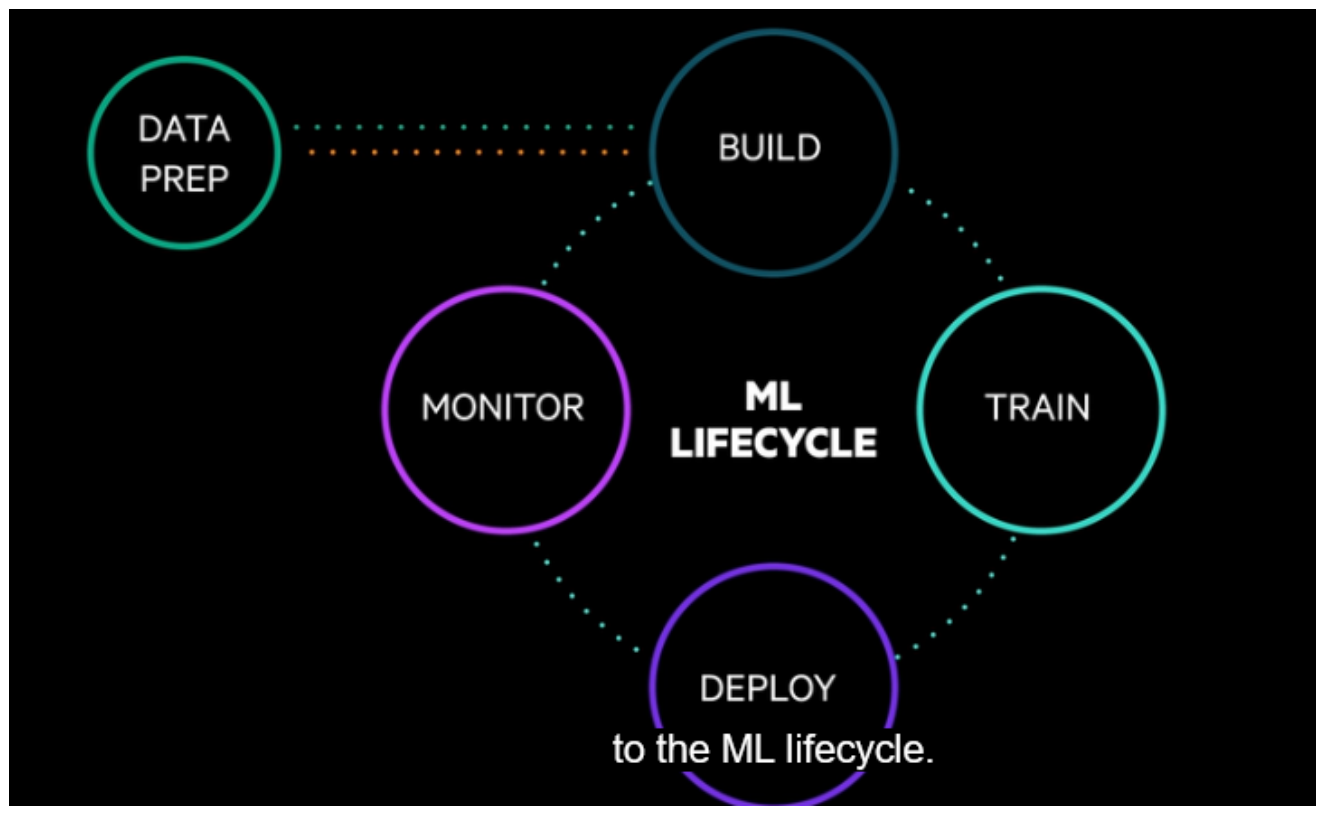
Introduction

MLOps • 2022

Enterprises and MLOps

ARE ENTERPRISES INTERESTED IN MLOPS?

Modern enterprises know that ML Ops will deliver DevOps-like speed and agility to the ML lifecycle.



Today's modern enterprises understand the benefits machine learning can provide and want to expand its use

However, because many enterprises lack standardized life cycle processes, many data science projects fail.
This in turn inhibits model deployment into current business processes and applications.

MLOps increase the ability to successfully deploy new data science models. Incorporating these standardized processes and tools eliminate the need to deploy infrastructure, allowing data scientists and data engineers to focus on model development.

MLOps goals

WHY ARE ENTERPRISES INTERESTED IN MLOPS?

According to a report by Algorithmia “2020 State of Enterprise Machine Learning”, many companies haven’t figured out how to achieve their ML/AI goals. Because bridging the gap between ML model building and practical deployments is still a challenging task.

Although AI budgets are on the rise, only 22 percent of companies that use machine learning have successfully deployed an ML model into production.

DevOps

MLOps is a DevOps extension that establishes effective practices and processes around designing, building, and deploying ML models into production.

WHAT IS DEVOPS?

- The DevOps movement started to coalesce some time between 2007 and 2008.
- DevOps is a combination of the term "development" and "operations". DevOps describes the adoption of iterative software development, automation, and programmable infrastructure deployment and maintenance.
- It is a methodology based on the principles of
    - Continuous Integration and Continuous Delivery or continuous deployment (CI/CD);
    - robust automation and trust between teams;
    - the idea of collaboration and increased communication between teams;
    - the end-to-end service life cycle (build, test, release);
- DevOps, which streamlines the practice of software changes and updates.

DevOps & MLOps

MLOps and DevOps are similar when it comes to continuous integration of source control, unit testing, integration testing, and continuous delivery of the software module or the package.

DIFFERENCES

However, in ML there are a few notable differences:

- **Continuous Integration (CI)** is no longer only about testing and validating code and components, but also testing and validating data, data schemas, and models.
- **Continuous Deployment (CD)** is no longer about a single software package or service, but a system (an ML training pipeline) that should automatically deploy another service (model prediction service) or roll back changes from a model.
- **Continuous Testing (CT)** is a new property, unique to ML systems, that's concerned with automatically retraining and serving the models.

DataOps

MLOps work together DataOps in the ML lifecycle.

WHAT IS DATAOPS?

DataOps is a term introduced in 2014 by IBM.

- DataOps involves processes and tools to ship high-quality data frequently which requires a combination of data engineering, data quality, data security and data integration.
- Used by data architects, data engineers, data analysts, and data scientists, this methodology powers data pipelines and machine learning models to help companies derive value from their data.

ModelOps

MLOps is ispired to ModelOps.

WHAT IS MODELOPS?

- MLOps and ModelOps emerged under these names particularly in late 2018 and 2019.
- Some argue that ModelOps is more general than MLOps, as it’s not only about machine learning models but any kind of model.
- The typical ModelOps process includes four different phases: build, manage, deploy and monitor.

AIOps

AIOps is sometimes confused with MLOps.

WHAT IS AIOPS?

- Artificial intelligence for IT operations.
- AIOps is another topic entirely and refers to the process of solving operational challenges through the use of artificial intelligence.
- It is the application of AI, and related technologies, such as machine learning and natural language processing (NLP) to traditional IT Ops activities and tasks.

a bit of MLOps

BASES

Machine Learning Operations (MLOps) defines language-, framework-, platform-, and infrastructure-agnostic practices to design, develop, and maintain machine learning applications.
MLOps should follow a “convention over configuration” implementation.

MLOps is as it combines the complexities of machine learning models with the complexities of the modern organization.

CAPABILITIES

The enterprises want MLOps for the following capabilities:

- MLOps enables **automated** testing of machine learning artifacts
- MLOps enables supporting machine learning models and datasets to build these models as first-class citizens within **CI/CD systems**.
- MLOps reduces **technical debt** across machine learning models.
- MLOps must be a **language-, framework-, platform-, and infrastructure-agnostic** practice.

a bit of MLOps

MLOPS AND ML

MLOps is designed to facilitate the installation of ML software in a production environment.
The term "machine learning" (ML) is the concepts apply to both artificial intelligence and data science fields.
Generally, the goal of a machine learning project is to build a statistical model by using collected data and applying machine learning algorithms to them.

Machine learning is the science of computer algorithms that automatically learn and improve from experience rather than being explicitly programmed. The algorithms analyze sample data, known as training data, to build a software model that can make predictions.

ML model uses what it has learned from previous data to make the best prediction it can based on the assumption that the unseen data is somehow related to the previous data.

Machine learning (deep learning) appears to be the most appropriate one because such problems have a large number of elements with different representations.

a bit of MLOps

MLOPS AND ML

There are six interactive phases in the ML development process:

- Business and Data Understanding
- Data Engineering
- Model Engineering
- Quality Assurance for ML Systems
- Deployment
- Monitoring and Maintenance

Machine Learning Engineering (MLE)

Instead of the term MLOps, we can use the definition of **Machine Learning Engineering (MLE)**.

MLE encompasses **all stages** from data collection, to model building, to make the model available for use by the product or the consumers. MLE is the use of **scientific principles, tools, and techniques** of machine learning and traditional software engineering to design and build complex computing systems.

The machine learning engineer has to understand how to work with data and be able to contribute to the full life cycle of a ML project.

In addition, the MLE often works with diverse stakeholders.

Where is MLOps?

ML software is used for:

- Image/speech/audio/pattern recognition
- Undertanding natural language and text generation
- Recommender
- Prediction of a numerical value, of a category for a data
- Planning and optimisation
- Supervised, Unsupervised and Reinforcement learning

By deploying the ML model, we can provide the following functionality:

- Recommendation, identifies the relevant product in a large collection
- Top-K Items Selection, which organizes a set of items in a particular order
- Classification, which assigns the input examples to one of the previously defined classes
- Prediction, which assigns some most probable value to an entity of interest
- Question Answering, which answers an explicit question
- Automation, which can be a set of user steps performed automatically,
- Fraud and Anomaly Detection, to identify an action or transaction being a fraud or suspicious
- Information Extraction and Annotation, to identify important information in a text

Scale

MLOPS FOR SCALE

MLOps is an essential component to massively deploying machine learning efforts.

Good MLOps practices:

- Keep track of versioning, especially with experiments in the design phase;
- Understand whether retrained models are better than the previous versions;
- Ensure that model performance is not degrading in production.

Three Pillars

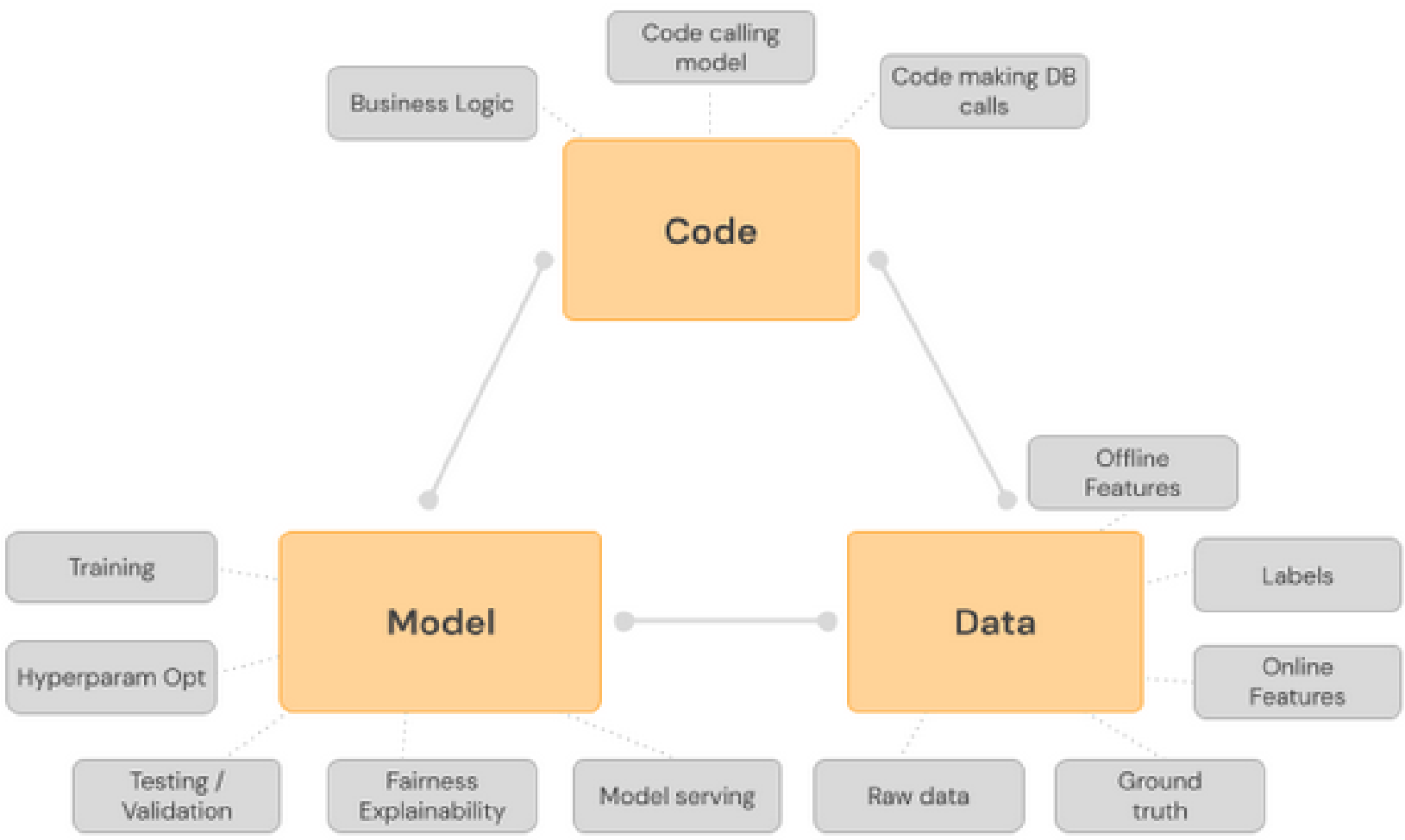
Three pillars

THREE PILLARS

MLOps is a process that helps organizations and business leaders generate long-term value and reduce risk associated with data science, machine learning, and AI initiatives.

MLOps is the result by **ModelOps**, **DataOps** and **DevOps**.
The pillars of an intelligent application are:

- **ML Model**: We might think of the ML model as a function that takes some input data and produces an output (classification, sentiment, recommendation, or clusters).
- **Data**: the raw input data, labels and features, as well as the new data to which the model is applied.
- **Code**: Code is the language an application uses to function. The code pillar might refer to business logic, calls to models, receiving outputs, decision making and calls to other data systems.



Data Engineering

DATA

It is the step to **acquire** and **prepare** the data to be analyzed.
Typically, data is being integrated from various resources and has different formats. Collecting good data sets has a huge impact on the quality and performance of the ML model.
Therefore, the data, which has been used for training of the ML model, indirectly influence the overall performance of the production system.

PREPARATION PHASE

The preparation phase is an intermediate phase aimed to prepare data for analysis, this phase is reported to be the most expensive with respect to **resources and time**.
The data preparation follows the data acquisition step, which is an iterative and agile process for exploring, combining, cleaning and transforming raw data into curated datasets.

FINAL GOAL & ISSUE

The final goal of these operations is to create **training and testing datasets** for the ML algorithms.

Issue: **data quality**, since ML models are built on data, they are sensitive to the semantics, amount and completeness of incoming data.

Data Engineering

PIPELINE

Stages of the data engineering pipeline: Data Ingestion, Exploration and Validation, Data Wrangling (Cleaning), and Data Splitting.
It is possible to automate the phases of the pipeline.

1. DATA INGESTION

Collecting data by using various frameworks and formats, such as internal/external databases, data marts, OLAP cubes, data warehouses, OLTP systems, Spark, HDFS, CSV, etc.
This step might also include synthetic data generation or data enrichment.

- Data Sources Identification: Data provenance
- Space Estimation: Check how much storage space it will take.
- Space Location: Create a workspace with enough storage space.
- Obtaining Data: Get the data and convert them to a format that can be easily manipulated without changing the data itself.
- Back up Data: Always work on a copy of the data.
- Privacy Compliance: Ensure sensitive information is deleted or protected (e.g., anonymized) to ensure GDPR compliance.
- Metadata Catalog: Recording the basic information about the size, format, aliases, last modified time, and access control lists.
- Test Data: Sample a test set, put it aside, if you are selecting a particular kind of ML model by using the test set.

Data Engineering

2. EXPLORATION AND VALIDATION

Includes **data profiling** to obtain information about the content and structure of the data.

Data validation operations are user-defined error detection functions, which scan the dataset in order to spot some errors.
The **validation** is a process of assessing the quality of the data by running dataset validation routines (error detection methods).

- Use RAD tools: Using Jupyter notebooks is a good way to keep records.
- Attribute Profiling: Obtain and document the metadata about each attribute: as Name, Number of Records, Data Type, Numerical Measures, Type of distribution.
- Label Attribute Identification.
- Data Visualization
- Attributes Correlation: Compute and analyze the correlations between attributes.
- Additional Data: Identify data that would be useful for building the model.

Data Engineering

3. DATA WRANGLING (CLEANING)

The process of re-formatting or re-structuring particular attributes and correcting errors in data.

Is recommend writing scripts or functions for all data transformations in the data pipeline to re-use all these functionalities.

