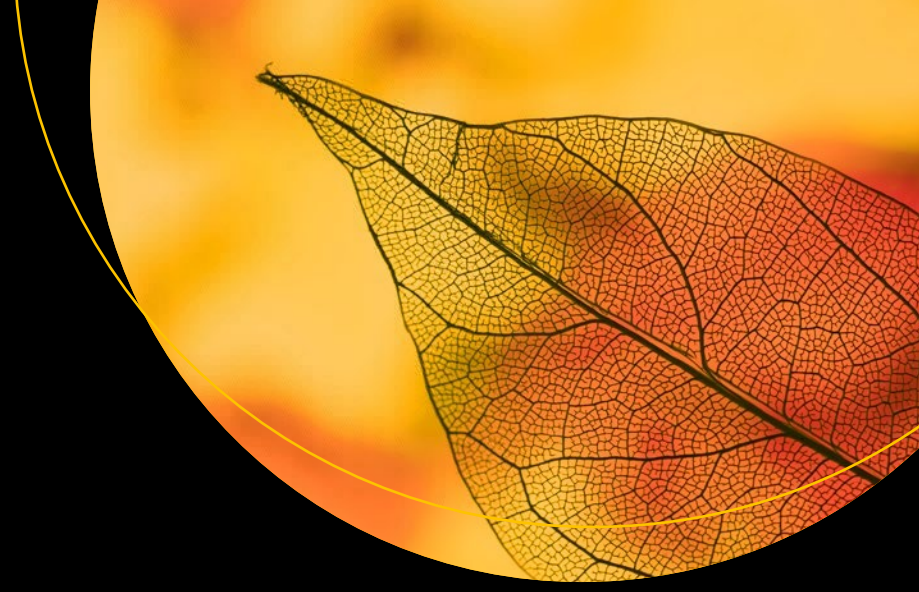

# MLOps with Ray

## Best Practices and Strategies for Adopting Machine Learning Operations

—

Hien Luu
Max Pumperla
Zhe Zhang

Apress®

# MLOps with Ray

## Best Practices and Strategies for Adopting Machine Learning Operations

**Hien Luu**
**Max Pumperla**
**Zhe Zhang**

Apress®

*MLOps with Ray: Best Practices and Strategies for Adopting Machine Learning Operations*

Hien Luu
Santa Clara, CA, USA

Max Pumperla
Bad Segeberg, Germany

Zhe Zhang
Sunnyvale, CA, USA

ISBN-13 (pbk): 979-8-8688-0375-8
ISBN-13 (electronic): 979-8-8688-0376-5
https://doi.org/10.1007/979-8-8688-0376-5

Copyright © 2024 by Hien Luu, Max Pumperla and Zhe Zhang

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: Laura Berendson
Coordinating Editor: Gryffin Winkler

Cover designed by eStudioCalamar
Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub (https://github.com/Apress). For more detailed information, please visit https://www.apress.com/gp/services/source-code.

If disposing of this product, please recycle the paper

# Table of Contents

# About the Authors

**Hien Luu** is a passionate AI/ML engineering leader who has been leading the Machine Learning platform at DoorDash since 2020. Hien focuses on developing robust and scalable AI/ML infrastructure for real-world applications. He is the author of the book *Beginning Apache Spark 3* and a speaker at conferences such as MLOps World, QCon (SF, NY, London), GHC 2022, Data+AI Summit, and more.

**Max Pumperla** is a data science professor and software engineer located in Hamburg, Germany. He's an active open source contributor, maintainer of several Python packages, and author of machine learning books. He currently works as a software engineer at Anyscale. As the head of product research at Pathmind Inc., he was developing reinforcement learning solutions for industrial applications at scale using Ray RLlib, Serve, and Tune. Max has been a core developer of DL4J at Skymind and helped grow and extend the Keras ecosystem.

**Zhe Zhang** has been leading the Ray Engineering team at Anyscale since 2020. Before that, he was at LinkedIn managing the Big Data/AI Compute team (providing Hadoop/Spark/TensorFlow as services). He has been working on open source for about a decade. Zhe is a committer and PMC member of Apache Hadoop; he is the lead author of the HDFS Erasure Coding feature, which is a critical part of Apache Hadoop 3.0. In 2020, Zhe was elected as a Member of the Apache Software Foundation.

# About the Technical Reviewer

**Ashutosh Parida** is an accomplished leader in Artificial Intelligence and Machine Learning (AI/ML) and currently serving as Assistant Vice President, heading AI/ML product development at DeHaat, a leading AgriTech startup in India. With over a decade of experience in data science, his expertise spans various domains, including vision, NLU, recommendation engines, and forecasting.

With a bachelor's degree in Computer Science and Engineering from IIIT Hyderabad and a career spanning 18 years at global technology leaders like Oracle, Samsung, Akamai, and Qualcomm, Ashutosh has been site lead for multiple projects and has launched products serving millions of users. He also has open source contributions to his credit.

