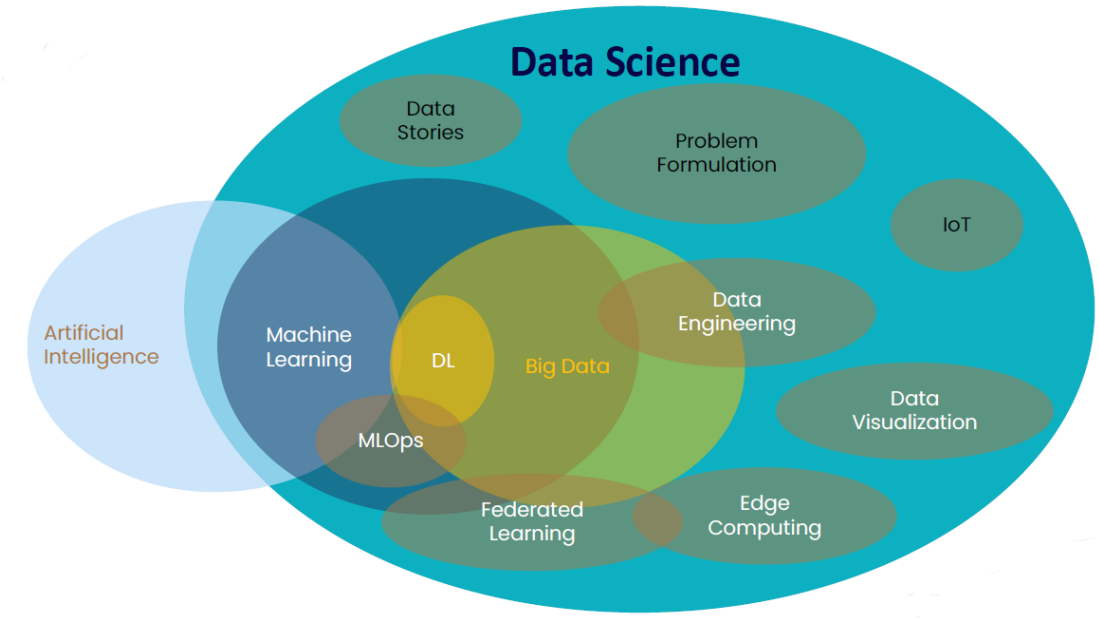
MLOps: [M]achine [L]earning [Op]eration[s]

Introduction

MLOps, short for Machine Learning Operations, is an emerging practice and implementing tools focused on streamlining and automating the lifecycle management of machine learning models. It borrows various principles from cross-domains, meaning, the continuous integration and continuous delivery/deployment (CI/CD) from DevOps, orchestrating data pipelines (DataOps) from Data Engineering domain, and model-building from Machine Learning domain to enable efficient and reliable deployment, monitoring, and maintenance of ML systems at scale.



Venn diagram of different domains under data science, and AI

Motivation for MLOps: Unlocking the potential of Machine Learning system

In the last ten years, machine learning has revolutionised many industries, enabling automation, streamlining processes, and providing insights which has flourished the companies. But as industries has started incorporating machine learning into their operations at a large scale, it has raised many complexities, few of which are how to guarantee that a trained model is deployed in a production environment without any problems? How to manage the lifecycle of numerous models to ensure they remain precise and up-to-date? How to assure the close-knit collaboration between data scientists, data engineers, business managers, software engineers, and other stakeholders to ensure that hand-offs occur efficiently, from data preparation and model training to model deployment

and monitoring? Are the machine learning models are effectively offering the best business decisions? MLOps addresses all that and then more.