- Transformations
- Outliers: Fix or remove outliers
- Missing Values
- Not relevant Data: Drop the attributes that provide no useful information
- Restructure Data

4. DATA SPLITTING

Splitting the data into training, validation, and test datasets to be used during the core machine learning stages to produce the ML model.

Model Engineering

CORE & ISSUE

The core of the ML workflow is the phase of writing and executing machine learning algorithms to obtain an ML model.

Issue: **model decay**, the performance of ML models in production degenerate over time because of changes in the real-life data that has not been seen during the model training.

PIPELINE

Stages of the model engineering pipeline: Model Training, Model Evaluation, Model Testing, and Model Packaging.
It is possible to automate the phases of the pipeline.

Model Engineering

1. MODEL TRAINING

The process of applying the machine learning algorithm on training data to train an ML model. It also includes feature engineering and the hyperparameter tuning for the model training activity.

- Feature engineering might include:
  - Discretize continuous features
  - Decompose features
  - Add transformations of features
  - Aggregate features into promising new features
  - Feature scaling: Standardize or normalize features
- Model Engineering include the following workflow:
  - Every ML model specification
  - Train many ML models from different categories
  - Measure and compare their performance.
  - Error Analysis
  - Consider further feature selection and engineering.
  - Hyperparameters tuning by using cross-validation. Please note that data transformation choices are also hyperparameters.

2. MODEL EVALUATION

Validating the trained model to ensure it meets original codified objectives before serving the ML model in production to the end-user.

Model Engineering

3. MODEL TESTING

Performing the final “Model Acceptance Test” by using the hold backtest dataset to estimate the generalization error.

4. MODEL PACKAGING

The process of exporting the final ML model into a specific format (e.g. PMML, PFA, or ONNX), which describes the model, in order to be consumed by the business application.

Model Deployment

BUSINESS APPLICATION

Once we trained a machine learning model, we need to deploy it as part of a business application.

OPERATIONS

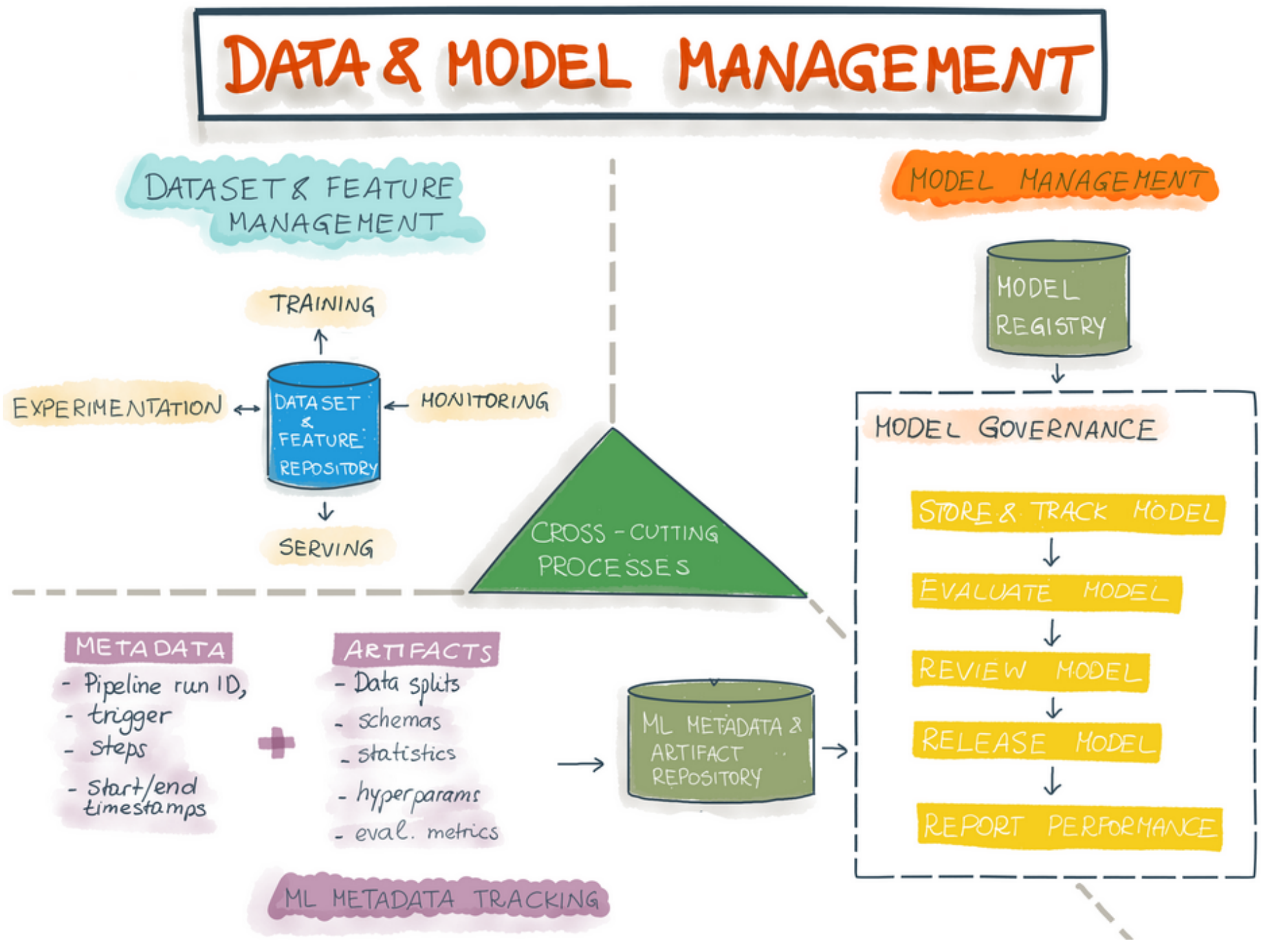
The final stage of the ML workflow is the integration of the previously engineered ML model into existing software.

This stage includes the following operations:

1. **Model Serving** - The process of addressing the ML model artifact in a production environment.
2. **Model Performance Monitoring** - The process of observing the ML model performance based on live and previously unseen data. In particular, we are interested in ML-specific signals, such as prediction deviation from previous model performance. These signals might be used as triggers for model re-training.
3. **Model Performance Logging** - Every inference request results in the log-record.

Data and Model Management

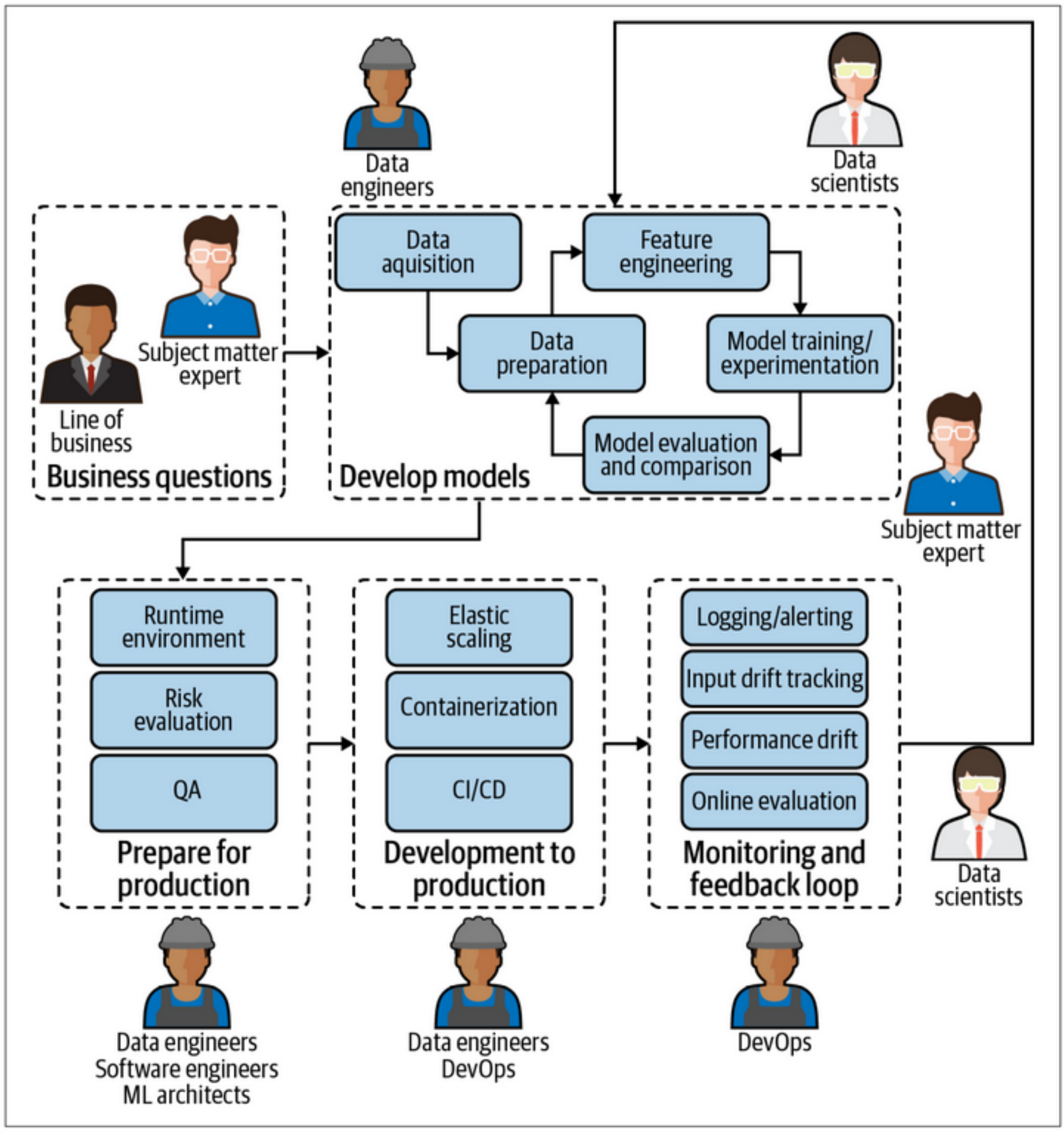
Data and model management are cross-purpose processes in the MLOps life cycle.



People

MLOps • 2022

People


Data engineers
Data scientists
Subject matter expert
Line of business
Business questions
Data aquisition
Feature engineering
Data preparation
Model training/ experimentation
Model evaluation and comparison
Develop models
Subject matter expert
Runtime environment
Risk evaluation
QA
Prepare for production
Elastic scaling
Containerization
CI/CD
Development to production
Logging/alerting
Input drift tracking
Performance drift
Online evaluation
Monitoring and feedback loop
Data scientists
Data engineers
Software engineers
ML architects
Data engineers
DevOps
DevOps

People: business questions

In this section we want to define the business needs, in the orientation of which the model is created.

Without this, other data professionals (particularly data scientists) risk starting the machine learning life cycle process trying to solve problems or provide solutions that don’t serve the larger business.

SUBJECT MATTER EXPERTS

Role?

- Provide business questions, goals, or KPIs around which ML models should be framed.
- Continually evaluate and ensure that model performance aligns with or resolves the initial need.
- That is, they need to understand not just model accuracy, precision, and recall, but the results or impact of the model on the business process identified up front.
- Have a role to play not only at the beginning of the ML model life cycle, but at the end (post-production) as well. Oftentimes, to understand if an ML model is performing well or as expected, data scientists need subject matter experts to close the feedback loop.

How MLOps?

- Mechanism or feedback loop for agging model results that don't align with business expectations.
- feedback mechanism and a platform for communication with data scientists about the models they are building.

People: develop models

Having the business goals now, we can develop the models. Everything starts from the Data acquisition and then carries out the following operations in cycle:

- Data preparation
- Feature engineering
- Model training/experimentation
- Model evaluation and comparison

Now we see other new people, but we remember the intervention of the subject matter experts at the closing of the feedback loop.

DATA SCIENTISTS

Role?

- Most are specialized in model building and assessment, and they are not necessarily experts in writing applications.
- Deliver operationalizable models so that they can be properly used in the production environment and with production data.

Operationalization?

- simply, operationalization of data science is the process of pushing models to production and measuring their performance against business goals. It's important also int the maintenance of those models—and the entire data pipeline—in production.

How in MLOps

- Automated model packaging and delivery for quick and easy deployment to production.
- Ability to develop tests to determine the quality of deployed models and to make continual improvements.
- Ability to investigate data pipelines of each model to make quick assessments and adjustments.
- Top-notch MLOps should allow data scientists the flexibility to select winning models from tests and easily deploy them.

People: develop models

DATA ENGINEERS

Role?

- Ability to investigate data pipelines of each model to make quick assessments and adjustments

How in MLOps?

- Ability to see the full details of individual data pipelines to address underlying data plumbing issues.
- Visibility into performance of all deployed models.
- optimize the retrieval and use of data to eventually power ML models.Generally, this means working closely with business teams, particularly subject matter experts, to identify the right data for the project at hand and possibly also prepare it for use.
- On the other end, they work closely with data scientists to resolve any data plumbing issues that might cause a model to behave undesirably in production.
- For maximum efficiency for the data engineer profile, MLOps must not consist of simple monitoring, but be a bridge to underlying systems for investigating and tweaking ML models.

People: prepare for production

After the Develop models phase, the outgoing model is ready and tested in a development environment but we don't know if it is also suitable for the production environment.
For the verification a runtime environment is created and there is also a study on the risk evaluation.
In this phase we see in particular two new figures, Software engineers and ML architects, but the role of Data engineers also remains important.

SOFTWARE ENGINEERS

Role?

- Integrate ML models in the company’s applications and systems.
- Ensure that ML models work seamlessly with other non-machine-learning-based applications.

How in MLOps?

- Versioning and automatic tests.
- The ability to work in parallel on the same application.
- it is necessary for example if the built ML model is not interpreted into a pre-existing non-ML software. Similarmy, software engineers are responsible for the maintenance of the non-ML software and a large part of that includes the functioning of the ML models in production.
- MLOps is a way for data scientists and software engineers to speak the same language and have the same baseline understanding of how different models deployed across the silos of the enterprise are working together in production.

People: prepare for production

ML ARCHITECTS

Role?

- Ensure a scalable and flexible environment for ML model pipelines, from design to development and monitoring.
- Introduce new technologies when appropriate that improve ML model performance in production.

How in MLOps?

- High-level overview of models and their resources consumed. As they have a strategic, tactical role, they need an overview of the situation to identify bottlenecks and use that information to find long-term improvements.
- Ability to drill down into data pipelines to assess and adjust infrastructure needs.
- Today, they often have to be knowledgeable not only on the ins and outs of data storage and consumption, but on how ML models work in tandem.
- This role requires collaboration across the enterprise, from data scientists and engineers to DevOps and software engineers. Without a complete understanding of the needs of each of these people and teams, machine learning architects cannot properly allocate resources to ensure optimal performance of ML models in production.

People: prepare for production

MODEL RISK MANAGER/AUDITOR

Model risk management(MRM)

Role?

- Minimize overall risk to the company as a result of ML models in production.
- Ensure compliance with internal and external requirements before pushing ML models to production.

How in MLOps?

- Robust, likely automated, reporting tools on all models (currently or ever in production), including data lineage.
- MRM can protect companies in any industry from catastrophic loss introduced by poorly performing ML models.
- MRM play the critical role of analyzing not just model outcomes, but the initial goal and business questions ML models seek to resolve to minimize overall risk to the company. They should be involved along with subject matter experts at the very beginning of the life cycle to ensure that an automated.
- As MRM and the teams with which they work often use different tools, standardization can offer a huge leg up in the speed at which auditing and risk management can occur.
- Automated reporting adds an extra layer of efficiency for MRM and audit teams in MLOps systems and processes.

People: development to production

Finally, we can push-to-production our model, through containerization and elastic scaling, adapting to the hardware and software present. All this must be done respecting the CI / CD pipeline, by the Data engineers and DevOps.

DEVOPS

Role?

- Conduct and build operational systems and test for security, performance, availability.
- Continuous Integration/Continuous Delivery (CI/CD) pipeline management.

How in MLOps?