Stay connected with Ashutosh on LinkedIn to stay updated on his pioneering work and gain valuable industry insights: `linkedin.com/in/ashutoshparida`.

CHAPTER 1

# Introduction to MLOps

Machine learning (ML) has proven to be a very powerful tool to learn and extract patterns from data. The ability to generate, store, and process a large amount of data, and easily access computing power in the last decade has contributed to many advancements in the ML field, such as image recognition, language translation, and large language models (LLMs), that is, BERT, DALLE, ChatGPT, and more.

ML has finally graduated from the academia lab and has been embraced with both open arms by the business world to help with solving real-world business problems and transforming industries by improving customer experience, reducing cost, improving business efficiency, and ultimately increasing their competitive advantage. According to McKinsey's "The state of AI in 2021"[1] report, the findings from the survey indicate that AI/ML adoption is continuing its steady rise across many companies in many regions of the world. One of the reasons for this rise is due to the impact that AI has on the business bottom line.

It is no longer a question whether AI/ML can deliver business values to organizations across the industries. A more pressing question on an AI/ML leadership team's mind nowadays is how can their organizations most efficiently integrate AI/ML into their business processes or products to deliver that value. To do so, they need the ability to operationalize ML in an iterative, consistent, effective, safe, and predictable manner.

The data about the number of ML project failures from various sources such as Gartner[2] and VentureBeat[3] suggests that operationalizing ML is a complex endeavor that requires a set of standardized processes and technology capabilities for building, deploying, and operationalizing ML models efficiently and quickly. Welcome to MLOps!

[1] Global Survey: The State of AI Adoption 2021 – www.mckinsey.com/capabilities/quantumblack/our-insights/global-survey-the-state-of-ai-in-2021

[2] Our Top Data and Analytics Predicts for 2019 – https://blogs.gartner.com/andrew_white/2019/01/03/our-top-data-and-analytics-predicts-for-2019/

[3] Why do 87% of data science projects never make it into production? – https://venturebeat.com/ai/why-do-87-of-data-science-projects-never-make-it-into-production/



© Hien Luu, Max Pumperla and Zhe Zhang 2024
H. Luu et al., *MLOps with Ray*, https://doi.org/10.1007/979-8-8688-0376-5_1

# MLOps Overview

Software engineering projects are designed to bring values to a company, organization, or a team. The realization of the intended values will only start when the development software artifacts are successfully deployed into production. The sooner this happens the better. Companies around the world have adopted DevOps as methodologies to reliably develop and deploy large-scale software to production by minimizing the gap between development and operations and promoting collaboration, communication, and knowledge sharing. DevOps is widely adopted in the industry because it helps with fast, frequent, and reliable software releases by emphasizing automation with continuous integration, continuous delivery, and continuous deployment.

Similarly, ML projects are designed to add certain value to a company, organization, or a team. The ROI of the ML projects will only start when the ML artifacts, the ML model and features, are deployed to production and properly monitored. The achievement of the ROI for ML projects can vary due to factors such as the project's complexity, data quality, and specific goals. Organizations may start seeing initial returns during the early stages of deployment, especially if the ML models are addressing specific pain points like reducing churn in customer subscriptions. The full ROI is typically realized when models are successfully integrated into operational processes, delivering sustained and measurable value over an extended period. Achieving this often takes time, as models may need refinement, optimization, and continuous monitoring to adapt to changing conditions.

How are ML projects different from the software engineering projects? What's unique about ML projects? Can DevOps methodologies help with ML projects? Let's examine these questions to deeply understand MLOps and the benefits it provides.

**Note** DevOps has been widely adopted across many organizations that are developing software as methodologies to improve software quality and reliability while reducing time to market for software engineering initiatives. It is both a paradigm shift to address social and technical issues in organizations engaged in software development and a continuous process of automation across the software development process.

The cornerstone of DevOps is a continuous process including continuous development, integration, deployment, and monitoring that is designed for fast, frequent, and reliable software releases.

The DevOps mindset requires software engineers to care not only about what they develop, but also care about the software deployment and operations.

## ML Projects

Similar to software engineering projects, ML projects have their own development lifecycle. However, their lifecycle is a highly iterative one due to the scientific nature of the ML model training process, which requires experimentation and has a large dependency on the quality of the data used to train the ML algorithm. As a result, the ML development process is not a linear one, like the standard software engineering development lifecycle, but rather it is cyclical in that it demands iteration, tuning, and improvement, as depicted in Figure 1-1.