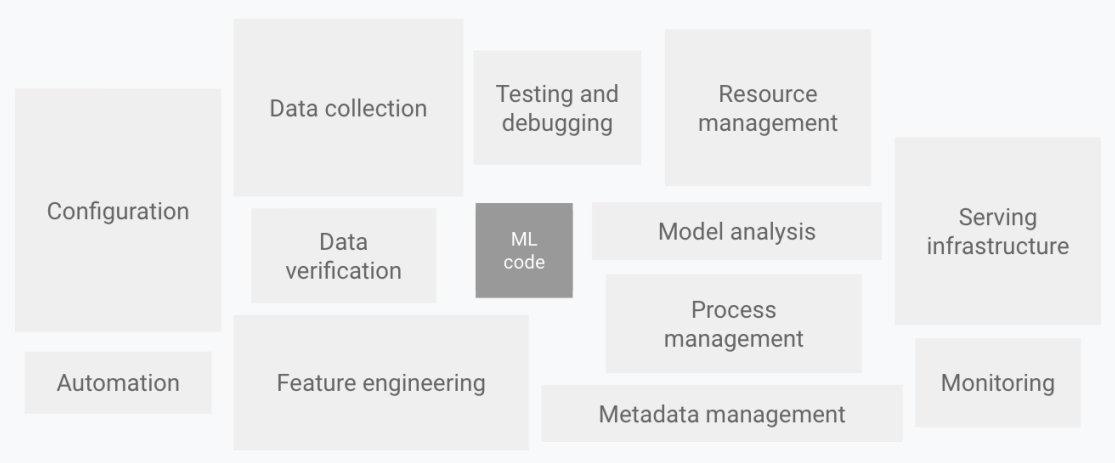


Image Source: Elements for ML systems

Teams at Google have been doing a lot of research on the technical challenges that come with building ML-based systems. A NeurIPS paper on hidden technical Debt in ML systems shows you developing models is just a very small part of the whole process. There are many other processes, configurations, and tools that are to be integrated into the system. To streamline this entire system, we have this new Machine learning engineering culture. The system involves everyone from the higher management with minimal technical skills to Data Scientists to DevOps and ML Engineers.

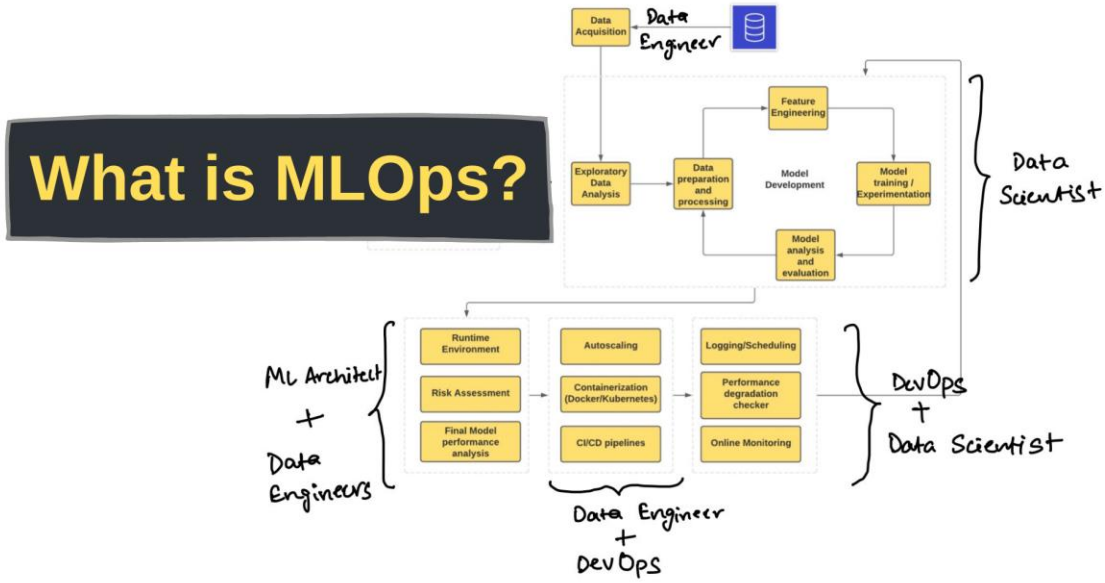


Image Source: What is MLOps?