- Seamless deployment pipeline.
- These roles require tight collaboration with data scientists, data engineers, and data architects.
- bridging the gap between traditional CI/CD and modern ML. That means systems that are fundamentally complementary and that allow DevOps teams to automate tests for ML

People: Monitoring and feedback loop

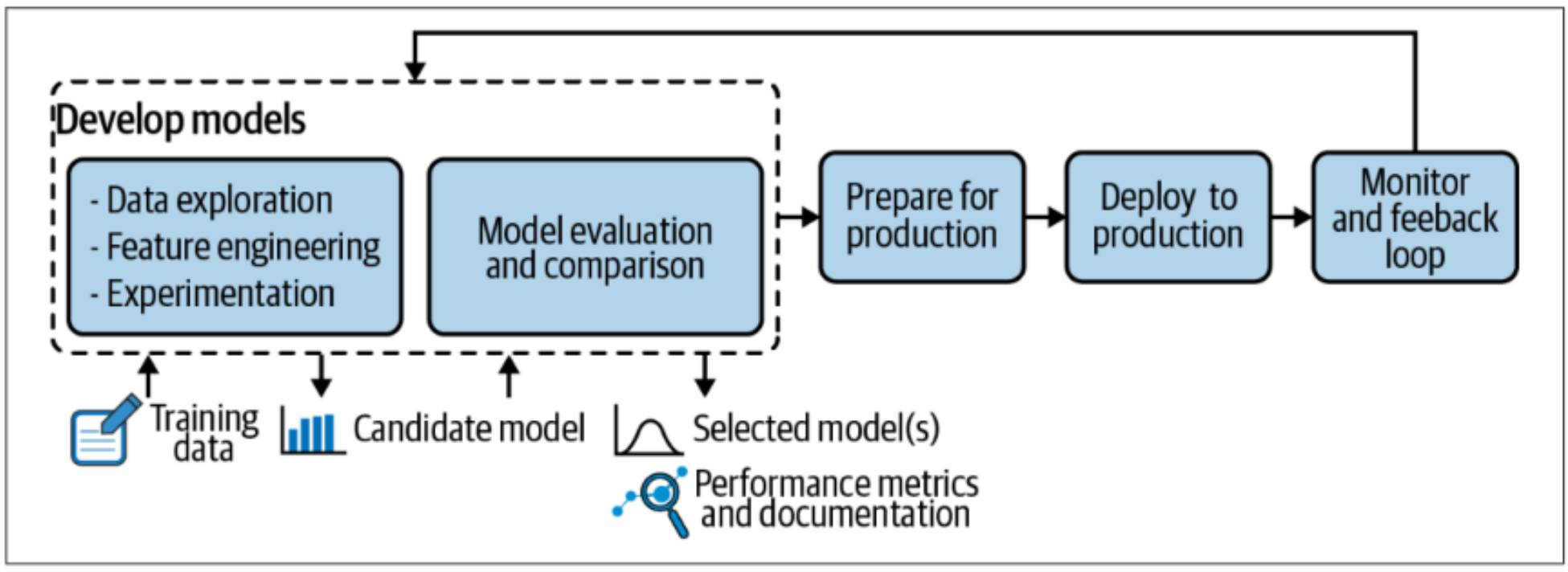
These steps just explained are execute in cycle. To these is added the important phase of monitoring.

The activities done here represent the main principles of DevOps, which want to check the cases of error to trace the cause and intervene if there are performance losses in the model. So: logging / alerting, Input drift tracking, performance drift and Online evaluation. The outgoing result from this is analyzed by Data scientists, who on the basis of updated data can think of developing a new model that is more efficient for current requests.

Developing Models

MLOps • 2022

Developing Models



What Is a Machine Learning Model?

ML MODEL

Machine learning models are based on statistical theory, and machine learning algorithms are the tools that build models from training data.

Their goal is to find a **synthetic representation of the data** they are fed, and this data represents the world as it was at the time of collection. Therefore, machine learning models can be used to make **predictions** when the future looks like the past, because their synthetic representation is still valid.

A model is the **set of parameters** necessary to rebuild and apply the formula. It is usually **stateless and deterministic**; the same inputs always give the same outputs.

What Is a Machine Learning Model?

REQUIRED COMPONENTS

Required components of a machine learning model:

- **Training data**: is usually labeled for the prediction case with examples of what is being modeled. It's important to have good training data.
- **A performance metric**: is what the model being developed seeks to optimize.
- **ML algorithm**: is important to note that some algorithms are more suited to certain tasks than others, but their selection also depends on what needs to be prioritized: performance, stability, interpretability, computation cost, etc.
- Hyperparameters: are **configurations** for ML algorithms. The algorithm contains the basic formula, the parameters it learns are the operations and operands that make up this formula for that particular prediction task, and the hyperparameters are the ways that the algorithm may go to find these parameters.
- **Evaluation dataset**: When using labeled data, an evaluation dataset that is different from the training set will be required to evaluate how the model performs on unseen data.

establishing business objectives

ESTABLISHING BUSINESS OBJECTIVES

- Main People: work of subject matter experts.
- Business objectives naturally come with performance targets, technical infrastructure requirements, and cost constraints;
- All of these factors can be captured as **key performance indicators**, or **KPIs**, which will ultimately enable the business performance of models in production to be monitored.

Data Exploration

Exploratory data analysis (EDA).
When data scientists or analysts consider data sources to train a model, they need to first get a grasp of what that data looks like.

Processes can include:

- Documenting how the data was **collected**
- Looking at **summarizing statistics** of the data
- Accurate **distribution** of the data
- **Cleaning, filling, reshaping, filtering, clipping, sampling**, etc.
- **Comparing** that data to other data

Feature Engineering and Selection

FEATURES

Feature engineering is the process of taking raw data from the selected datasets and transforming it into “features” that better represent the underlying problem to be solved.

Features are **how data is presented to a model**, serving to inform that model on things it may not infer by itself.

Examples of how features may be engineered:

- **Derivatives**: Infer new information from existing information
- **Enrichment**: Add new external information
- **Encoding**: Present the same information differently
- **Combination**: Link features together

Feature Engineering and Selection

FEATURE ENGINEERING TECHNIQUES

There are, however, many cases when information that data scientists need for their models is not available. In this case, there are techniques like impact coding, whereby data scientists replace a modality by the average value of the target for that modality, thus allowing the model to benefit from data in a similar range

Ultimately, most ML algorithms require a table of numbers as input, each row representing a sample, and all samples coming from the same dataset. When the input data is not tabular, data scientists can use other tricks to transform it.

Text or image inputs require more complex engineering. Deep learning has recently revolutionized this field by providing models that transform images and text into tables of numbers that are usable by ML algorithms. These tables are called **embeddings**, and they allow data scientists to perform transfer learning because they can be used in domains on which they were not trained.

TRANSFER LEARNING

Transfer learning is the technique of using information gained from solving one problem in solving a different problem. Transfer learning can be used to significantly accelerate learning of second or subsequent tasks, and it is very popular in deep learning, where the resources needed to train models can be enormous.

Feature Engineering and Selection

HOW FEATURE SELECTION IMPACTS MLOPS STRATEGY

Adding more features may produce a more accurate model, achieve more fairness when splitting into more precise groups, or compensate for some other useful missing information.
However, it also comes with downsides, all of which can have a significant impact on MLOps strategies down the line:

- The model can become more and more expensive to compute.
- More features require more inputs and more maintenance down the line.
- More features mean a loss of some stability.

Which inputs to use, how to encode them, how they interact or interfere with each other—such decisions require a certain understanding of the inner workings of the ML algorithm.
The good news is that some of this can be partly automated, using tools such as Auto-sklearn or AutoML applications that cross-reference features against a given target to estimate which features, derivatives, or combinations are likely to yield the best results, leaving out all the features that would probably not make that much of a difference.

Conclusion, when building models, the process of engineering and selecting features, like many other ML model components, is a delicate balance between considering MLOps components and performance.

Feature Engineering and Selection

FEATURE STORES

Feature factories, or feature stores, are repositories of different features associated with business entities that are created and stored in a central location for easier reuse.
They usually combine an offline part (slower, but potentially more powerful) and an online part (quicker and more useful for real-time needs), making sure they remain consistent with each other.

Experimentation

Experimentation takes place throughout the entire model development process, and usually every important decision or assumption comes with at least some experiment or previous research to justify it.

Goals of experimentation include:

- Assessing **how useful or how good** of a model can be built
- Finding the **best modeling parameters** (algorithms, hyperparameters, feature preprocessing, etc.).
- Tuning the b**ias/variance** trade-off for a given training cost to fit that definition of “best.”
- Finding a balance between **model improvement and improved computation costs**.

Data scientists need to be able to quickly iterate through all the possibilities for each of the model building blocks. Fortunately, there are **tools** to do all of this **semiautomatically**, where you only need to define what should be tested depending on prior knowledge and the constraints (e.g., computation, budget).

Trying all combinations of every possible hyperparameter, feature handling, etc., quickly becomes untraceable. Therefore, it is useful to define a time and/or computation budget for experiments as well as an acceptability threshold for usefulness of the model.

Experimentation

BIAS AND VARIANCE

A high **bias model** (also known as underfitting) fails to discover some of the rules that could have been learned from the training data, possibly because of reductive assumptions making the model overly simplistic.

A high **variance model** (or overfitting) sees patterns in noise and seeks to predict every single variation, resulting in a complex model that does not generalize well beyond its training data.

Evaluating and Comparing Models

CHOOSING EVALUATION METRICS

Choosing the proper metric by which to evaluate and compare different models for a given problem can lead to very different models. Unfortunately, there is no one-size-fits-all metric. You need to pick one that matches the problem at hand, which means understanding the limits and trade-offs of the metric (the **mathematics side**) and their impact on the optimization of the model (the **business side**).

To get an idea of how well a model will generalize, that metric should be evaluated on a part of the data that was not used for the model's training (a holdout dataset), a method called **cross-testing**. There can be multiple steps where some data is held for evaluation and the rest is used for training or optimizing, such as metric evaluation or hyperparameter optimization.
In addition, it allows one to assess whether the **data drifted** between the training and the holdout dataset.

Oftentimes, data scientists want to periodically retrain models with the same algorithms, hyperparameters, features, etc., but on more recent data. Naturally, the next step is to compare the two models and see how the new version fares.

Evaluating and Comparing Models

CROSS-CHECKING MODEL BEHAVIOR

Beyond the raw metrics, when evaluating a model, it’s critical to understand how it will behave. Depending on the impact of the model’s predictions, decisions, or classifications, a more or less deep understanding may be required.

Responsible AI

The DevOps team also needs to understand how to verify the model. Those required to document even more detail, including how the model was built and how it was tuned.

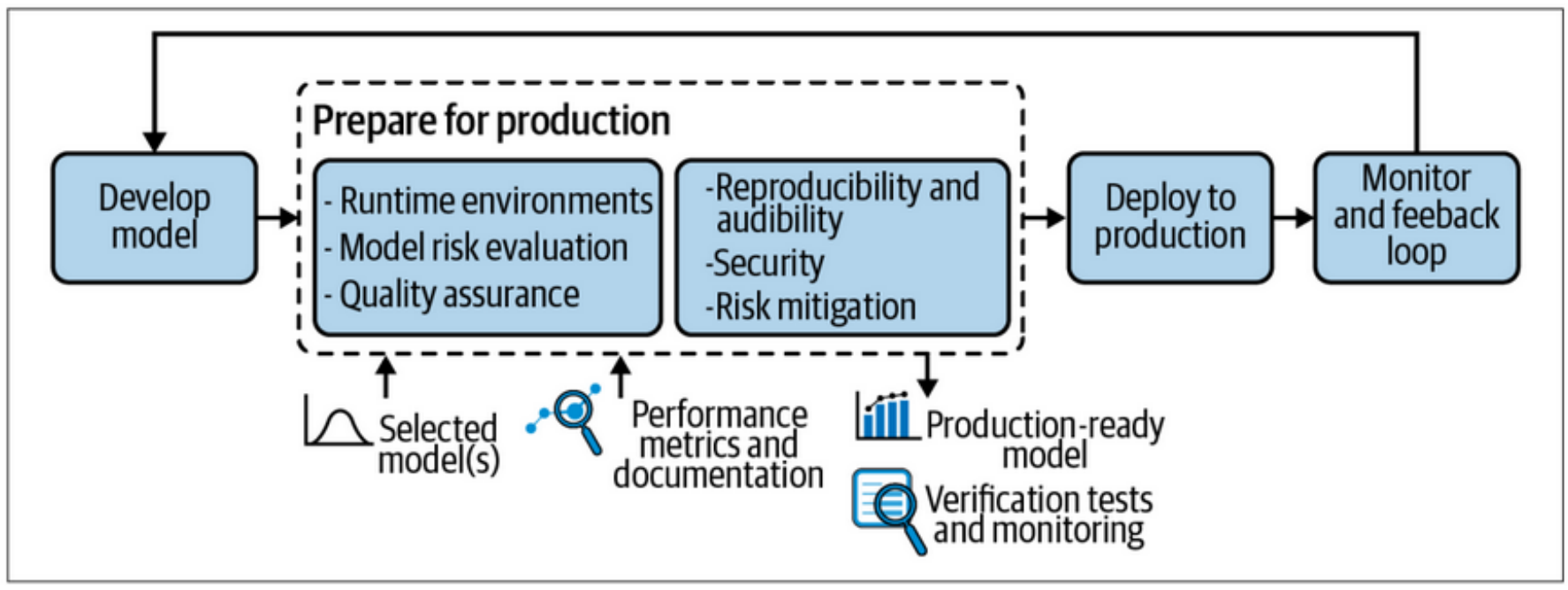
Automated model document generation, whereby a tool automatically creates documentation associated with any trained model.

Explain how the predictions are is important as a way to sanity-check the model or help better engineer features.
Explainability techniques are becoming increasingly important. They offer a way to mitigate uncertainty and help prevent unintended consequences.

Preparing for Production

MLOps • 2022

Preparing for Production



Not only is the **production environment** typically very different from the **development environment**, but the commercial risks associated with models in production are much greater.
It is important that the complexities of the transition to production are **understood and tested** and that the **potential risks** have been adequately mitigated.

MLOps includes having a clear validation step before sending models to production, and the key ideas to successfully prepare models for productions are:

- Clearly identifying the nature of the risks and their magnitudes;
- Understanding model complexity and its impact at multiple levels, including increased latency, increased memory and power consumption, lower ability to interpret inference in production, and a harder-to-control risk;
- Providing a simple but clear standard of quality;
- Automating all the validation that can be automated

Runtime Environments

Production environments take a wide variety of forms: custom-built services, data science platforms, dedicated services like TensorFlow Serving, low-level infrastructure like Kubernetes clusters, JVMs on embedded systems, etc. To make things even more complex, consider that in some organizations, multiple heterogeneous production environments coexist.

TWO CASES

When the development model can run without any modification in production, the technical steps required are reduced to a few clicks or commands, and all efforts can be focused on validation.

Instead, there are cases where the model needs to be reimplemented from scratch—possibly by another team, and possibly in another programming language. Given the resources and time required, there are few cases today where this approach makes sense.

Between these two extreme cases, there can be a number of transformations performed on the model or the interactions with its environment to make it compatible with production.

In all cases, it is crucial to perform validation in an environment that mimics production as closely as possible.

Runtime Environments

DATA ACCESS

Another technical aspect that needs to be addressed before validation and launch to production is data access.

In some cases, data can be frozen and bundled with the model. But when this is not possible, the production environment should **access a database** and thus have the appropriate network connectivity, libraries, or drivers required to communicate with the data storage installed, and authentication credentials stored in some form of production configuration.
When using external data access, model validation in situations that closely match production is even more critical as technical connectivity is a common source of production malfunction.

FINAL THOUGHTS

Training a model is usually the most impressive computation, requiring a high level of software sophistication, massive data volumes, and high-end machines with powerful GPUs.
This is because a model is trained once and can be used billions of times for inference.
Lowering the complexity of models or compressing extremely complex models can lower the infrastructure cost of operating machine learning models.

Model Risk Evaluation

Models attempt to mimic reality, but they are imperfect; their implementation can have bugs, as can the environment they are executing in.
Model Validation is essential to understand that involuntary malfunctions or malicious attacks are potentially threatening in most uses of machine learning in the enterprise.

For high-risk applications, it is essential that the whole team be fully aware of these risks so that they can design the validation process appropriately and apply the strictness and complexity appropriate for the magnitude of the risks.

THE ORIGINS OF ML MODEL RISK

ML model risk originates essentially from:

- Bugs, errors in designing, training, or evaluating the model;
- Bugs in the runtime framework, bugs in the model post-processing/conversion;
- Low quality of training data
- High difference between production data and training data
- Expected error rates, but with failures that have higher consequences than expected
- Adversarial attacks;
- Legal risk originating in particular from copyright infringement or liability for the model output;

These threats and how to mitigate them, which should ultimately be the goal of any MLOps system the organization puts in place.

Quality Assurance for ML

Software engineering has developed a mature set of tools and methodologies for **quality assurance** (QA).
Its purpose is to ensure compliance with processes as well as ML and computational performance requirements, with a level of detail that is proportionate to the level of risk.

Robust MLOps practices dictate that performing QA before sending to production is not only about technical validation. It is also the occasion to create documentation and validate the model against organizational guidelines.

Key Testing considerations

Obviously, model testing will consist of applying the model to carefully curated data and validating measurements against requirements.

Metrics must be collected on both statistical (accuracy, precision, recall, etc.) as well as computational (average latency, 95th latency percentile, etc.) aspects, and the test scenarios should fail if some assumptions on them are not verified.

Model stability is an important testing property to consider. When changing one feature slightly, one expects small changes in the outcome.

It is also important to be able to compare the model with its **previous version**.

Increasingly, failure to assess model fairness will have **business, regulatory, and reputational implications** for organizations.

Reproducibility and Auditability

REPRODUCIBILITY

In general, reproducibility in MLOps also involves the **ability to easily rerun the exact same experiment.** Data scientists may neet to have the ability to go back to different “branches” of the experiments—for example, restoring a previous state of a project.
It implies that the model comes with detailed documentation, the data used for training and testing, and with an artifact that bundles the implementation of the model plus the full specification of the environment it was run in.
Reproducibility is essential to prove model findings, but also to debug or build on a previous experiment. Every phase of either data processing, ML model training, and ML model deployment should produce identical results given the same input.