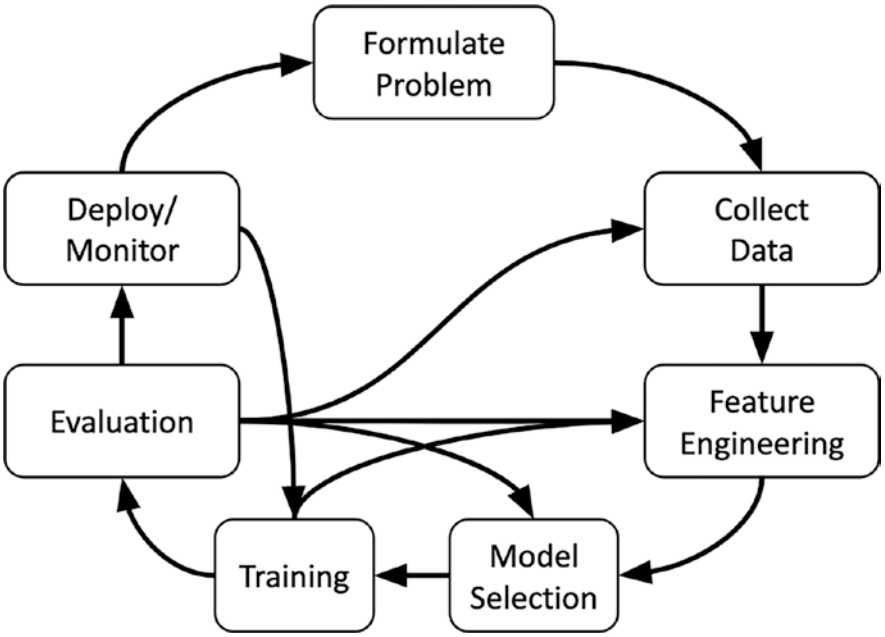


***Figure 1-1.*** *ML development process*

ML projects often are initiated to help fulfill certain business or product initiatives with measurable outcomes. As the first step of the ML project, it is important to formulate the problem with clear goals and objectives before going through the other steps in the clockwise manner. It is not unusual, but rather expected and necessary that data scientists will need to go back to previous steps, such as to collect additional data or change the way features are generated, if the model evaluation step indicates the ML model doesn't perform at the expected level or when the experiment results convince data scientists to change the current approach or continue to fine-tune the current approach.

Successful ML projects are the ones where data scientists are able to make progress throughout the ML development lifecycle as quickly as possible and able to go through the development loop as many times as needed in order to apply the insights from previous experiments to fine-tune their approach, the needed data, and ultimately produce the most optimal ML models. The ultimate goal is to produce the most optimal ML models that can perform predictions with new data at the accuracy level to meet business goals or objectives.

Although the ML development lifecycle is cyclical and appears complex, it can be simplified into five phases:

- Data collection and preparation
- Feature engineering
- Model training
- Model deployment
- Model monitoring

## ML Project Inputs and Artifacts

In traditional software engineering projects, software engineers write code to develop logic or algorithms to meet the given specifications to produce the output based on the input, as depicted in Figure 1-2.


***Figure 1-2.*** *Software projects*

In ML projects, data scientists spend a lot of time and focus on these two main activities: feature engineering and model development. The following paragraphs are meant to briefly describe these two activities and more importantly to call out the ML project inputs and artifacts.

The quantity and quality of the ML features have a huge contribution to the performance of the ML models. Data scientists typically spend a large portion of their time in collecting and analyzing the collected data, and then write code to transform the data into ML features to train the ML models with, as depicted in Figure 1-3.


***Figure 1-3.*** *Model training in ML projects*

Once data scientists are reasonably happy with the generated ML features, they then proceed to the ML model training step, which involves writing code to train ML models using the generated features, an ML algorithm, and a set of tuning parameters. This step often requires exploration and experimentation to evaluate, fine-tune, and iterate on the ML model to improve its performance. If the model evaluation results are suboptimal, data scientists might need to go back to the feature engineering step to collect more or different sets of data, or select a different ML algorithm, etc.

The main ML artifacts from the above activities are as follows:

- The data used to generate features
- The logic for generating features from the collected data
- The code and parameters for training the ML algorithm
- The produced ML model

Oftentimes, the ML models will need to be retrained because new business requirements emerge, or additional new data sources or ML libraries are available, or the performance of ML models has started to degrade, and more. Hence, it is important to version control or manage the ML artifacts listed previously and depicted in Figure 1-4.


***Figure 1-4.*** *ML artifacts*

ML projects are different traditional software engineering projects and present many unique challenges and can be summarized in the following points:

- Training ML models requires historical data, and hence ML projects involve more data-related activities, such as collecting and labeling data and analyzing and visualizing input data to better understand its statistical characteristics.
- Model development is a high iterative process that requires exploration and experimentation.
- The ML model performance can degrade over time as the statistical characteristics of the new data drift or change in relative to the statistical characteristics of the historical data.

- ML projects typically require more collaboration between data scientists, data engineers, ML engineers, and domain experts, as the success of the project often depends on a combination of technical expertise and domain knowledge.