Differences Between Traditional Software systems and Machine Learning systems

In the realm of software development, traditional software systems and Machine Learning (ML) systems differ fundamentally despite some overlaps. They differ in their approach to problem-solving, development process, implemented and managed. Here are some key differences:

1. **Problem Scope**:
   - **Traditional Software**: The problem, solution, and success metrics are clear from the beginning. For example, creating a marketplace for home rentals has a defined goal and success is a working platform [1].
   - **ML Systems**: The problem and solution may be unclear at the start and can change as the project progresses. Success is often measured by the optimization of model metrics like accuracy or precision [1].
2. **Quality Dependency**:
   - **Traditional Software**: Quality depends on the code. Changes in data do not impact the functionality unless new business rules are needed [2].
   - **ML Systems**: Quality depends on input data and hyperparameter tuning. New data may require retraining the model to maintain or improve performance [2].
3. **Continuous Evolution**:
   - **Traditional Software**: A programmer writes explicit rules or instructions for the computer to follow [3]. Once developed and tested, traditional software typically does not need to evolve unless there is a change in requirements, a need for optimization, or detected bugs. Updates are predictable and often scheduled.
   - **ML Systems**: Algorithms learn from data without explicit programming. Models improve over time with feedback [3]. ML models in ML systems often require continuous updating and retraining. This is due to factors such as evolving data, changing patterns, and the nonstationary nature of many real-world scenarios (i.e., the underlying distribution of the problem changes over time). *Example*: A model that predicts stock prices will need regular updates due to the dynamic nature of financial markets.
4. **Data Dependency**:
   - **Traditional Software**: Relies less on data and more on predetermined logic [3]. The functionality is mostly independent of external data. While it processes data, its core behavior does not change on the basis of the data it has processed in the past.
   - **ML Systems**: Heavily reliant on data to draw decision logic [4]. The effectiveness of ML models is intrinsically tied to the quality, quantity, and relevance of the training data. An ML model's behavior is shaped by its past data, making it crucial to ensure high-quality data for training. *Example*: A chatbot trained on a limited and poor quality dataset might give irrelevant