ML reproducibility must provide relevant metadata and information to reproduce models. **Model metadata management** includes the type of algorithm, features and transformations, data snapshots, hyperparameters, performance metrics, verifiable code from source code management, and the training environment.

Repeatability also makes debugging much easier.

Reproducibility and Auditability

AUDITABILITY

Auditability is related to reproducibility, but it adds some requirements.

For a model to be auditable, it must be possible to access the full history of the ML pipeline from a central and reliable storage and to easily fetch metadata on all model versions including:

- The full documentation
- An artifact that allows running the model with its exact initial environment
- Test results, including model explanations and fairness reports
- Detailed model logs and monitoring metadata

Reproducibility & Documentation

All facets of the model need to be documented and reusable, including:

- **Assumptions**: When a data scientist makes decisions and assumptions they should all be explicit and logged.
- **Randomness**: A lot of ML algorithms and processes, such as sampling, make use of pseudo-random numbers.
- **Data**: To get repeatability, the same data must be available.
- **Settings**: All processing that has been done must be reproducible with the same settings.
- **Results**: Data scientists need to be able to compare in-depth analysis of models
- **Implementation**: Different implementations of the same model can actually yield different models, enough to change the predictions on some close calls.
- **Environment**: the environment in which several of those steps run may be more or less implicitly tied to the results. Preferably, data scientists should make sure that the runtime environment is also repeatable. Given the pace at which ML is evolving, this might require techniques that freeze the computation environments.

Fortunately, part of the underlying documentation tasks associated reproducibility can be **automated**, and the use of an integrated platform for design and deployment can greatly decrease the reproducibility costs

ML Security

ML security management needs to secure and manage endpoints to make sure that only **authorized users** can create, change, or delete endpoints. **Authentication, SSO, and role-based access control (RBAC)** are further important aspects of ML security.

In addition, **key and secret management** should provide solutions for creating, saving, and managing keys.

Finally, models are supposed to pass **security audits**. Therefore, it is very recommendable to involve IT and company experts from the beginning of a project to fully consider all security requirements.

TOOL

Protection against **ML-specific cybersecurity attacks** is also very important. With models often being trained on sensitive data, data and information security play an important role.

Using the Adversarial ML Threat Matrix might be helpful.

ML Security

Machine learning introduces a new range of potential threats where an attacker provides malicious data designed to cause the model to make a mistake.

ADVERSARIAL ATTACKS

A more modern but quite analogous example of a machine learning model security issue is an adversarial attack for deep neural networks in which an image modification that can seem minor or even impossible for a human eye to notice can cause the model to drastically change its prediction.

If an attacker can get access to the training data, even partially, then they get control over the system. This kind of attack is traditionally known as a **poisoning attack** in computer security.

ML Security

OTHER VULNERABILITIES

Models can also leak data in many ways. Since the machine learning models can fundamentally be considered a summary of the data they have been trained on, they can leak more or less precise information on the training data.

Responsibilities must be assigned clearly and in a way that ensures an appropriate balance between security and capacity of execution.
It is also important to put in place feedback mechanisms, and employees and users should have an easy channel to communicate breaches

Model Risk Mitigation

CHANGING ENVIRONMENTS

The speed of change can amplify the risk depending on the application.

To control this risk, monitoring via MLOps should be reactive enough and the procedure should consider the period necessary for remediation.

If the environment changes often, even if remediation never seems urgent, a model can always be slightly off, never operating within its nominal case, and the operating cost can be challenging to evaluate. This can only be detected through dedicated MLOps, including relatively long-term monitoring and reevaluating the cost of operating the model.

Model Risk Mitigation

INTERACTIONS BETWEEN MODELS

Complex interactions between models is probably the most challenging source of risk. This class of issue will be a growing concern as ML models become pervasive, and it's an important potential area of focus for MLOps systems.

Moreover, the total complexity is heavily determined by how the interactions with models are designed at a local scale and governed at an organizational scale. Using models in chains (where a model uses inputs from another model) can create significant additional complexity as well as totally unexpected results.

MODEL MISBEHAVIOR

A number of measures can be implemented to avoid model misbehavior, including examining its inputs and outputs in real time. While training a model, it is possible to characterize its domain of applicability by examining the intervals on which the model was trained and validated.

More sophisticated methods can be used, including anomaly detection to identify records where the model is used outside of its application domain.

Model Risk Mitigation

RISK ASSESSMENT

Risk assessment should be performed at the beginning of each project and reassessed periodically.
Example of risk: unavaiable model for a given periodof time, bad prediction returned by model, decrease of accuracy or fairness.

Risk assessment is generally based on two metrics: the probability and the impact of the adverse event.

5 x 5 risk matrix

Probability						
	Highly probable	5 Moderate	10 Major	15 Major	20 Severe	25 Severe
	Probable	4 Moderate	8 Moderate	12 Major	16 Major	20 Severe
	Possible	3 Minor	6 Moderate	9 Moderate	12 Major	15 Major
	Unlikely	2 Minor	4 Moderate	6 Moderate	8 Moderate	10 Major
	Rare	1 Minor	2 Minor	3 Minor	5 Moderate	6 Moderate
		Very low	Low	Medium	High	Very high
				Impact		

RISK MITIGATION

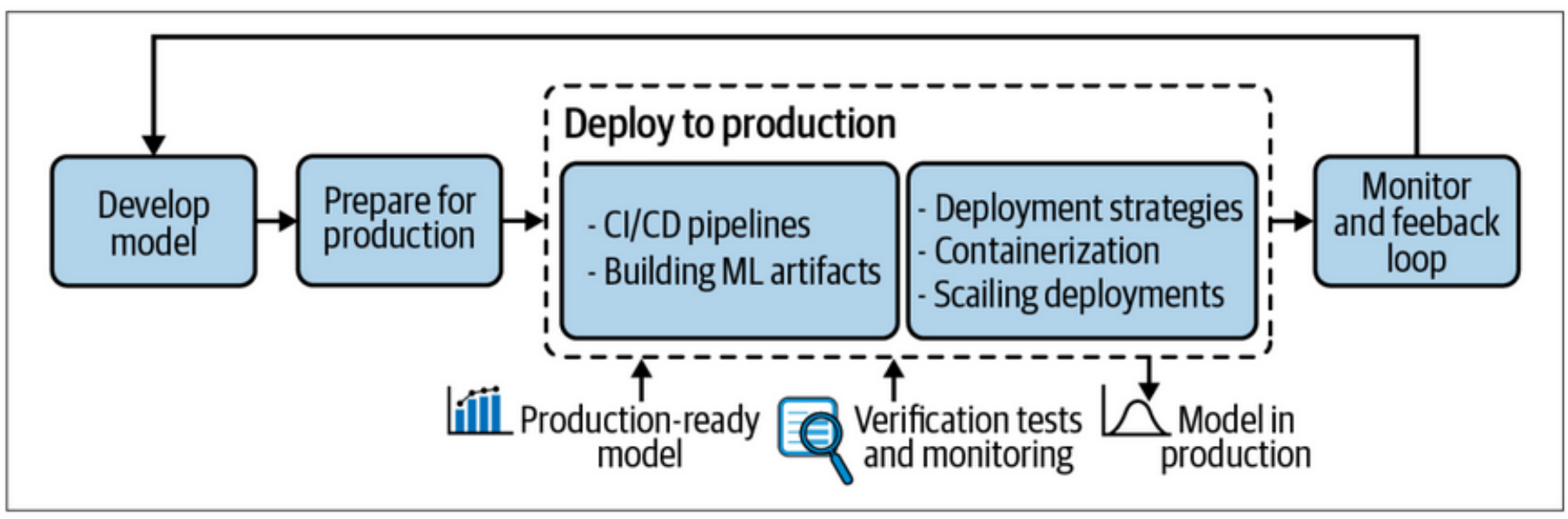
MLOps infrastructure is important for machine learning models because prediction models are only as good as the data they are trained on, which means the training data must be a good reflection of the data encountered in the production environment.
Another reason, machine learning model performance is often very sensitive to the production environment it is running in, including the versions of software and operating systems in use.

Deploying to Production

MLOps • 2022

Deploying to Production

The **to-be-deployed models** depend on the technology chosen and comprise a set of code (commonly Python, R, or Java) and data artifacts. Any of these can have version dependencies on runtimes and packages that need to match in the production environment because the use of different versions may cause model predictions to differ.



CI/CD Pipelines

CI/CD is a common acronym for continuous integration and continuous delivery. CI/CD methodology should be adapted based on the needs of the team and the nature of the business.

After successfully developing a model, a data scientist should push the code, metadata, and documentation to a central repository and trigger a CI/CD pipeline.
An example of such pipeline could be:

1. Build the model
    a. Build the model artifacts
    b. Send the artifacts to long-term storage
    c. Run basic checks (smoke tests/sanity checks)
    d. Generate fairness and explainability reports
2. Deploy to a test environment
    a. Run tests to validate ML performance, computational performance
    b. Validate manually
3. Deploy to production environment
    a. Deploy the model as canary
    b. Fully deploy the model

The desire in MLOps is to automate the CI/CD pipeline as far as possible. This not only speeds up the deployment process, but it enables more extensive regression testing and reduces the likelihood of errors in the deployment.

Bulding ML Artifacts

The very first step is using centralized version control systems.

The most common version control system is Git, an open source software initially developed to manage the source code for the Linux kernel. It allows for maintaining a clear history of changes, safe rollback to a previous version of the code, multiple contributors to work on their own branches of the project before merging to the main branch, etc. While Git is appropriate for code, it was not designed to store other types of assets common in data science workflows.

WHAT'S IN AN ML ARTIFACT?

Once the code and data are in a centralized repository, a testable and deployable bundle of the project must be built. These bundles are usually called **artifacts** in the text of CI/CD.
Each of the following elements needs to be bundled into an artifact that goes through a testing pipeline and is made available for deployment to production:

- Code for the model and its preprocessing
- Hyperparameters and configuration
- Training and validation data
- Trained model in its runnable form
- An environment including libraries with specific versions, environment variables, etc.
- Documentation
- Code and data for testing scenarios

Bulding ML Artifacts

THE TESTING PIPELINE

the testing pipeline can validate a wide variety of properties of the model contained in the artifact. One of the important operational aspects of testing is that good tests should make it as easy as possible to diagnose the source issue when they fail.

Automating these tests as much as possible is essential and, indeed, is a key component of efficient MLOps. A lack of automation or speed wastes time, but, more importantly, it discourages the development team from testing and deploying often, which can delay the discovery of bugs or design choices that make it impossible to deploy to production.

The most widespread tool for software engineering continuous integration is Jenkins, a very flexible build system that allows for the building of CI/CD pipelines regardless of the programming language, testing framework, etc. Jenkins can be used in data science to orchestrate CI/CD pipelines, although there are many other options.

Deployment Strategies

DEPLOYMENT, WHAT IS?

The process of running a new model version on a target infrastructure. Fully automated deployment is not always practical or desirable and is a business decision as much as a technical decision,

CONSIDERATIONS WHEN SENDING MODELS TO PRODUCTION

When sending a new model version to production, the first consideration is often to avoid downtime, in particular for real-time scoring.

The basic idea is that rather than shutting down the system, upgrading it, and then putting it back online, a new system can be set up next to the stable one, and when it's functional, the workload can be directed to the newly deployed version. This deployment strategy is called blue-green—or sometimes red-black—deployment.
There are many variations and frameworks (like Kubernetes) to handle this natively.

Another more advanced solution to mitigate the risk is to have canary releases (also called canary deployments). The idea is that the stable version of the model is kept in production, but a certain percentage of the workload is redirected to the new model, and results are monitored.

- Canary testing can be used to carry out A/B testing, which is a process to compare two versions of an application in terms of a business performance metric.

Deployment Strategies

MAINTENANCE IN PRODUCTION

There are three maintenance measures:

- Resource monitoring: Just as for any application running on a server, collecting IT metrics such as CPU, memory, disk, or network usage can be useful to detect and troubleshoot issues.
- Health check: To check if the model is indeed online and to analyze its latency, it is common to implement a health check mechanism that simply queries the model at a fixed interval (on the order of one minute) and logs the results.
- ML metrics monitoring: This is about analyzing the accuracy of the model and comparing it to another version or detecting when it is going stale.

Finally, when a malfunction is detected, a rollback to a previous version may be necessary.

Containerization

Containerization technology is increasingly used to tackle these challenges. These tools bundle an application together with all of its related configuration files, libraries, and dependencies that are required for it to run across different operating environments. Unlike **virtual machines (VMs)**, containers do not duplicate the complete operating system; multiple containers share a common operating system and are therefore far more resource efficient.

The most well-known containerization technology is the open source platform Docker. Released in 2014, it has become the de facto standard. It allows an application to be packaged, sent to a server (the Docker host), and run with all its dependencies in isolation from other applications.

Scaling Deployments

As ML adoption grows, organizations face two types of growth challenges:

- The ability to use a model in production with high-scale data
- The ability to train larger and larger numbers of models

Handling more data for real-time scoring is made much easier by frameworks such as Kubernetes. Since most of the time trained models are essentially formulas, they can be replicated in the cluster in as many copies as necessary. With the auto-scaling features in Kubernetes, both provisioning new machines and load balancing are fully handled by the framework, and setting up a system with huge scaling capabilities is now relatively simple.

Scaling Deployments

When the volume of data becomes too large, there are essentially two types of strategies to distribute the computation:

- Using a framework that handles distributed computation natively, in particular Spark. Spark is an open source distributed computation framework. It is useful to understand that Spark and Kubernetes do not play similar roles and can be combined. Kubernetes orchestrates containers, but Kubernetes is not aware of what the containers are actually doing; as far as Kubernetes is concerned, they are just containers that run an application on one specific host. Spark is a computation framework that can split the data and the computation among its nodes. A modern way to use Spark is through Kubernetes. To run a Spark job, the desired number of Spark containers are started by Kubernetes; once they are started, they can communicate to complete the computation, after which the containers are destroyed and the resources are available for other applications, including other Spark jobs that may have different Spark versions or dependencies.
- Another way to distribute batch processing is to partition the data. There are many ways to achieve this, but the general idea is that scoring is typically a row-by-row operation (each row is scored one by one), and the data can be split in some way so that several machines can each read a subset of the data and score a subset of the rows.

Scaling Deployments

SCALABLE SYSTEM

A computational system is said to be horizontally scalable (or just scalable) if it is possible to incrementally add more computers to expand its processing power. For example, a Kubernetes cluster can be expanded to hundreds of machines.

ELASTIC SYSTEM

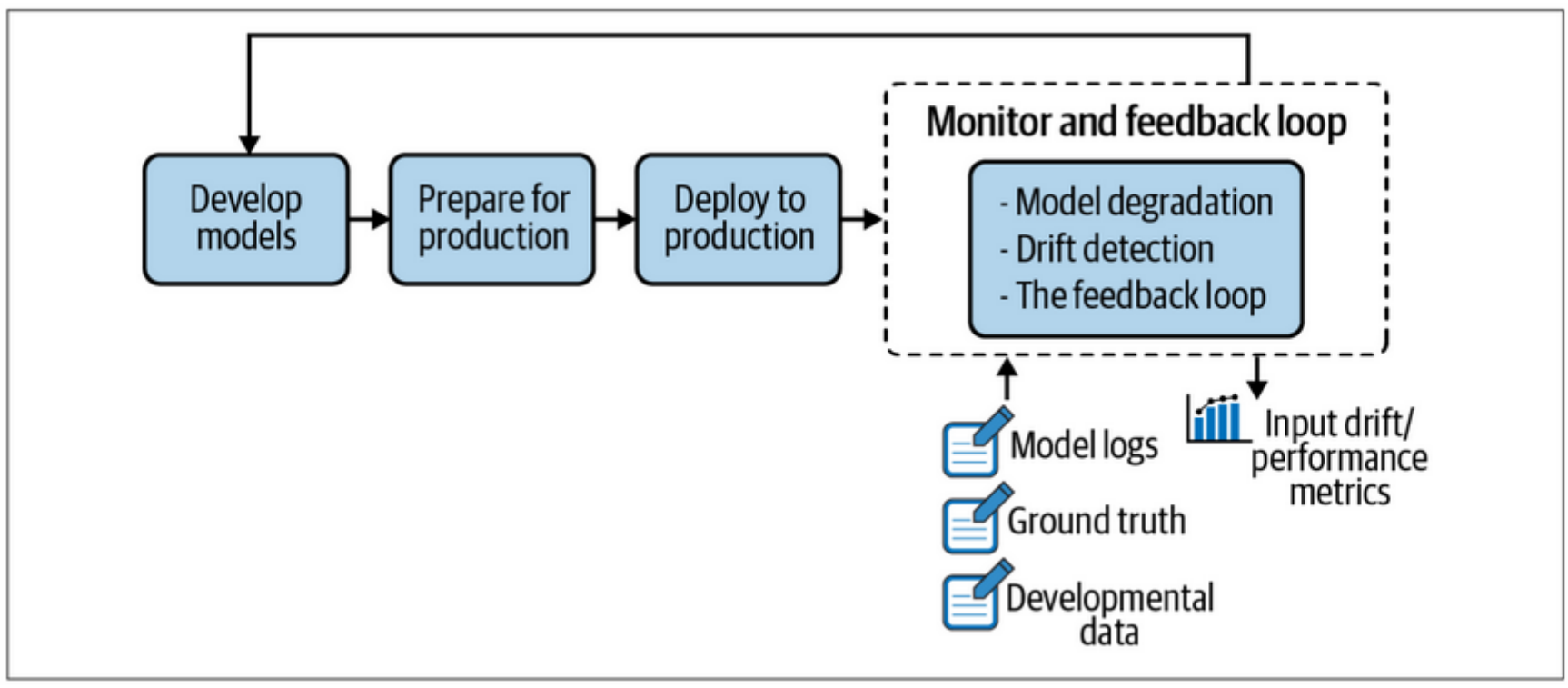
An elastic system allows, in addition to being scalable, easy addition and removal of resources to match the compute requirements. For example, a Kubernetes cluster in the cloud can have an auto-scaling capability that automatically adds machines when the cluster usage metrics are high and removes them when they are low. In principle, elastic systems can optimize the usage of resources; they automatically adapt to an increase in usage without the need to permanently provision resources that are rarely required.