It has been said that MLOps is to machine learning, as DevOps is to software engineering. The main goal of MLOps is to help companies around the world to accelerate their ML projects to production in a repeatable, consistent, and efficient manner by prescribing a set of best practices around both the technical and non-technical elements.

# MLOps: The Missing Element

As more and more companies around the world recognize the power of AI/ML and allocate budget to invest in applying AI/ML to add business values, to increase their competitiveness, and more, they rightly so want to see the ROI of their investment in ML projects. Their ROI can only start when the ML models are deployed to production and integrate into their products or business processes. What can ultimately help with bringing ML models to production efficiently and quickly? From the collective wisdom and learning, the ML practitioner community and industry have come up with the answer, which is called MLOps.

In this section, we will examine some of the common challenges and pitfalls in operationalizing ML projects, and then discuss at the high level how MLOps can help with those with the goal of improving ML project success.

## Operationalize ML Project Challenges

As reported by numerous surveys from Gartners and NewVantage Partners, many organizations are not as successful as they would like to be when it comes to getting the ROI from their investment in ML projects due to the challenges of operationalizing ML models in production quickly, efficiently, and consistently. According to a survey conducted by NewVantage Partners in 2020,[4] only about 15% of leading organizations have deployed AI capabilities into production at any scale.

[4] AI Stats News: Only 14.6% Of Firms Have Deployed AI Capabilities In Production –www.forbes.com/sites/gilpress/2020/01/13/ai-stats-news-only-146-of-firms-have-deployed-ai-capabilities-in-production/

It is fair to say most organizations now understand operationalizing ML projects is challenging and it is not the same as operationalizing traditional software projects.

The section will highlight a few common challenges in ML projects and shed some lights about how those challenges come about. These challenges we believe are a subset of the ones that are contributing to ML project failure. Challenges around a lack of talent or unclear business objectives are beyond the scope of this book.

## Applying Machine Learning

In the real-world ML project, it is generally well understood that the central piece of these projects is applying ML to enable data-driven decisions and products.

ML, as a discipline, is highly experimental and iterative in nature, especially compared to typical software engineering.

Applying ML involves training a model on a dataset and then evaluating its performance on the same dataset as well as a separate dataset. Rarely the first version of the model will meet the expected performance, and therefore, this process can be repeated multiple times, where each iteration potentially uses a different ML algorithm or architecture, hyperparameter settings, and feature engineering techniques.

At the beginning of applying ML, it is very challenging to know the exact combination of the aforementioned parts that will lead to an ML model that performs well. Therefore, exploration, experimentation, and iteration are necessary parts of finding the best combination and quickly pruning out the bad ones.

Similar to other scientific endeavors, ML experimentation and iteration require the ability to record the experiment input, the approach, results, and more. This will facilitate and speed up the process of analyzing the result by comparing multiple experimentation runs to determine the next step.

Additionally, ML is still an evolving field, with new techniques, approaches, and ML libraries being developed all the time. As a result, ML practitioners must be open to experiment with and take advantage of these new changes to improve the ML model performance.

The name of the game here is velocity. If ML practitioners are bogged down not having the processes or tools to explore, experiment, and iterate easily and quickly, then getting ML models to production might not be feasible or will take a long time to realize the ROI of the ML project investment.

## Garbage In, Garbage Out

The classic saying of "Garbage In, Garbage Out" in computing is about how problematic input data will produce problematic outputs, which is especially relevant to ML, given that the model training process relies heavily on the quality of the input data.

ML model training is the process where the labeled training data is fed into the ML algorithm to learn its pattern. It is widely known the ML model performance is only as good as the quality of that data. Some of the well-known ML practitioners recently started advocating for the data-centric AI approach to further highlight the benefits of high-quality training data. For more details about this approach, see the notes below.

In addition to data quality, other data-related aspects that have large influence on the ML model performance are data freshness and changes to data statistical properties.

If there is a lack of data infrastructure, data engineering rigor, and personnel support around these data-related aspects, then that will have a negative impact on the ML model performance, which ultimately slows down the path to bring ML models to production.

---

**Note** Model-centric AI vs. data-centric AI – same goal, but different approach

Model-centric AI is an approach and mindset in improving ML model performance by focusing on tweaking the hyperparameters or changing the model architecture or algorithms until the desired metrics are achieved. This approach is what traditionally the industry has been practicing.

Data-centric AI has the same goal as the model-centric AI, but it takes a different approach by holding the hyperparameters and the model architecture and algorithm fixed while applying error analysis driven data iteration to improve the model performance. Formally, data-centric AI is defined as the discipline of systematically engineering the data used to build an AI system, according to the Data-centric AI Resource Hub website[5]. This approach was introduced and advocated by Andrew Ng in his "A Chat w/ Andrew on MLOps: From Model-centric to Data-centric AI" video[6].