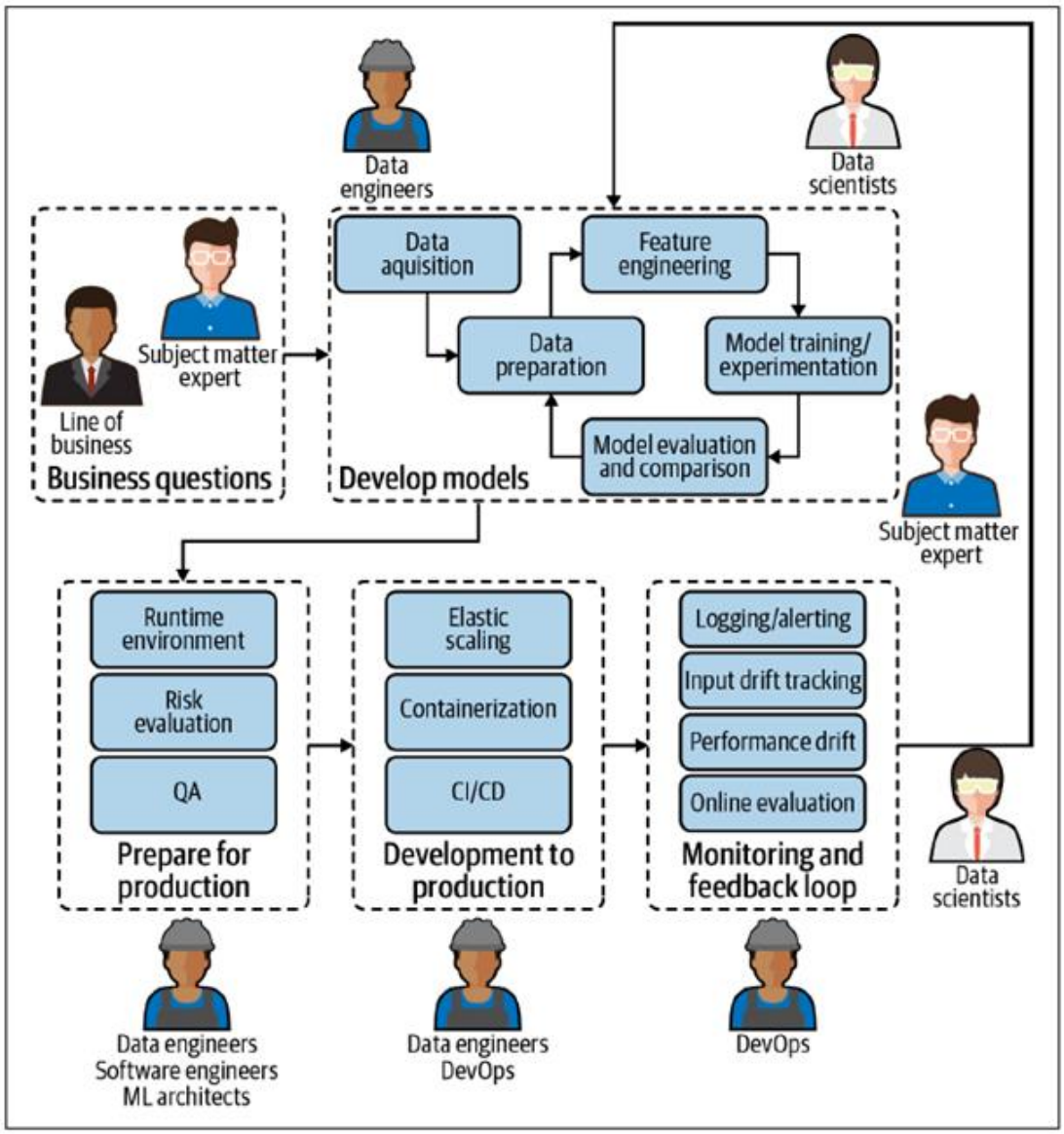
answers, whereas one trained on a diverse and comprehensive dataset can provide more accurate and context-aware responses.

5. **Validation and Testing**:
   - **Traditional Software**: The functionality is mostly independent of external data. While it processes data, its core behavior does not change on the basis of the data it has processed in the past.
   - **ML Systems**: The effectiveness of ML models is intrinsically tied to the quality, quantity, and relevance of the training data. An ML model's behavior is shaped by its past data, making it crucial to ensure high-quality data for training. *Example*: A chatbot trained on a limited and poor quality dataset might give irrelevant answers, whereas one trained on a diverse and comprehensive dataset can provide more accurate and context-aware responses.
6. **Outcome Predictability**:
   - **Traditional Software**: Exhibits deterministic behavior with predictable outcomes based on the given input[3]. For a given input, they will always produce the same output, provided the system state and configuration remain unchanged. Their behavior is explicitly programmed, meaning that developers define a set of rules and conditions that dictate the software's response to various inputs.
   - **ML Systems**: Outcomes are probabilistic and can vary depending on the data it was trained on [3]. They generate outputs based on patterns identified during training. Consequently, they might produce different outcomes for slightly varying inputs, and there is inherent uncertainty in their predictions.

In summary, traditional software operates on a set of clear rules and is less dependent on data, whereas ML systems are designed to learn from data, requiring continuous improvement and adaptation to evolving data. This fundamental difference impacts how problems are defined, how success is measured, and how the systems are developed and maintained.

Components of a ML system

ML systems are intricate, with many workflows intertwined, many stakeholders collaborating to convert raw data into useful predictions or insights. It is essential to comprehend these components to create and execute reliable ML solutions.