CURIOSITY

A dedicated tool to manage large numbers of pipelines can also be used; for example, Netflix released Spinnaker, an open source continuous deployment and infrastructure management platform.

Monitoring

Monitoring



Machine learning models need to be monitored at two levels:

- At the resource level, including ensuring the model is running correctly in the production environment. It's a traditional DevOps topic.
- At the performance level, meaning monitoring the pertinence of the model over time. Model performance monitoring attempts to track this degradation, and, at an appropriate time, it will also trigger the retraining of the model with more representative data.

Monitoring

Once the ML model has been deployed, it need to be monitored to assure that the ML model performs as expected.

An example of check list for model monitoring activities in production:

- Monitor **dependency changes** throughout the complete pipeline result in notification.
    - Data version change.
    - Changes in source system.
    - Dependencies upgrade.
- Monitor **data invariants** in training and serving inputs: Alert if data does not match the schema, which has been specified in the training step.
- Monitor whether **training and serving features** compute the same value.
- Monitor **computational performance** of an ML system. Both dramatic and slow-leak regression in computational performance should be notified.
- Monitor how stale the system in production is.
    - Measure the age of the model. Older ML models tend to **decay in performance**.
- Monitor **degradation of the predictive quality** of the ML model on served data. Both dramatic and slow-leak regression in prediction quality should be notified.

How often should models be retrained?

Unfortunately, there is no easy answer, as this question depends on many factors, including:

- The domain: Models in areas like cybersecurity or real-time trading need to be updated regularly to keep up with the constant changes inherent in these fields. Physical models, like voice recognition, are generally more stable, because the patterns don’toften abruptly change.
- The cost: Organizations need to consider whether the cost of retraining is worth the improvement in performance.
- The model performance: the model performance is restrained by the limited number of training examples, and thus the decision to retrain hinges on collecting enough new data.

It is therefore critical for organizations to have a clear idea of **deployed models’ drift** and **accuracy** by setting up a process that allows for easy monitoring and notifications. An ideal scenario would be a pipeline that **automatically triggers checks** for degradation of model performance.

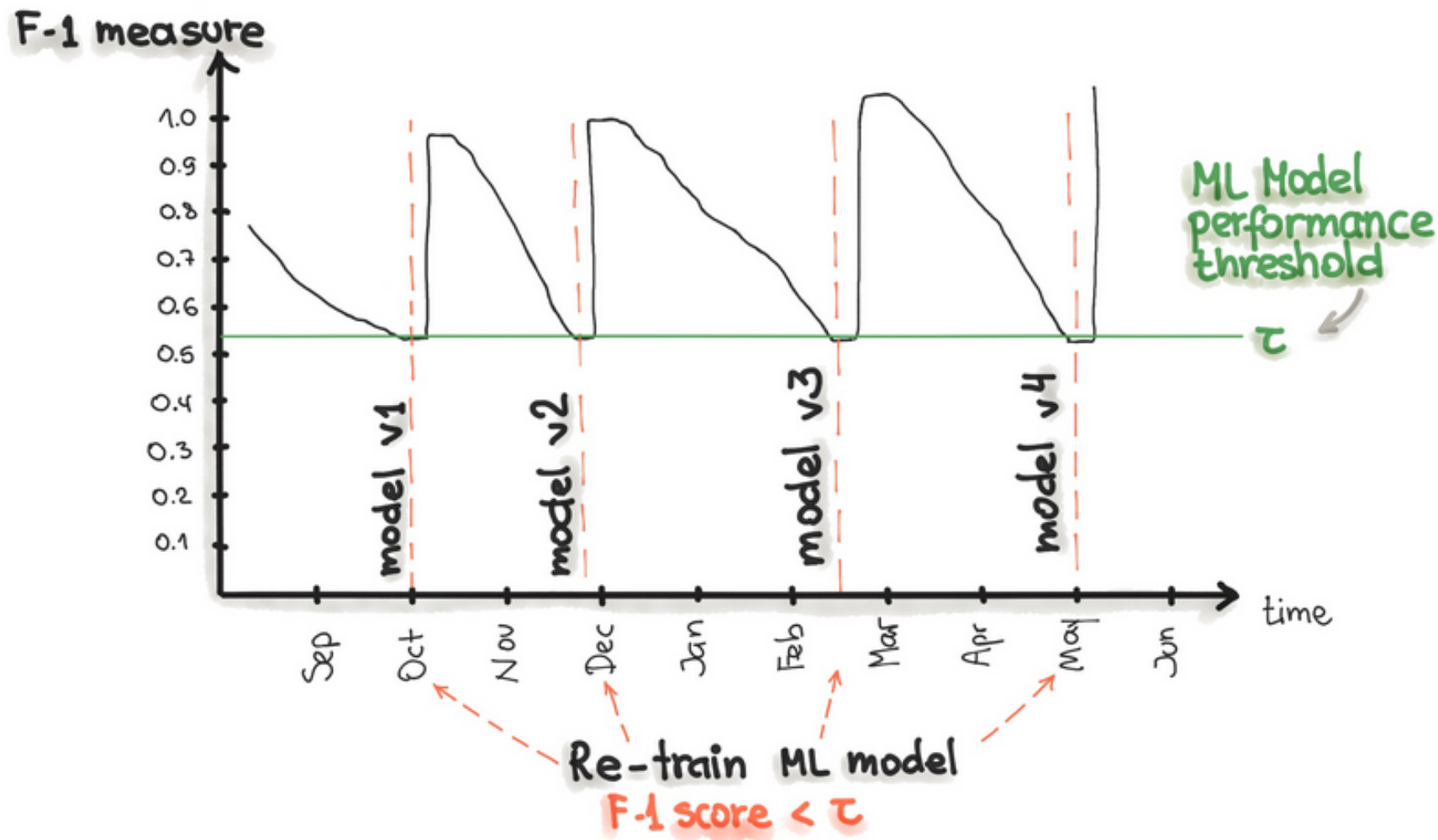
Monitoring & Notifications

It’s important to note that the goal of **notifications** is not necessarily to kick off an automated process of retraining, validation, and deployment. Model performance can change for a variety of reasons, and retraining may not always be the answer. The point is **to alert the data scientist** of the change; that person can then diagnose the issue and evaluate the next course of action.

Practically, every deployed model should come with **monitoring metrics** and corresponding **warning thresholds** to detect meaningful business performance drops as quickly as possible.

Model Decay Monitoring

Now see that the decrease of the precision, recall, and F1-score triggers the model retraining, which leads to model recovery.



Understanding Model Degradation

Once a machine learning model is trained and deployed in production, there are two approaches to monitor its performance degradation: **ground truth evaluation** and **input drift detection**.

GROUND TRUTH EVALUATION

Ground truth retraining requires **waiting for the label event**.
With the new ground truth collected, the next step is to compute the performance of the model based on ground truth and compare it with registered metrics in the training phase. When the difference surpasses a threshold, the model can be deemed as outdated, and it should be retrained.

The metrics to be monitored can be of two varieties:

- Statistical metrics like accuracy, ROC AUC, log loss, etc.
- Business metrics, like cost-benefit assessment.

The main advantage of the first kind of metric is that it is domain agnostic, so the data scientist likely feels comfortable setting thresholds.

There are three main challenges:

- Ground truth is not always immediately, or even imminently, available. For some types of models, teams need to wait months (or longer) for ground truth labels to be available, which can mean significant economic loss if the model is degrading quickly.
- Ground truth is only partially available. In some situations, it is extremely expensive to retrieve the ground truth for all the observations, which means choosing which samples to label and thus inadvertently introducing bias into the system.

Understanding Model Degradation

INPUT DRIFT DETECTION

It's a more practical approach.
The resulting distribution shift is called a drift.

Input drift is based on the principle that a model is only going to predict accurately if the data it was trained on is an accurate reflection of the real world. The beauty of this approach is that all the data required for this test already exists, so there is no need to wait for ground truth or any other information. Identifying drift is one of the most important components of an adaptable MLOps strategy, and one that can bring agility to the organization’s enterprise.

Drift Detection in practise

The logic is that if the data distribution (e.g., mean, standard deviation, correlations between features) diverges between the **training and testing phases** on one side and the **development phase** on the other, it is a strong signal that the model's performance won't be the same.

EXAMPLE CAUSES OF DATA DRIFT

There are two frequent root causes of data drift:

- Sample selection bias, where the training sample is not representative of the population.
- Non-stationary environment, where training data collected from the source population does not represent the target population. This often happens for timedependent tasks with strong seasonality effects.

Drift Detection in practise

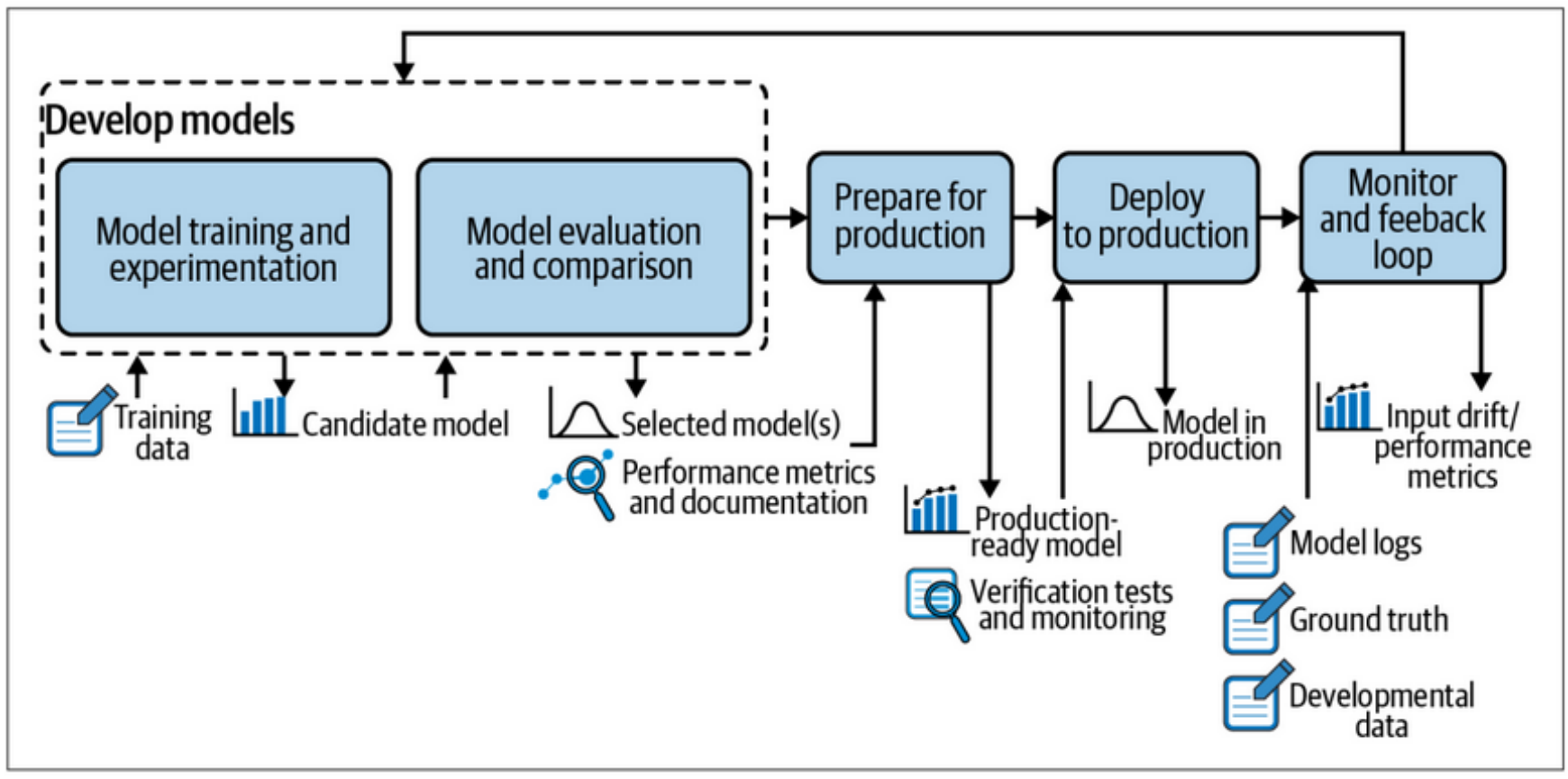
INPUT DRIFT DETECTION TECHNIQUES

There are two common approaches.

- Univariate statistical tests
    - This method requires applying a **statistical test** on data from the source distribution and the target distribution for each feature.
    - The basic approaches rely on these two tests:
        - For continuous features, the Kolmogorov-Smirnov test is a nonparametric hypothesis test that is used to check whether two samples come from the same distribution.
        - For categorical features, the Chi-squared test is a practical choice that checks whether the observed frequencies for a categorical feature in the target data match the expected frequencies seen from the source data.
- Domain classifier:
    - data scientists train a model that tries to discriminate between the original dataset (input features and, optionally, predicted target) and the development dataset. In other words, they stack the two datasets and train a classifier that aims at predicting the data's origin. The performance of the model (its accuracy, for example) can then be considered as a metric for the drift level.
    - If this model is successful in its task it implies that the data used at training time and the new data can be distinguished, so it's fair to say that the new data has drifted.

The Feedback loop

The Feedback loop



Data collected in the monitoring and feedback loop is sent to the model development phase. From there, the system analyzes whether the model is working as expected. If the model's performance is **degrading**, an update will be triggered, either automatically or manually by the **data scientist**. This usually means either retraining the model with new labeled data or developing a new model with additional features.

This infrastructure is comprised of three main components:

- **A logging system** that collects data from several production servers
- **A model evaluation store** that does versioning and evaluation between different model versions
- **An online system** that does model comparison on production environments, either with the **shadow scoring (champion/challenger)** setup or with **A/B testing**

Logging System

Monitoring a live system, with or without machine learning components, means collecting and aggregating data about its states.

An event log of a machine learning system is a record with a **timestamp** and the following information.

- **Model metadata:** Identification of the model and the version.
- **Model inputs:** Feature values of new observations, which allow for verification of whether the new incoming data is what the model was expecting and thus allowing for detection of data drift.
- **Model outputs:** Predictions made by the model that give a concrete idea about the model performance in a production environment.
- **System action:** the more common situation than the system will take an action based on this prediction.
- **Model explanation:** In some highly regulated domains such as finance or healthcare, predictions must come with an explanation. This kind of information is usually computed with techniques such as Shapley value computation and should be logged to identify potential issues with the model

Logging System

FEATURES AND DATA TESTS

- Data validation: Automatic check for data and features schema/domain.
- Features importance test to understand whether new features add a predictive power.
    - Measure data dependencies, inference latency, and RAM usage for each new feature. Compare it with the predictive power of the newly added features.
    - Drop out unused/deprecated features from your infrastructure and document it.
- Features and data pipelines should be policy-compliant (e.g. GDPR).
- Feature creation code should be tested by unit tests (to capture bugs in features).

Model Evaluation

Once the logging system is in place, it periodically fetches data from the production environment for monitoring.

With several trained candidate models, the next step is to compare them with the deployed model. In practice, this means evaluating all the models (the candidates as well as the deployed model) on the same dataset. If one of the candidate models outperforms the deployed model, there are two ways to proceed:

- either update the model on the production environment
- move to an online evaluation via a **champion/challenger** or **A/B testing setup**.

Testing ML training should include routines, which verify that algorithms make decisions aligned to business objective. This means that ML algorithm loss metrics (MSE, log-loss, etc.) should correlate with business impact metrics (revenue, user engagement, etc.)

Formally, the model evaluation store serves as a structure that centralizes the data related to model life cycle to allow comparisons. By definition, all these comparisons are grouped under the umbrella of a **logical model**.