---

[5] Data-centric AI Resource Hub – https://datacentricai.org/

[6] A Chat with Andrew on MLOps: From Model-centric to Data-centric AI –www.youtube.com/watch?v=06-AZXmwHjo

## In the Beginning

Traditionally, ML has been approached from a perspective of individual scientific experiments which are predominantly carried out in isolation by data scientists. Data scientists are knowledgeable and trained in the ML field, and are mainly tasked with model training-related tasks.

As a result, data scientists tend to focus less on areas that are peripheral to ML model training area, such as spending time on automating the data pipelines, focusing on developing robust and high-quality code, automating end-to-end model training pipelines, and integrating ML models into the data-driven products in production.

Enterprise ML projects are not about experimenting and developing ML models one time and moving on to other projects; they demand those software engineering-related activities to be automated, version controlled, monitored, reproducible, and more.

Without having data scientists shifting toward a more software engineering-centric mindset or having the necessary tooling, infrastructure, or processes to support data scientists with those software engineering-related activities, then there will be a negative impact on the road to operationalizing ML in your organization. In addition, there needs to be a culture shift toward a product-oriented mindset at the organization level.

## Team Sport

The end-to-end sequence of developing ML models and incorporating them into the data-driven decision products is a complex and interdisciplinary process, such as data engineering, machine learning, software engineering, and developer operations. Undoubtedly, it is a team sport, and thus responsibilities and ownership need to be clear and collaboration across teams is required to ensure those products are stable, are kept up to date, and more importantly continue to add value to the business.

A typical team sport consists of role and responsibilities, and given productionalizing ML is branded as a team sport, let's capture some of the typical roles and responsibilities across the activities in the ML development lifecycle.

Table 1-1 is meant to capture the core set of ML development activities and not meant to be comprehensive.

***Table 1-1.*** *ML development activity assignment*

| Activity | Role |
|---|---|
| Data preparation | Data engineer |
| Feature engineering | Data scientist |
| Model training | Data scientist |
| Model deployment | ML engineer |
| Model monitoring | ML engineer |

Some enterprises might include other roles into the ML operationalization process, such as business stakeholder or data governance officer.

For medium to large enterprises, each of those roles might be carried out by one or more people. For smaller enterprises or startups, two or more or all of those roles are carried by an individual.

A more elaborated depiction of the intersections of the various roles from the "Machine Learning Operation (MLOps): Overview, Definition and Architecture"[7] paper is depicted in Figure 1-5.

[7] Machine Learning Operations (MLOps): Overview, Definition, and Architecture – https://arxiv.org/ftp/arxiv/papers/2205/2205.02302.pdf