The realistic picture of an ML model life cycle

Image Source: What Is MLOps? by Mark Treveil, Lynn Heidmann

1. **Data Acquisition:**
   The process of acquiring data from different sources, importing them, read and make them accessible for further workflows in the ML system. The key aspects are:

- **Sources:** Data can be ingested from a variety of sources, including databases, data lakes, APIs, and streaming platforms.
- **Formats:** Data can come in multiple formats such as CSV, Parquet, JSON, images, and more.
- **Real-time vs. Batch:** While real-time ingestion processes data as it's generated, batch ingestion involves processing data at intervals.

2. **Data Preparation** At this stage, data from the previous stage is transformed from raw data into a format suitable for training ML models. The key aspects include:

- **Cleaning:** Handling missing values, removing outliers, and correcting erroneous data.
- **Feature Engineering:** Creating new features from existing ones to improve model performance.
- **Normalization & Scaling:** Adjusting the feature scales to ensure consistent model training.
- **Encoding:** Converting categorical variables into numerical formats for the model.
- **Data Augmentation (for image data):** Creating new training samples by applying transformations (e.g., rotation, flipping) to existing images.

3. **Model Training** The core process where data is fed into algorithms to develop predictive models. The key aspects include:

- **Algorithm Selection:** Choosing the right machine learning or deep learning algorithm for the problem.
- **Hyperparameter Tuning:** Optimizing model parameters to enhance performance.
- **Training Loops:** Iteratively updating the model weights based on the prediction error.
- **Validation:** Using a separate data set to avoid overfitting and ensure the model generalizes well.

4. **Model Evaluation** The process of assessing the performance of an ML model.The key aspects include:

- **Metrics:** Depending on the problem type, different metrics are used (e.g., accuracy, precision, recall, F1 score for classification; MSE or RMSE for regression).
- **Confusion Matrix:** A table used to evaluate classification models by comparing actual vs. predicted classes.
- **Cross-Validation:** A technique where the training set is split multiple times to assess the robustness of the model.
- **AUC-ROC:** A grpahical representation for the performance of the classification model

5. **Model Serving** Deploying trained models to make predictions on new data involves several key aspects:

- **Deployment Platforms**: Models can be deployed on various platforms, including cloud services, on-premises servers, and edge devices.
- **API Endpoints**: Models are exposed as services with API endpoints. These endpoints allow querying the model for predictions.
- **Batch vs. Real-Time Serving**:
  - **Batch Serving**: Provides predictions on bulk data all at once.
  - **Real-Time Serving**: Offers immediate predictions for each input.

6. **Model Monitoring** Continuously observing and analyzing deployed models' behavior and performance is crucial. Here are the key aspects of model monitoring:

- **Performance Monitoring**: Track accuracy, latency, and other relevant metrics after deployment. Ensure the model performs as expected.

- **Logging**: Keep records of prediction requests, responses, and errors. Useful for debugging and auditing.
- **Alerts**: Set up automatic notifications for anomalies or performance degradations. Prompt action when issues arise.
- **Model Drift Detection**: Monitor data distribution over time. Detect changes that may impact model performance.

Key Components of MLOps

1. Continuous Integration & Continuous Deployment (CI/CD)

CI/CD pipelines automate the process of building, testing, and deploying ML models. This ensures that changes to models or code are quickly integrated and deployed into production environments.

Image Source: Amazon Web Services

2. Model Versioning & Registry

Model versioning and registry systems track and manage different versions of ML models, making it easy to rollback changes if necessary and ensuring reproducibility of experiments.

Image Source: Medium

3. Model Monitoring & Observability

Monitoring and observability tools track the performance of deployed models in real-time, detecting drift, anomalies, and degradation in model accuracy, and triggering alerts for corrective actions.

Image Source: Databricks

Steps in the MLOps process

Where MLOps sees the biggest benefit is in the iterative orchestration of tasks. While data scientists are reviewing new data sources, engineers are adjusting ML configurations. Making simultaneous adjustments in real-time vastly reduces the time spent on improvements.