Model Evaluation

LOGICAL MODEL

A logical model is a **collection of model templates and their versions** that aims to solve a business problem.
A **model version** is obtained by training a model template on a given dataset. All versions of model templates of the same logical model can usu ally be evaluated on the same kinds of datasets.
The information about the evaluation of these versions on various datasets is then stored in the model evaluation store.

MODEL EVALUATION STORE

Model evaluation stores are structures that centralize the data related to model life cycles to allow comparisons.

The two main tasks of a model evaluation store are:

- **Versioning** the evolution of a logical model through time. Each logged version of the logical model must come with all the essential information concerning its training phase,
- **Comparing** the performance between different versions of a logical model. To decide which version of a logical model to deploy, all of them must be evaluated on the same dataset.

Online Evaluation

This online evaluation gives the most truthful feedback about the behavior of the candidate model when facing **real data**.
There two main modes of online evaluation:

- **Champion/challenger** (otherwise known as **shadow testing**), where the candidate model shadows the deployed model and scores the same live requests
- **A/B testing**, where the candidate model scores a portion of the live requests and the deployed model scores the others

Both cases require **ground truth**, so the evaluation will necessarily take longer than the lag between prediction and ground truth obtention.

Online Evaluation

CHAMPION/CHALLENGER

Champion/challenger involves deploying one or several additional models (the challengers) to the production environment. These models receive and score the same incoming requests as the active model (the champion). The predictions are simply logged for further analysis.

This setup allows for two things:

- Verification that the **performance** of the new models is better than, or at least as good as, the old model. Because the two models are scoring on the **same data**, there is a direct comparison of their **accuracy** in the production environment.
- Measurement of how the new models **handle realistic load**.

To be able to compare the champion and the challenger models, the **same information must be logged for both**, including input data, output data, processing time, etc.

Online Evaluation

A/B TESTING

It is a widely used technique in website optimization. For ML models, it should be used only when champion/challenger is not possible.

Unlike the champion/challenger framework, with A/B testing, the candidate model returns predictions **for certain requests**, and the original model handles the other requests. Once the test period is over, **statistical tests** compare the performance of the two models, and teams can make a decision based on the statistical significance of those tests.

Before the A/B test
Define a **qualitative business metric**, define a precise **population**, define the **statistical protocol,** because the resulting metrics are compared using statistical tests.

During the A/B test
It is important not to stop the experiment.

After the A/B test
Check the collected data to make sure the **quality** is good. From there, run the **statistical tests**; if the metric difference is statistically significant in favor of the candidate model, the original model can be replaced with the new version.

Online Evaluation

BAYESIAN TESTING

Bayesian, and in particular **multi-armed bandit**, tests are an increasingly popular alternative.

Multi-armed bandit testing is adaptive: the algorithm that decides the split between models adapts according to live results and **reduces the workload** of underperforming models.

While multi-armed bandit testing is more **complex**, it can **reduce the business cost** of sending traffic to a poorly performing model.

Iteration

ITERATING ON THE EDGE

- Iterating on an ML model deployed to millions of devices, such as smartphones, sensors, or cars, presents different challenges to iteration in a corporate IT environment.
- One approach is to relay all the feedback from the millions of model instances to a central point and perform training centrally. This as Tesla's autopilot system.
- Google has taken a different approach with its smartphone keyboard software, GBoard. Instead of centralized retraining, every smartphone retrains the model locally and sends a summary of the improvements it has found to Google centrally. These improvements from every device are averaged and the shared model updated.

ML infrastructure test

Most important test:

DIFF-TEST

Training the ML models should be reproducible, which means that training the ML model on the same data should produce **identical ML models**.
Diff-testing of ML models relies on deterministic training, which is hard to achieve due to non-convexity of the ML algorithms, random seed generation, or distributed ML model training.

STRESS TESTING

Test ML API usage.

- **Unit tests** to randomly generate input data and training the model for a single optimization step (e.g gradient descent).
- **Crash tests** for model training. The ML model should restore from a checkpoint after a mid-training crash.

CANARY TESTING

ML models are canaried before serving.
Testing that an ML model successfully loads into production serving and the prediction on real-life data is generated as expected.

Model Governance

MLOps • 2022

Model Governance

Model Governance encompasses a set of processes and frameworks that help in the deployment of ML.

Governance is the set of controls placed on a business to ensure that it delivers on its responsibilities to all stakeholders, from shareholders and employees to the public and national governments.
These responsibilities include financial, legal, and ethical obligations.

The operationalization of ML models is also the use of MLobliges companies responsibility and compliance with legal requirements.

To fulfill these obligations, a company requires processes through which it is able to:

- control access to ML models
- put guidelines and legal requirements into practice
- track interactions with the ML models and their results
- document the foundation of an ML model (stakeholders, business context, training data, feature selection, guidelines for model reproduction, choice of parameters, results of model evaluation and validation) Collectively, these processes are referred to as Model Governance.

Main Components

It should be clear that we need MLOps and Model Governance to deploy ML models successfully.

- **Comprehensive model documentation or reports.** This includes the report of metrics by using appropriate visualization techniques and dashboards
- **Versioning of all models** to create transparency for stakeholders (explainability and reproducibility)
- **Auditing of ML systems** (automated approval auditing or CE certification as part of conformity testing)
- Comprehensive data documentation to guarantee **high data quality and adherence to data protection**
- **Management of ML metadata**
- **Validation of ML models**
- **Continuous monitoring and logging** of model metrics

Governments

Recently, governments across the world have imposed regulations to protect the public from the impact of the use of personal data by businesses.

- The 2016 EU General Data Protection Regulation (GDPR)
- The 2018 California Consumer Privacy Act (CCPA)

Governments are now starting to turn their regulatory eye to ML specifically, hoping to mitigate the negative impact of its use. ML is rapidly being embedded in decision-making systems that impact every aspect of our lives. The European Union is leading the way with planned legislation to define the acceptable uses of various forms of AI.

Governments

PUBLIC OPINION

Public opinion determines what products people buy, where they invest their money.
The general public needs to be reassured that ML is fair. Right now, opinion on ML is in the balance.

But there have also been well-publicized scandals that have rocked the public's acceptance of this technology. The **Facebook-Cambridge Analytica affair**, where the companies used the power of ML to manipulate public opinion on social media, shocked the world. This looked like ML with explicitly malicious intent.

If businesses and governments want to reap the benefits of ML, this means **developing strong governance** of their MLOps process. They must assess the risks, determine their own set of fairness values, and then implement the necessary process to manage them.

Those responsible for MLOps must manage the inherent tension between different user profiles, striking a balance between getting the **job done efficiently** and **protecting against all possible threats**. This balance can be found by assessing the specific risk of each project and matching the governance process to that risk level.

Governance to MLOps

Governance initiatives in MLOps fall into one of two categories:

- Data governance
- Process governance

DATA GOVERNANCE

A framework for ensuring appropriate use and management of data
ML projects usually involve significant pipelines, consisting of data cleaning, combination, and transformation steps.
Data governance tools that can address these problems are in their infancy.

PROCESS GOVERNANCE

Process governance focuses on formalizing the steps in the MLOps process and associating actions with them. Typically these actions are reviews, sign-offs, and the capture of supporting materials, such as documentation.
The aim is twofold:

- To ensure every governance-related consideration is made at the correct time and correctly acted upon.
- To enable oversight from outside of the strict MLOps process. Auditors, risk managers, compliance officers, and the business as a whole all have an interest in being able to track progress and review decisions at a later stage.

Current Regulations

Many existing regulations do, however, have a significant impact on ML governance. These take two forms:

- **Industry-specific regulation**. This is particularly significant in the finance and pharmaceutical sectors. Examples:
    - GxP,Good Clinical Practice, or GCP, guidelines;
    - Black–Scholes model, financial Model Risk Management Regulation ;
    - UK Prudential Regulation Authority's (PRA) regulation, defines principles for good MRM (Model risk management)
- **Broad-spectrum regulation**, particularly addressing data privacy.
    - Example EU General Data Protection Regulation (GDPR)

Current Regulations

GDPR AND CCPA DATA PRIVACY REGULATIONS

GDPR attempts to protect personal data from industrial misuse with a goal of limiting the potential discrimination against individuals. The GDPR sets out principles for the processing of personal data, and processing includes the collection, storage, alteration, and use

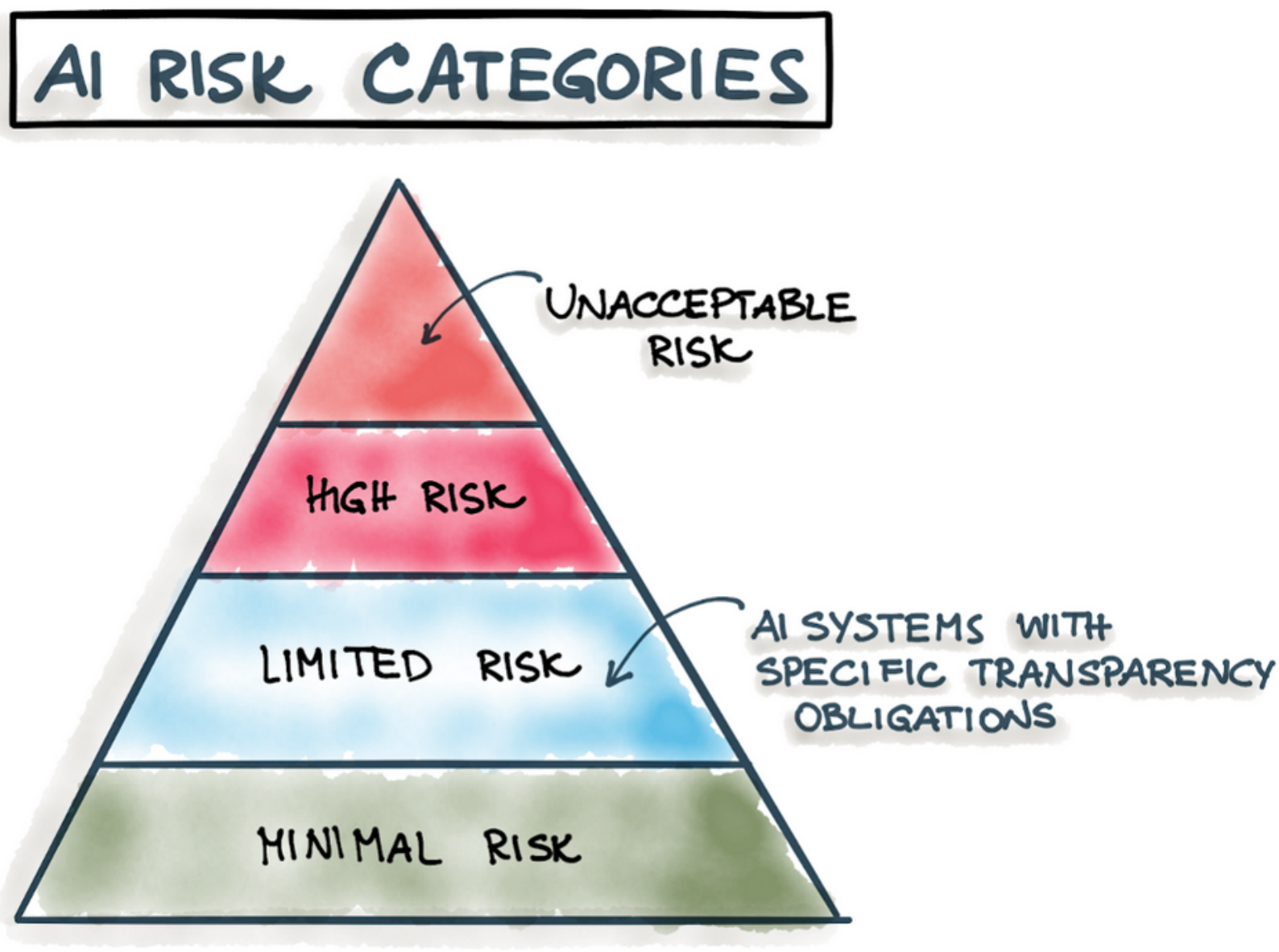
GDPR Principles:

- Lawfulness, fairness, and transparency
- Purpose limitation
- Data minimization
- Accuracy
- Storage limitation
- Integrity and confidentiality (security)
- Accountability

The California Consumer Privacy Act (CCPA) is quite similar to GDPR in terms of who and what is protected, although the scope, territorial reach, and financial penalties are all more limited.

Risk Categories

The risk category determines the scope of the regulations: the higher the determined risk, the more requirements have to be met.



Risk Categories

1. UNACCEPTABLE RISK

AI software considered to be a significant risk for security, livelihoods, and human rights is forbidden

2. HIGH RISK

AI software is subject to strict requirements. These include the following aspects: Robustness, security, accuracy (precision), documentation and logging as well as appropriate risk assessment and mitigation. Further requirements include high-quality training data, non-discrimination, traceability, transparency, human monitoring.

Examples of ML systems in this category include **private and public services (credit scoring) or systems used in education or vocational training** to decide on a person's access to education and career path (e.g., exam scoring).

3. LIMITED RISK

This AI software is subject to a transparency obligation. For example, chatbot users must be informed that they are interacting with AI software.

4. MINIMAL RISK

AI software in this category is not subject to any regulation (e.g., spam filters).

Responsible AI

The most common banner for this consensus is Responsible AI: the idea of developing machine learning systems that are **accountable, sustainable, and governable**.
Responsible AI covers two dimensions: intentionality and accountability.

INTENTIONALITY

Ensuring that models are designed and behave in ways aligned with their purpose.
Intentionality also includes explainability, meaning the results of AI systems should be explainable by humans

ACCOUNTABILITY

Accountability is about having an overall view of which teams are using what data, how, and in which models.
Central control, management, and the ability to audit the enterprise AI effort.
Trust that data is reliable and being collected in accordance with regulation as well as a centralized understanding of which models are being used for which business process.
This is closely tied to traceability: if something goes wrong, is it easy to find where in the pipeline it happened?

Responsible AI: Key Elements

Responsible AI is about the responsibility of data practitioners, not about AI itself being responsible: this is a very important distinction.

ELEMENT 1: DATA

The quality of the data used will make the biggest impact on the accuracy of the model. Some considerations:

- The quality of data over time: consistency, completeness, and ownership.
- Data must be manageable, securable, and traceable. Personal data must be strictly managed.

ELEMENT 2: BIAS

Bias problems are not new; for example, hiring discrimination has always been an issue. What is new is that, thanks to the IT revolution, data to assess biases is more available.
Bias checks should be integrated in governance frameworks so that issues are identified as early as possible.

Responsible AI: Key Elements

ELEMENT 3: INCLUSIVENESS

The human-in-the-loop (HITL) approach aims to combine the best of human intelligence with the best of machine intelligence. Machines are great at making smart decisions from vast datasets, whereas people are much better at making decisions with less information. Human judgment is particularly effective for making **ethical and harm-related judgments**.

ELEMENT 4: MODEL MANAGEMENT AT SCALE

Some key considerations for managing ML at scale:

- A scalable model life cycle needs to be largely automated.
- Errors will propagate out rapidly and widely.
- Existing software engineering techniques can assist ML at scale.
- Decisions must be explainable, auditable, and traceable.
- Reproducibility is key to understanding what went wrong, who or what was responsible, and who should ensure it is corrected.
- Model performance will degrade over time: monitoring, drift management, retraining, and remodeling must be built into the process.

Responsible AI: Key Elements

ELEMENT 5: GOVERNANCE

Responsible AI sees strong governance as the key to achieving fairness and trustworthiness.

Governance is, therefore, both the foundation and the glue of MLOps initiatives.

Model Governance as Part of Model Management

Model governance encompasses the **recording, auditing, validation, approval, and monitoring** of models.
Model governance is the **final supervisory authority** to approve a model for being deployed into the production environment.

The list below summarizes the model governance components with the necessary tasks and artifacts. For the implementation of these tasks, model governance uses information from the **ML metadata, the artifact repository, and the model registry**.

The **model registry** saves all model versions to ensure reproducibility and accountability

The **evaluation** of a model is very important as well. To evaluate a model candidate, it can be released using shadow deployment. Then, its performance can be compared to the current productive model by comparing performance and business metrics

Model audits are another fundamental aspect. Any model changes must be checked and approved to control risks in different categories. The auditing component in this model governance framework can be considered to be a less strict variant of the certificate of conformity.

Finally, a report contains the summary, visualization, and highlighting of model performance metrics collected during the **monitoring** process.

A Template for MLOps Governance

How to implement a robust governance framework of MLOps, 8 steps:

1. Understand and classify the analytics use cases.
    - Identify the key distinguishing features of the different use cases and categorize these features.
2. Establish an ethical position.
    - The position a business takes is a trade-off between the cost to implement the position and public perception.
    - The MLOps governance process needs to ensure that deployed models match the chosen ethical stance.
3. Establish responsibilities.
    - Identify the groups of people responsible for overseeing MLOps governance as well as their roles.
4. Determine governance policies.
    - The finalized governance policies should provide:
        - A process for determining the classification of any analytics initiative.
        - A matrix of initiative classification against governance consideration, where each cell identifies the measures required.
5. Integrate policies into the MLOps process.
    - Now is the time to revisit, enhance, and document the process. Successful adoption of the governance process can only happen if it is communicated clearly and buy-in is sought from each stakeholder group.