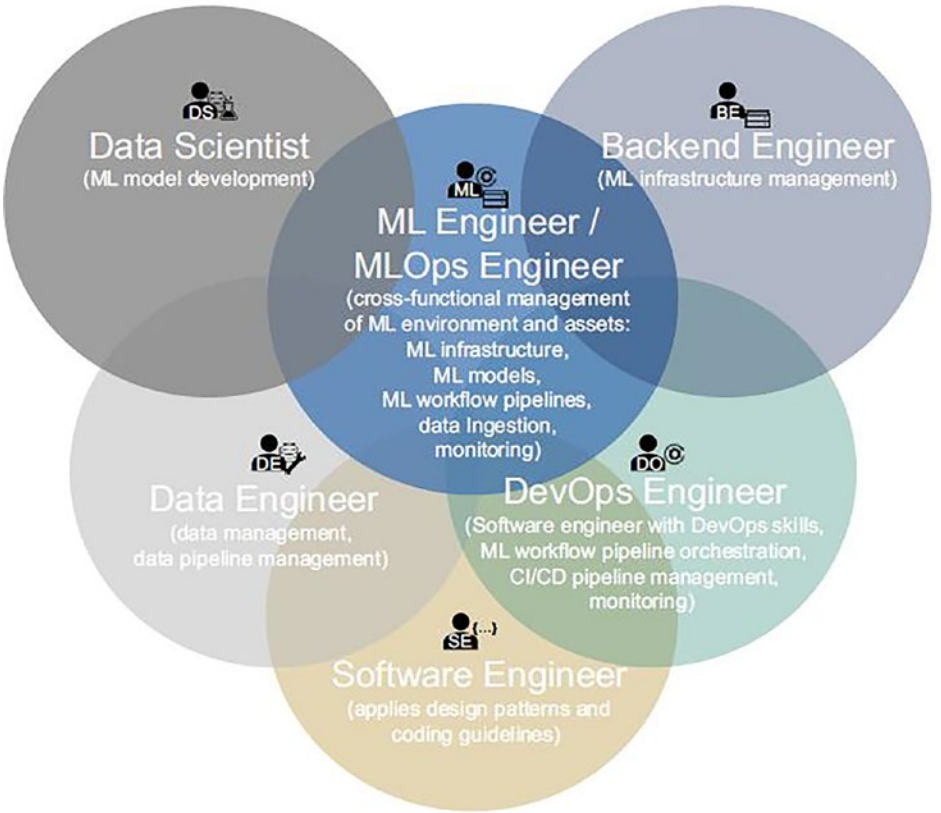


***Figure 1-5.*** *An elaborated depiction of the intersections of various roles in MLOps*

In order to be a winning team in a team sport, it needs to have all the needed personnel and a clear map of responsibilities, and a clear protocol around communication and coordination. Similarly, for enterprises to be successful at operationalizing ML, they need the necessary and sufficient talent that formed a multi-disciplinary team to perform those roles, and there need to be standard processes, clear communication protocols, and boundaries of ownership and responsibilities so all the teams involved are on the same page and aligned, fulfilling the expected deliverables and performing proper handoff at the right time.

## Summary of Challenges

It is reasonable to say that successfully operationalizing ML consistently, efficiently, and at scale is not an easy task; however, it is feasible and results are worthwhile.

ML has proven to be a disruptive technology that can help with reducing cost, improving operational efficiency, and increasing business bottom-line for those enterprises that are willing to invest in it.

This section summarizes the challenges listed above in the context of the following three key dimensions, which MLOps aims to address and will be described in the "The Promise" section: automation, reproducibility, and monitoring.

## Automation

As described above in the "Applying Machine Learning" section, it is a highly experimental and iterative process. This means that most if not all activities in the process of applying ML can greatly benefit from the automation. Manual process presents many challenges including error prone, time consuming, inconsistency, and not easily reproducible.

Automation leads to increased velocity because those activities can be repeated easily and consistently and enable data scientists to be able to go through the development lifecycle quicker.

As described in the "Garbage In, Garbage Out" section, the data-related activities are quite important to the ML model performance. Automation of the data-related activities such as running and monitoring data pipelines to ensure data quality and freshness will ultimately contribute to the optimal ML model performance.

## Reproducibility

Complex ML projects often require collaboration between multiple data scientists to discuss their hypothesis and validating ML model training experiments. To be effective at this, they need the ability to reproduce the experiments easily, and this requires the artifacts that were used in the previous experiments to be readily available and accessible.

It is very common for data scientists to iterate on an existing ML model to create a new one. Common reasons for the new ML model iterations include changing business requirements, new training data sources are available, customer behavior changes, and more. The ability to reproduce most of what was already done in the previous ML model version will greatly speed up the new iteration.

### Monitoring

It is often said that deploying ML models into production is only half the battle, and that continuously maintaining their performance afterward is the other half. This is because ML model performance in production can and often does degrade, which can have negative impacts on customers or the enterprise.

This degradation can occur for a variety of reasons, such as quality issues with the ML features used for predictions, changes in user behaviors, changes in the environment (such as the COVID pandemic), and more. Therefore, it is essential to continuously monitor the performance of deployed ML models and alert data scientists when the performance falls below a certain threshold.

Similar rigor and continuous monitoring should also be applied to the data pipelines that produce the data for training or generating features used during prediction time.

By actively monitoring the data, features, and performance of ML models, data scientists and ML engineers can identify issues early and take appropriate action to maintain the effectiveness of the models over time.

## MLOps: The Promise

Now that we gain a high-level understanding of the development lifecycle and inputs and artifacts of ML projects, as well as the common challenges in operationalizing ML models, let's examine what MLOps is and how it can help with addressing those challenges in the context of the three dimensions listed in the "Summary of Challenges" section.

A quick search on the Internet will review many slightly different interpretations of MLOps, as well as the definition. However, they all tend to converge toward a common goal and a set of themes.

To understand the essence of MLOps, we will peel back the layers of what it encompasses, similar to peeling an onion. MLOps consists of three fundamental layers: paradigm, engineering discipline, and principle. Each layer contributes to the ultimate goal of MLOps and will be explored in detail. By examining each layer individually and collectively, we will gain a comprehensive understanding of the principles and practices that underpin MLOps, as well as its significance in enabling efficient and effective management of the entire machine learning lifecycle. The MLOps onion is depicted in Figure 1-6.