Benefits of MLOps

Reducing Time-to-Market

- ***Rapid Prototyping***: MLOps allows teams to quickly iterate over models, helping them move from conceptualization to deployment swiftly.
- ***Automation***: Automated pipelines reduce manual errors and speed up repetitive tasks like data pre-processing, model validation, and deployment.
- ***Feedback Loops***: Faster model training and deployment cycles enable organizations to get faster feedback, aiding in timely model adjustments.

Ensuring Model Reliability and Robustness

- ***Reproducibility***: MLOps emphasizes versioning not just for code but also for data and model configurations, ensuring that experiments are reproducible.
- ***Continuous Evaluation***: With continuous monitoring, models are regularly evaluated against new data to detect and address perfor- mance degradation.
- ***Alert Mechanisms***: MLOps tools can raise instant alerts if a model's performance dips below a threshold or if anomalies are detected, ensuring timely interventions.

Simplifying Model Management

- ***Centralized Model Registry***: MLOps practises often include main- taining a centralised model registry where all models, their ver- sions, and metadata are stored, simplifying model discovery and management.
- ***Rollbacks and Canary Deployments***: If a newly deployed model fails, MLOps enables swift rollbacks to the previous stable version. Canary deployments allow one to test new models on a fraction of the traffic before a full-scale rollout.
- ***Scalability***: MLOps infrastructure is designed to scale, ensuring that the models can handle varying loads, from a few requests per sec- ond to thousands.

Ensuring Regulatory and Ethical Compliance

- ***Audit Trails***: MLOps tools log every action, change, and decision made in the ML workflow, enabling traceability and satisfying reg- ulatory requirements.
- ***Bias Detection and Fairness Checks***: MLOps emphasises continuous monitoring for biases and unfair decision-making, helping organi- sations uphold ethical standards and avoid reputational risks.
- ***Data Security and Privacy***: With the increasing importance of data in ML workflows, MLOps ensures that data are stored, accessed, and used securely, respecting privacy norms and regulations.

In conclusion, MLOps plays a crucial role in enabling organizations to operationalize and scale their machine learning initiatives effectively. By integrating best practices from

software engineering and data science, MLOps fosters collaboration, agility, and reliability in the development and deployment of ML solutions.

How generative AI is evolving MLOps

The release of OpenAI's ChatGPT sparked interests in AI capabilities across industries and disciplines. This technology, known as generative AI, has the capability to write software code, create images and produce a variety of data types, as well as further develop the MLOps process [7].

Generative AI is a type of deep-learning model that takes raw data, processes it and "learns" to generate probable outputs. In other words, the AI model uses a simplified representation of the training data to create a new work that's similar, but not identical, to the original data. For example, by analyzing the language used by Shakespeare, a user can prompt a generative AI model to create a Shakespeare-like sonnet on a given topic to create an entirely new work.

Generative AI relies on foundation models to create a scalable process. As AI has evolved, data scientists have acknowledged that building AI models takes a lot of data, energy and time, from compiling, labeling and processing data sets the models use to "learn" to the energy is takes to process the data and iteratively train the models. Foundation models aim to solve this problem. A foundation model takes a massive quantity of data and using self-supervised learning and transfer learning can take that data to create models for a wide range of tasks [7].

This advancement in AI means that data sets aren't task specific—the model can apply information it's learned about one situation to another. Engineers are now using foundation models to create the training models for MLOps processes faster. They simply take the foundation model and fine-tune it using their own data, versus taking their data and building a model from scratch [7].

Citations

1. [What is MLOps? Machine Learning Operations Explained] - Source: FreeCodeCamp
2. [How machine learning differs from traditional-software] - Source: Medium
3. [difference between machine learning traditional software]- Source: vitalflux
4. [Traditional Programming vs Machine Learning]- Source: insightsoftware
5. [Test & Evaluation Best Practices for Machine Learning-Enabled Systems]- Source: arxiv
6. [Introduction to MLOps]- Source: Medium:ogre51
7. [MLOps and the evolution of data science]- Source: IBM