A Template for MLOps Governance

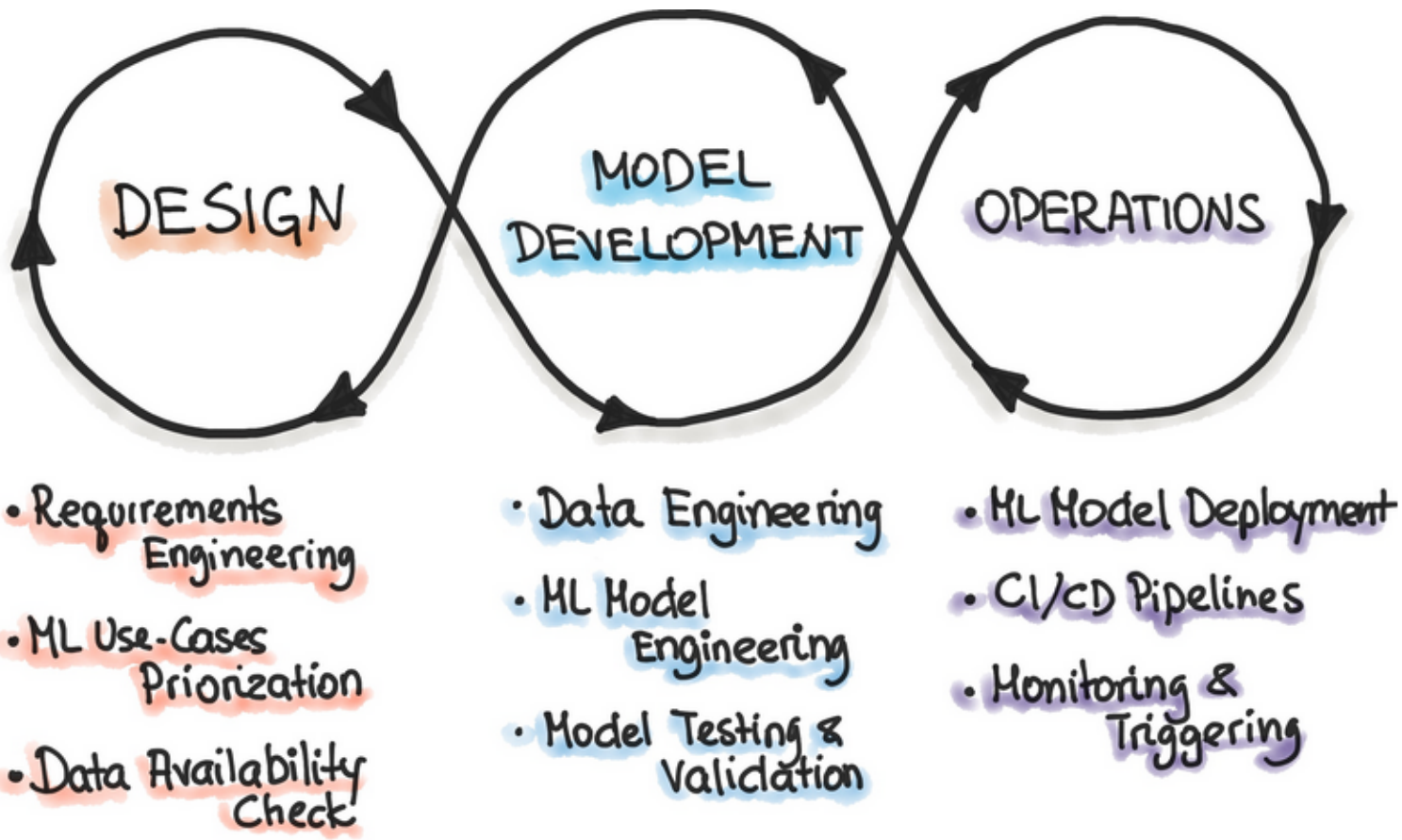
6. Select the tools for centralized governance management.
    - MLOps governance is much more effective if the overarching process is managed and tracked from one system. This system should:
        - Centralize the definition of the governance process
        - Enable tracking and enforcement of the complete governance process
        - Provide a single point of reference for the discovery of analytics projects
        - Enable collaboration between teams
    - The current workflow, project management, and MLOps tools can only partially support these objectives. A new category of ML governance tools is emerging to support this need directly and more fully.
7. Engage and educate.
    - Every individual needs to learn what they must do, when, and how.
    - This exercise will require considerable documentation, training, and time.
8. Monitor and refine.
    - Measuring success requires metrics and checks. It requires people to be tasked with monitoring and a way to address problems.
    - Monitoring and auditing can be time consuming, so look to automate metrics.
    - Monitoring changes in the governance landscape is essential. This might be regulatory, or it might be about public opinion.

Important Concepts in MLOps

Iterative-Incremental Process

MLOps • 2022

Iterative-Incremental Process



THREE PHASES

The complete MLOps process includes three broad phases of **“Designing the ML-powered application”, “ML Experimentation and Development”,** and **“ML Operations”**.

All three phases are **interconnected** and **influence** each other. For example, the design decision during the design stage will propagate into the experimentation phase and finally influence the deployment options during the final operations phase.

First Phase

DESIGNING THE ML-POWERED APPLICATION

Focus: business understanding, data understanding and designing the ML-powered software.

In this stage, we **design** the machine learning solution to solve its problem, and assess the further development of the project.

The design phase aims to inspect the **available data** that will be needed to train our model and to specify the f**unctional and non-functional requirements** of our ML model. We should use these requirements to design the architecture of the ML-application, establish the serving strategy, and create a test suite for the future ML model

Second Phase

ML EXPERIMENTATION AND DEVELOPMENT

Focus: to verifying the applicability of ML for our problem by implementing Proof-of-Concept for ML Model.

Here, we run iteratively different steps, such as identifying or polishing the suitable ML algorithm for our problem, data engineering, and model engineering.

The primary goal in this phase is to deliver a **stable quality ML model** that we will run in production.

Third Phase

ML OPERATIONS

Focus: to deliver the previously developed ML model in production by using established DevOps practices such as testing, versioning, continuous delivery, and monitoring.

Automation

MLOps • 2022

Automation

The level of automation of the Data, ML Model, and Code pipelines determines the **maturity** of the ML process. With increased maturity, the velocity for the training of new models is also increased.

The objective of an MLOps team is to **automate the deployment of ML models** into the core software system or as a service component.

Triggers for automated model training and deployment can be calendar events, messaging, monitoring events, as well as changes on data, model training code, and application code.

Automated testing helps discovering problems quickly and in early stages. This enables fast fixing of errors and learning from mistakes.

THREE LEVELS

See three levels of automation, starting from the initial level with manual model training and deployment, up to running both ML and CI/CD pipelines automatically.

There are 3 ways about implementing MLOps:

- **MLOps level 0 (Manual process)**
- **MLOps level 1 (ML pipeline automation)**
- **MLOps level 2 (CI/CD pipeline automation)**

MLOps level 0

This is a typical data science process, which is performed at the beginning of implementing ML. An entirely manual ML workflow and the data-scientist-driven process might be enough if your models are rarely changed or trained.

CHARACTERISTICS

- **Manual, script-driven, and interactive process**: every step is manual, including data analysis, data preparation, model training, and validation.
- **Disconnect between ML and operations**: the process separates data scientists who create the model, and engineers who serve the model as a prediction service.
- **Infrequent release iterations**: the assumption is that your data science team manages a few models that don’t change frequently
- **No Continuous Integration (CI)**: because few implementation changes are assumed, you ignore CI.
- **No Continuous Deployment (CD)**: because there aren’t frequent model version deployments.
- Deployment refers to the prediction service.
- Lack of active performance monitoring: the process doesn’t track or log model predictions and actions.

MLOps level 0

CHALLENGES

To address the challenges of this manual process, it’s good to use MLOps practices for CI/CD and CT. By deploying an ML training pipeline, you can enable CT, and you can set up a CI/CD system to rapidly test, build, and deploy new implementations of the ML pipeline

TOOLS

The common way to process is to use **Rapid Application Development (RAD) tools**, such as Jupyter Notebooks.

MLOps level 1

The goal of MLOps level 1 is to perform continuous training (CT) of the model by automating the ML pipeline. This way, you achieve continuous delivery of model prediction service.

Whenever new data is available, the process of **model retraining is triggered**. This level of automation also includes **data and model validation** steps.

CHARACTERISTICS

- **Rapid experiment**: ML experiment steps are orchestrated and done automatically.
- **CT of the model in production**: the model is automatically trained in production, using fresh data based on live pipeline triggers.
- **Continuous delivery of models**: the model deployment step, which serves the trained and validated model as a prediction service for online predictions, is automated.
- **Pipeline deployment**: deploy a whole training pipeline, which automatically and recurrently runs to serve the trained model as the prediction service.

MLOps level 1

ADDITIONAL COMPONENTS

- **Data and model validation**: the pipeline expects new, live data to produce a new model version that's trained on the new data.
- **Feature store**: a feature store is a centralized repository where you standardize the definition, storage, and access of features for training and serving.
- **Metadata management**: information about each execution of the ML pipeline is recorded in order to help with data and artifacts lineage, reproducibility, and comparisons. It also helps you debug errors and anomalies
- **ML pipeline triggers**: you can automate ML production pipelines to retrain models with new data.

CHALLENGES

This setup is suitable when you deploy new models based on new data, rather than based on new ML ideas.

If you manage many ML pipelines in production, you need a CI/CD setup to automate the build, test, and deployment of ML pipelines.

MLOps level 2

Introduce a CI/CD system to **perform fast and reliable ML model deployments in production**.
Now **automatically build, test, and deploy** the Data, ML Model, and the ML training pipeline components.

This MLOps setup includes the following components:

- Source control
- Test and build services
- Deployment services
- Model registry
- Feature store
- ML metadata store
- ML pipeline orchestrator.

MLOps level 2

CHARACTERISTICS

- **Development and experimentation**: you iteratively try out new ML algorithms and new modeling where the experiment steps are orchestrated. The output of this stage is the source code of the ML pipeline steps, which are then pushed to a source repository.
- **Pipeline continuous integration**: you build source code and run various tests. The outputs of this stage are pipeline components.
- **Pipeline continuous delivery**: you deploy the artifacts produced by the CI stage to the target environment. The output of this stage is a deployed pipeline with the new implementation of the model.
- **Automated triggering**: the pipeline is automatically executed in production based on a schedule or in response to a trigger. The output of this stage is a newly trained model that is pushed to the model registry.
- **Model continuous delivery**: you serve the trained model as a prediction service for the predictions. The output of this stage is a deployed model prediction service.
- **Monitoring**: you collect statistics on model performance based on live data. The output of this stage is a trigger to execute the pipeline or to execute a new experiment cycle.

Continuous Deployment

MLOps • 2022

Continuous Deployment

For Model deployment, we first specify the “ML assets” as

- ML model,
- its parameters and
- hyperparameters,
- training scripts,
- training and testing data.

We are interested in the identity, components, versioning, and dependencies of these **ML artifacts**. The target destination for an ML artifact may be a (micro-) service or some infrastructure components.

A deployment service provides **orchestration, logging, monitoring, and notification** to ensure that the ML models, code and data artifacts are stable.

ML engineering

MLOps is an ML engineering culture that includes some practices.

CONTINUOUS INTEGRATION (CI)

extends the **testing** and **validating** code and components by adding testing and validating data and models.

CONTINUOUS DELIVERY (CD)

concerns with delivery of an **ML training** pipeline that automatically deploys another the ML model prediction service.

CONTINUOUS TRAINING (CT)

is unique to ML systems property, which **automatically retrains** ML models for re-deployment.

CONTINUOUS MONITORING (CM)

concerns with **monitoring production data** and **model performance metrics,** which are bound to business metrics.

Versioning

MLOps • 2022

Versioning

The goal of the versioning is to treat ML training scripts, ML models and data sets by tracking ML models and data sets with version control systems.
With data scientists building, testing, and iterating on several versions of models, they need to be able to keep all the versions straight.

When ML model and data changes:

- ML models can be **retrained** based upon **new training data**.
- Models may be **retrained** based upon **new training approaches**.
- Models may be **self-learning**.
- Models may **degrade over time**.
- Models may be **deployed** in **new applications**.
- Models may be subject **to attack** and require **revision**.
- Models can be quickly rolled back to a **previous serving version**.
- Corporate or government compliance may require **audit or investigation** on both ML model or data, hence we need **access to all versions** of the productionized ML model.
- Data may reside across **multiple systems**.
- Data may only be able to reside in **restricted jurisdictions**.
- Data **storage** may **not be immutable**.
- Data **ownership** may be a factor.

Versioning

Analogously to the best practices for developing reliable software systems, every **ML model specification** (ML training code that creates an ML model) should go through a **code review phase**.
Furthermore, every ML model specification should be **versioned** in a VCS to make the training of ML models **auditable** and **reproducible**.

See: https://www.inovex.de/de/blog/machine-learning-model-management/

Experiments Tracking

MLOps • 2022

Experiments Tracking

In contrast to the traditional software development process, in ML development, **multiple experiments** on model training can be executed **in parallel** before making the decision what model will be promoted to production.

The experimentation during ML development might have the following scenario:

- One way to track multiple experiments is to **use different (Git-) branches**, each dedicated to the separate experiment.
- The output of each branch is **a trained model**.
- Depending on the **selected metric**, the trained ML models are compared with each other and the **appropriate model** is selected.

Tools

Such low friction branching is fully supported by the tool DVC, which is an **extension of Git** and an **open-source version control system** for machine learning projects.

Another popular tool for ML experiments tracking is the Weights and Biases (wandb) library, which **automatically tracks the hyperparameters and metrics** of the experiments.

TO DO:
-> https://dvc.org/
-> https://wandb.ai/site

“ML Test Score” System

MLOps • 2022

“ML Test Score” System

The “ML Test Score” measures the overall readiness of the ML system for production.

The final ML Test Score is computed as follows:

- For each test, half a point is awarded for executing the **test manually**, with the results documented and distributed.
- A full point is awarded if the there is a system in place to run that **test automatically** on a repeated basis.
- Sum the score of each of the four sections individually: **Data Tests, Model Tests, ML Infrastructure Tests, and Monitoring**.
- The final ML Test Score is computed by taking the minimum of the scores aggregated for each of the sections: Data Tests, Model Tests, ML Infrastructure Tests, and Monitoring.

The following table provides the interpretation ranges:

Points	Description
0	More of the research project than a productionized system.
(0,1]	Not totally untested, but it is worth considering the possibility of serious holes in reliability.
(1,2]	There has been first pass at basic productionization, but additional investment may be needed.
(2,3]	Reasonably tested, but it is possible that more of those tests and procedures may be automated.
(3,5]	Strong level of automated testing and monitoring.
>5	Exceptional level of automated testing and monitoring.

ML-based Software Delivery Metrics

MLOps • 2022

ML-based Software Delivery Metrics

There are four key metrics to measure and improve ones ML-based software delivery: Deployment Frequency, Lead Time for Changes, Mean Time To Restore, and Change Fail Percentage.
These are the same for capture the effectivenes of the software development and delivery of elite/high performing organisations.

DEPLOYMENT FREQUENCY

How often does your organization deploy code to production or release it to end-users?

MLOps
ML Model Deployment Frequency depends on

1. Model retraining requirements. Two aspects: Model decay metric, New data availability.
2. The level of automation of the deployment process, which might range between *manual deployment* and *fully automated CI/CD pipeline*.

LEAD TIME FOR CHANGES

How long does it take to go from code committed to code successfully running in production?
MLOps
ML Model Lead Time for Changes depends on

1. Duration of the explorative phase in Data Science in order to finalize the ML model for deployment/serving.
2. Duration of the ML model training.
3. The number and duration of manual steps during the deployment process.

ML-based Software Delivery Metrics

MEAN TIME TO RESTORE (MTTR)

How long does it generally take to restore service when a service incident or a defect that impacts users occurs?

MLOps

ML Model MTTR depends on the number and duration of manually performed model debugging, and model deployment steps. In case, when the ML model should be retrained, then MTTR also depends on the duration of the ML model training. Alternatively, MTTR refers to the duration of the rollback of the ML model to the previous version.

CHANGE FAILURE RATE

What percentage of changes to production or released to users result in degraded service and subsequently require remediation?

MLOps

ML Model Change Failure Rate can be expressed in the difference of the currently deployed ML model performance metrics to the previous model's metrics, such as Precision, Recall, F-1, accuracy, AUC, ROC, false positives, etc. ML Model Change Failure Rate is also related to A/B testing.