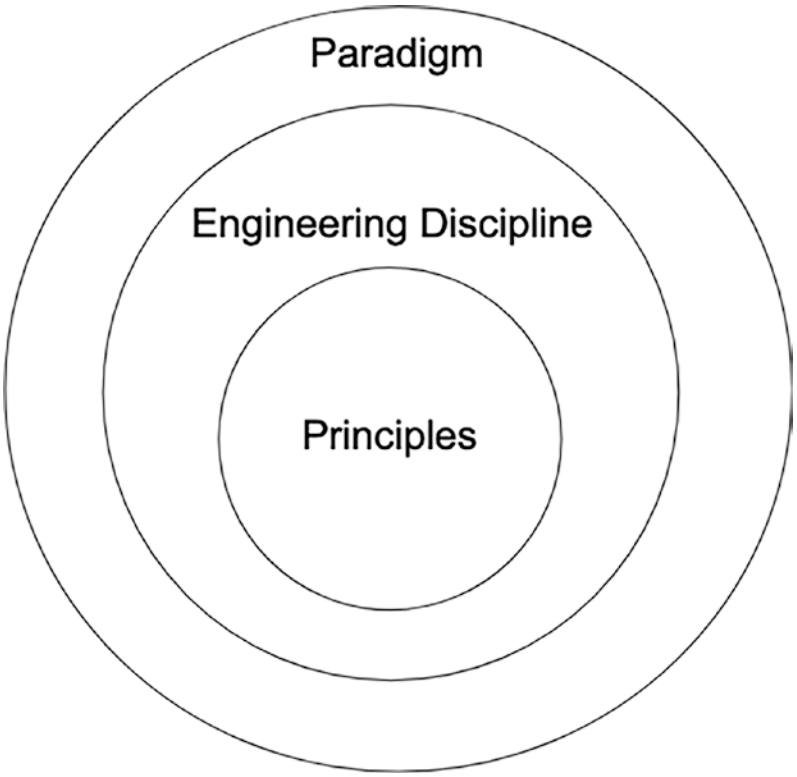


***Figure 1-6.*** *MLOps onion – paradigm, engineering discipline, principles*

## Paradigm

MLOps represents a paradigm shift in the way organizations approach machine learning. This paradigm recognizes that machine learning is not just a research or experimental activity but a critical component of business operations that must be managed with the same level of rigor and discipline as other technology systems.

Organizations that are at the forefront of reaping the benefits of their successful ML projects are those that have embraced this paradigm and mindset, treating ML models and artifacts as first-class software components. These organizations have also adopted an operational ML mindset, designing, building, and managing ML systems with a focus on reliability, scalability, and efficiency in line with MLOps principles.

The MLOps paradigm consists of a set of best practices, concepts, and a development culture. These aspects will be described in the "Engineering Discipline" and "Principle" sections.

## Engineering Discipline

From the rise of applying ML to business problems across many organizations around the world, MLOps emerged as a new engineering discipline that combines the engineering best practices and principles of three existing disciplines, namely, machine learning, data engineering, and devops, as depicted in Figure 1-7.

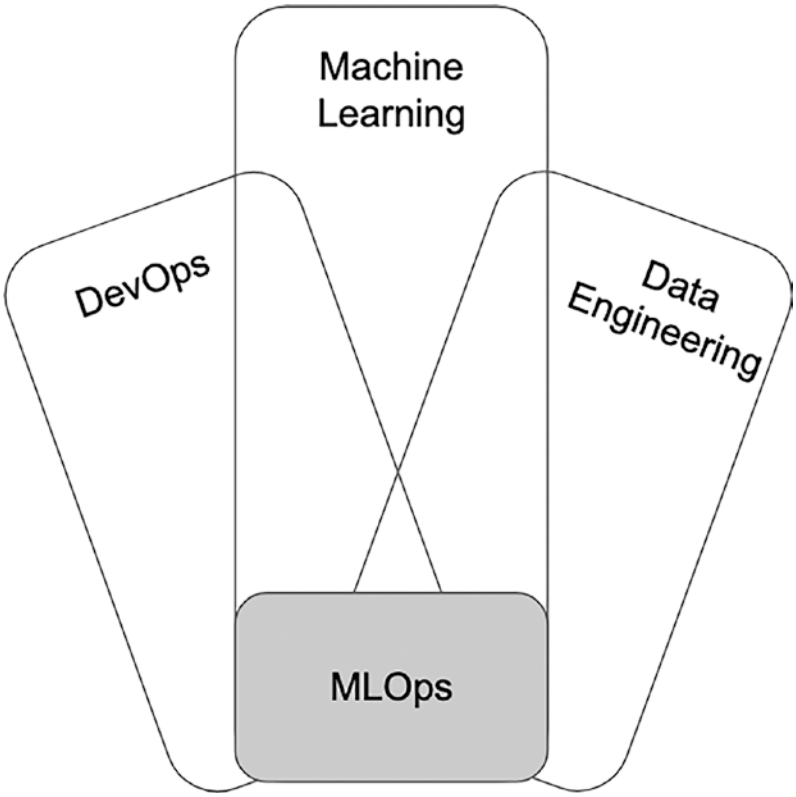


***Figure 1-7.*** *MLOps engineering discipline – intersection of three other disciplines*

MLOps engineering discipline involves the application of engineering principles and practices to the development, deployment, monitoring, and maintenance of machine learning models. Its main goal is to enable organizations to operationalize ML models in an efficient, high velocity, scalable, and maintainable manner. In other words, to reduce friction to get ML models from an idea into production and integrate with software systems in the shortest amount of time with as little risk as possible.