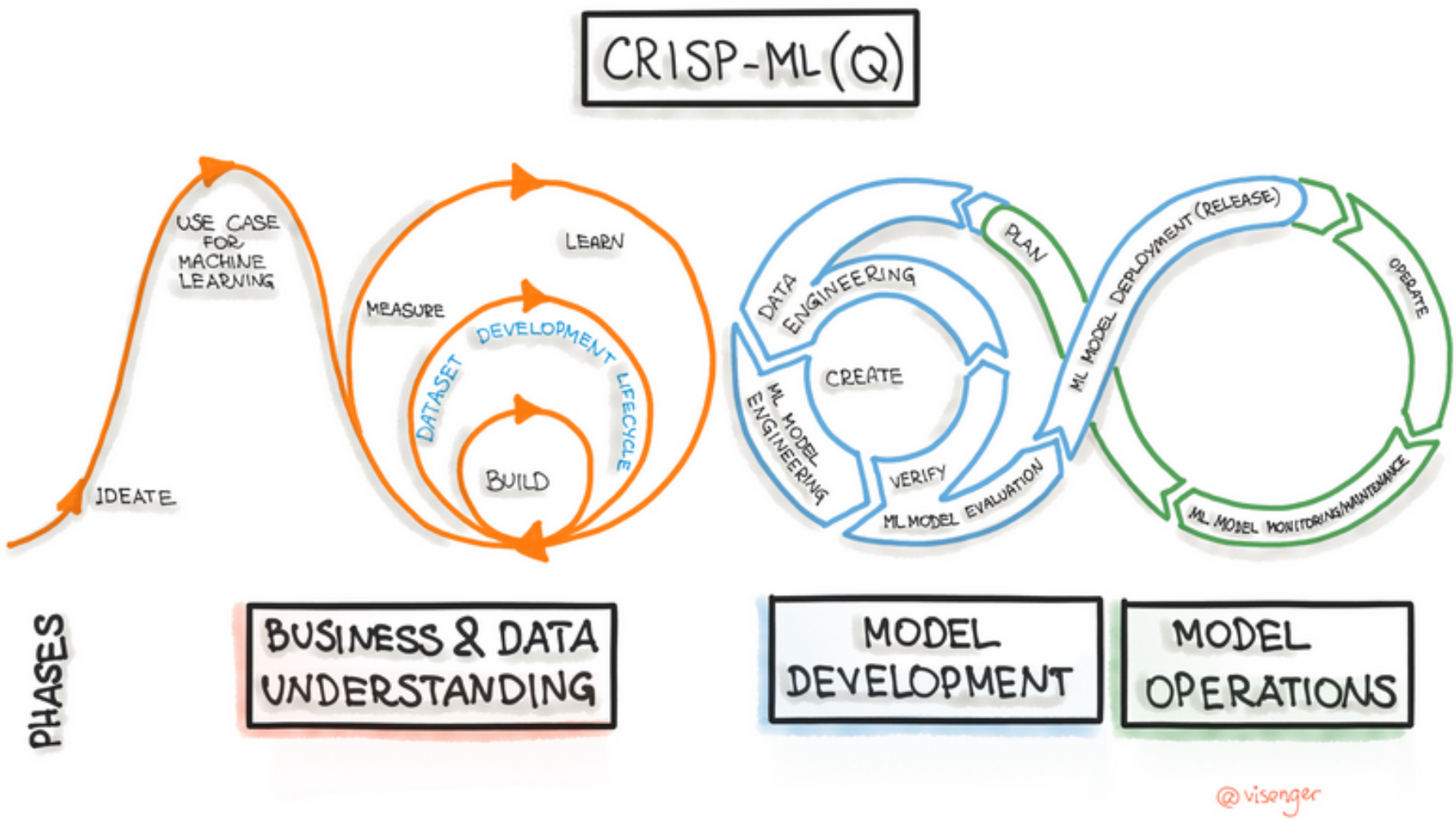
CRISP-ML(Q)

MLOps • 2022

CRISP-ML(Q)

To guide ML practitioners through the development life cycle, the **Cross-Industry Standard Process for the development of Machine Learning applications with Quality assurance** methodology (CRISP-ML(Q)) was recently proposed.

CRISP-ML(Q) is a systematic process model for machine learning software development that creates an awareness of possible risks and emphasizes quality assurance to diminish these risks to ensure the ML project's success.



CRISP-ML(Q)

The CRISP-ML(Q) process model describes six phases:

1. Business and Data Understanding
2. Data Engineering (Data Preparation)
3. Machine Learning Model Engineering
4. Quality Assurance for Machine Learning Applications
5. Deployment
6. Monitoring and Maintenance.

For each phase of the process model, the quality assurance approach in CRISP-ML(Q) requires:

- the definition of **requirements and constraints** (e.g., performance, data quality requirements, model robustness, etc.)
- instantiation of the **specific tasks** (e.g., ML algorithm selection, model training, etc.)
- specification of **risks** that might negatively impact the efficiency and success of the ML application (e.g., bias, overfitting, lack of reproducibility, etc.)
- **quality assurance methods** to mitigate risks when these risks need to be diminished (e.g., cross-validation, documenting process and results, etc.).

Business and Data Understanding

Identify the scope of the ML application, the success criteria, and a data quality verification. The goal of this first phase is to ensure the feasibility of the project.

We gather success criteria along with business, machine learning, and economic success criteria during this phase. These criteria are required to be measurable. Therefore, defining clear and measurable **Key Performance Indicators** (**KPI**) such as “time savings per user and session” is required.

Data collection and data quality verification are essential to achieving business goals. Therefore, one crucial requirement is the documentation of the statistical properties of data and the data generating process.

Tool: Machine Learning Canvas, Business Model Canvas

Data Engineering (Data Preparation)

Prepare data for the following modeling phase.
Task: Data selection, data cleaning, feature engineering, data standardization.

DATA SELECTION
Select data by discarding samples that do not satisfy data quality requirements. At this point, we also might tackle the problem of unbalanced classes by applying over-sampling or under-sampling strategies.

DAT CLEANING
Perform error detection and error correction steps for the available data. Adding unit testing for data will mitigate the risk of error propagation to the next phase.

FEATURE ENGINEERING
Depending on the machine learning task, we might need to perform feature engineering and data augmentation activities. For example, such methods include one-hot encoding, clustering, or discretization of continuous attributes.

DATA STANDARDIZATION
Denotes the process of unifying the ML tools' input data to avoid the risk of erroneous data. Finally, the normalization task will mitigate the risk of bias to features on larger scales.

Machine Learning Model Engineering

Specify one or several machine learning models to be deployed in the production. The translation to the ML task depends on the business problem that we are trying to solve.

Therefore we should ensure that the method and the results of the modeling phase are reproducible by collecting the model training method's metadata. Typically we collect the following metadata: algorithm, training, validation and testing data set, hyper-parameters, and runtime environment description.

Documenting trained models increases the transparency and explainability in ML projects.

Finally, we package the ML workflow in a pipeline to create repeatable model training during the modeling phase.

Tasks: model selection, model specialization, and model training.

Tools: Model Cards Toolkit. (https://arxiv.org/pdf/1810.03993.pdf)

Evaluating Machine Learning Models

Also known as **offline testing**.

During this phase, the performance of the trained model needs to be validated on a test set.

Additionally, the model robustness should be assessed using noisy or wrong input data.

Finally, the model deployment decision should be met automatically based on success criteria or manually by domain and ML experts.

Deployment

Denotes a process of the ML model **integration into the existing software system**.

After succeeding in the evaluation step in the ML development life cycle, the ML model is graduated to be deployed in the (pre-) **production environment**.
Deployment approaches will differ depending on the use case and the training and prediction modus, either batch or online.

Tasks: inference hardware definition, model evaluation in a production environment (online testing, e.g., A/B tests), providing user acceptance and usability testing, providing a fall-back plan for model outages, and setting up the deployment strategy to roll out the new model gradually (e.g. canary or green/blue deployment).

Monitoring and Maintenance

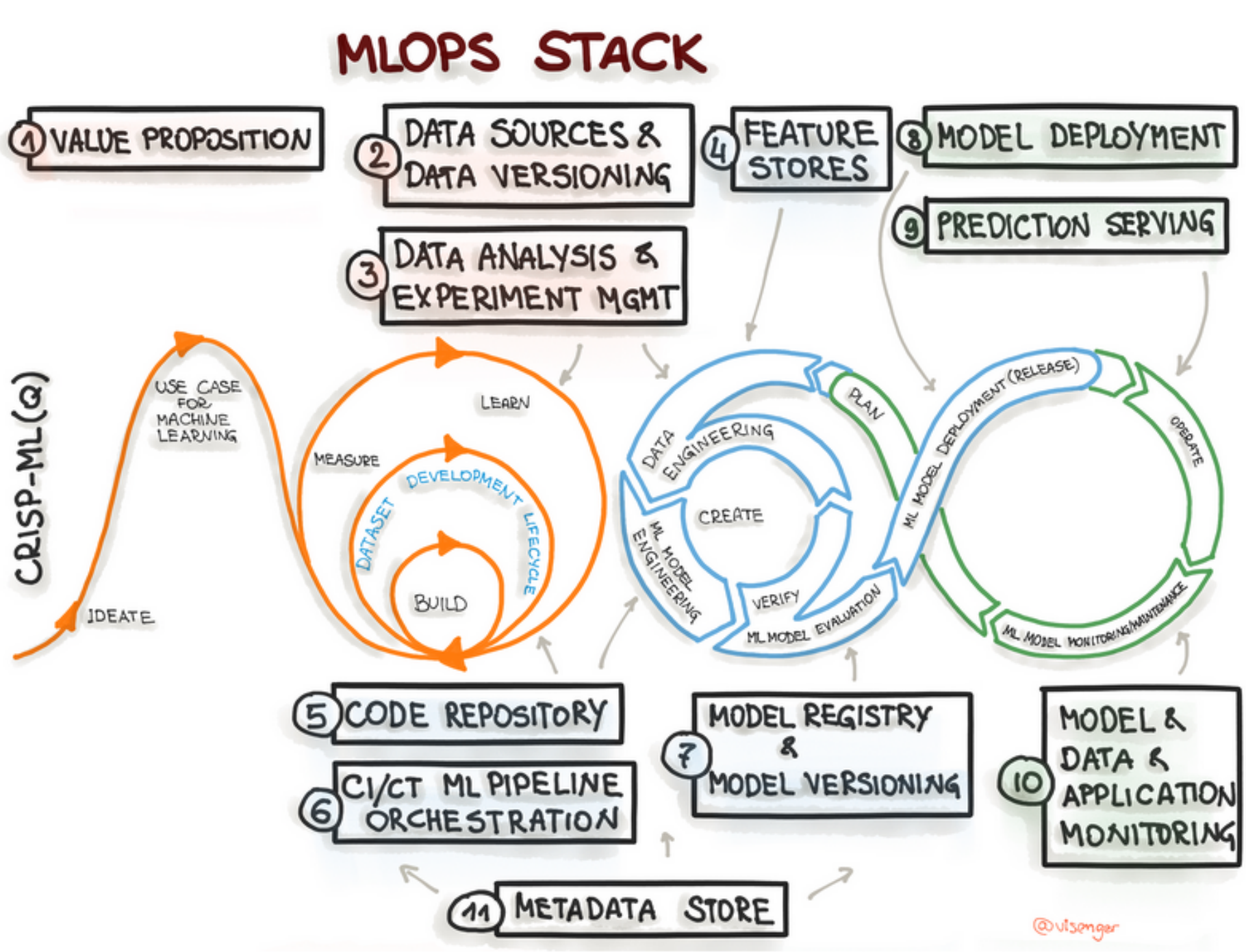
Once the ML model has been put into production, it is essential to monitor its performance and maintain it.
When an ML model performs on real-world data, the main risk is the "**model staleness**" effect when the performance of the ML model drops.

Therefore, the best practice to prevent the model performance drop is to perform the monitoring task when the model performance is **continuously evaluated** to decide whether the model needs to be re-trained. This is known as the **Continued Model Evaluation pattern**.

The decision from the monitoring task leads to the second task - updating the ML model.

MLOps stack

Mapping the CRISP-ML(Q) process model to the MLOps stack.



Infrastructure

MLOps • 2022

Building vs Buying vs Hybrid MLOps Infrastructure

Cloud computing companies have invested hundreds of billions of dollars in infrastructure and management.

Now, should you build or buy your infrastructure? Maybe you should go hybrid?

Tech companies that want to survive long-term usually have in-house teams and build custom solutions. If they have the skills, knowledge, and tools to tackle complex problems, there's nothing wrong with that approach.
But there are other factors that are worth taking into account, like:

- time and effort
- human resources
- time to profit
- opportunity cost.

Factors

TIME AND EFFORT

Data scientists often spend their time building solutions to add to their existing infrastructure in order to complete projects. 65% of their time was **spent on engineering heavy**, non-data science tasks such as tracking, monitoring, configuration, compute resource management, serving infrastructure, feature extraction, and model deployment.

This wasted time is often referred to as '**hidden technical debt**', and is a common bottleneck for machine learning teams.
Building an in-house solution, or maintaining an underperforming solution can take **from 6 months to 1 year**. Even once you've built a functioning infrastructure, just to maintain the infrastructure and keep it up-to-date with the latest technology requires lifecycle management and a dedicated team.

HUMAN RESOURCES

Operationalizing machine learning requires a lot of **engineering**. For a smooth machine learning workflow, each **data science team** must have an **operations team** that understands the unique requirements of deploying machine learning models.

Investing in an end-to-end MLOps platform, these processes can be completely **automated**, making it easier for operations teams to focus on optimizing their infrastructure.

Factors

COST

Having a **dedicated operations team** to manage models can be expensive on its own. If you want to scale your experiments and deployments, you'd need to hire **more engineers** to manage this process. It's a major investment, and a slow process to find the right team.

After calculating all the different costs associated with hiring and onboarding an entire team of engineers, your return on investment drops, which brings us to our next factor.

TIME TO PROFIT

It can take **over a year** to build a functioning machine learning infrastructure. It can take even longer to build a data pipeline that can produce value for your organization.
Companies like **Uber, Netflix, and Facebook** have dedicated years and massive engineering efforts to scale and maintain their machine learning platforms to stay competitive.

For most companies, an investment like this is not possible, and also not necessary. The machine learning landscape has matured since Uber, Netflix and Facebook originally built their in-house solutions. There are more pre-built solutions that offer all you need out-of-the-box, at a fraction of the cost.
Instead of building all the infrastructure necessary to make their models operational, data scientists can focus on research and experimentation to deliver the best model for their business problem.

Factors

OPPORTUNITY COST

As mentioned above, one survey shows that 65% of a data scientist's time is spent on non-data science tasks.

Using an **MLOps platform automates** technical tasks and reduces DevOps bottlenecks. Data scientists can spend their time doing more of what they were hired to do – deliver high-impact models – while the cloud provider takes care of the rest.

Adopting an **end-to-end MLOps platform** has a considerable competitive advantage that allows your machine learning development to scale massively.

MLOps Infrastructure

BUILD

Building your own platform and infrastructure will take more and more of your **focus and attention** as demand increases. The time that could be spent on model R&D and data collection will be taken by **infrastructure management**. This isn't great unless it's part of your core business (if you're a **cloud service provider, PaaS or IaaS**).

BUY

Buying a fully managed platform gives you **great flexibility and scalability**, but then you're faced with **compliance, regulations, and security issues**.

HYBRID

Hybrid cloud infrastructure for MLOps is the best of both worlds, but it poses **unique challenges**, so it's up to you to decide if it fits your business model.

Hybrid MLOps Infrastructure

Some companies have been entrusted with **private & sensitive data**. It can't leave their servers because in the chance of a small vulnerability, the ripple effect would be catastrophic. This is where **Hybrid cloud infrastructure for MLOp**s comes in.

Today's infrastructure is a mix of **cloud** and **on-prem**.

Cloud infrastructure is increasingly popular, but it's still rare to find a large company that has completely abandoned on-premise infrastructure (most of them for obvious reasons, like sensitive data).

Hybrid cloud adoption are growing.
Hybrid cloud environments add an additional layer of complexity that makes managing IT even more challenging.

The vast majority of **cloud stakeholders** (96%) face challenges managing both on-prem and cloud infrastructure.

Cloud Centers of Excellence (CCoEs)

A Cloud Center of Excellence (CCoE) is a **cross-functional team** of people responsible for developing and managing the **cloud strategy, governance, and best practices** that the rest of the organization can leverage to transform your business using the cloud.

The CCoE leads the organization as a whole in **cloud adoption, migration, and operations**. It may also be called a **Cloud Competency Center, Cloud Capability Center, or Cloud Knowledge Center**.

CCoE adoption is increasing

IN COMPANIES

CCoEs can take on many forms.

For some companies it is an official team with a formal reporting structure and clear responsibilities for decision making. Other companies have teams that manage aspects of cloud strategy, but with limited responsibilities.

Many companies, especially those early in their adoption of cloud, have an informal approach to CCoE. A typical approach includes team members who have not been assigned specific responsibilities. However, within the group, there may be experts that are generally acknowledged as owning the functions.

References

References

Link

- https://devops.com/what-is-mlops-dataops-and-why-do-they-matter/
- https://neptune.ai/blog/mlops
- https://ml-ops.org/content
- ttps://click.cloudcheckr.com/rs/222-ENM-584/images/CloudCheckr-White-Paper-The-Cloud-Infrastructure-Report-2020.pdf
- https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning#top_of_page

Book

- Introducing MLOps How to Scale Machine Learning in the Enterprise (Mark Treveil, Nicolas Omont, Clément Stenac etc.)