## Data Engineering

It is well understood that data is the lifeblood of AI/ML, and the quantity and quality of data will determine the level of performance that ML models can achieve.

Data engineering discipline brings the following main contributions to MLOps discipline:

- Lay the foundation to prepare data that will be used to train ML models.
- Take care of the framework and infrastructure for processing, storage, and consuming data from different data sources in various formats.
- Ensure the data that will be used to train ML models meets the quality standards and freshness requirements via automated, efficient, tested, monitored data pipelines.

## Machine Learning

At the heart of applying ML to enterprise business problems is the practice and application of ML techniques.

Machine learning discipline brings the following main contributions to MLOps discipline:

- Analyze and draw insights from the prepared data to determine the data statistical properties to ensure the right level of representation and fairness of the ML model training data.
- Determine and select the most appropriate combination of ML algorithm and tuning parameters to product ML models that will perform well on new data.
- Develop intuition about the ML model performance to iterate and optimize the ML model performance to meet business success metrics before deployment ML models to production.

## DevOps

MLOps leverages much of the best practices from well-known and trusted DevOps discipline. There are more artifacts in MLOps than in DevOps, and these additional ones add more complexity and challenges, and they must be treated accordingly in the context of DevOps.

DevOps discipline brings the following main contributions to MLOps discipline:

- Promote collaboration, communication, and knowledge sharing to close the gap between development and operations. Given MLOps is a bigger team sport than DevOps, this contribution is extremely applicable and critical to the MLOps success.
- Ensure automation with continuous integration, delivery, and continuous deployment to drive fast, frequent, and reliable ML model releases. This will help with accelerating the time it takes to go from ideas to ML models in production and maintain their expected performance with ML model retraining when necessary.
- Ensure continuous testing, quality assurance, continuous monitoring, logging, and feedback loops. Given ML model performance depends largely on the training data and new data used for prediction and data changes constantly, therefore continuous monitoring of the data statistical properties and model performance are key to ensuring ML models perform as expected and minimize the risks affecting user experiences or other negative impacts.

The best practices of the three engineering disciplines, data engineering, ML, and DevOps, are a great start; however, some adjustments and additions listed above will need to be incorporated into MLOps due to its unique and highly iterative and experimental nature, being a larger team sport, and artifacts include data, code, and model, which bring additional challenges and management. Below are a few of the notable ones:

- Testing in ML projects is more involved than testing traditional software systems. In addition to unit and integration tests, they also need data validation, trained model quality evaluation, and model validation.

- Deploying ML models involve a multi-step pipeline that includes automatically retrain and deploy model, and deploying features to online feature store on a regular basis.
- Monitoring ML models in production is a must due to the natural tendency that model performance will degrade. This requires regular and proactive tracking of both the summary statistics of the data used for predictions and the online model performance.
- Continuous ML model training is unique to ML systems. This additional practice is the ability to automatically retrain and serve the models with guardrails when there are any changes to the data, code, and ML models.

## Principles

In the "Engineering Discipline" section above, there were numerous discussions about some of the best practices in the three disciplines that MLOps is built upon. This section aims to codify some of those and include a few additional ones into a set of principles that any MLOps adoption should consider.

These principles are meant to guide the MLOps practice in an organization. As for the level of rigor of or focus on each principle, that will vary depending on that organization's AI strategy, objective, use cases, and culture.

### Automation

Automation refers to the process of removing manual processes and humans involved as much as possible and investing in process and tooling to carry out the critical steps in ML development lifecycle. This includes execute, build, test, train, deploy ML artifacts, such data, code, and ML models.

Some of the ML development activities need to run repeatedly on a certain cadence, such as data pipelines, feature generation pipelines, model training pipelines, and more. These activities will benefit the most from automation.

The need for automation increases

- As the number of ML models reaches a point where manual management becomes impractical and resource-intensive
- As the team member (data scientist, data engineer, ML engineer) size reaches a point where manual coordination and communication become more challenging, such as 10 or more
- As the organization relies more and more on the value that ML models bring to the business

Automation provides fast feedback to participants in the ML project development and thus increases the overall productivity and collaboration.

## Versioning

The main artifacts in ML projects are data, code, and ML model. An MLOps best practice is to treat these artifacts as first-class citizens, such as code in the DevOps discipline, using version control systems.

Similar to the best practices for developing software systems, ML model training code should go through a code review process, in addition to versioning, to make the training of ML models auditable and reproducible.

One of the oldest stories in the ML practitioner community is about the inability to reproduce or retrain an ML model because the original ML model author left the company and the training code and metadata were never checked into a version control system. This best practice should be able to help with this.

One of the challenges with versioning the training data is due to its size.

## Experiment Tracking

As mentioned before, ML development is a scientific endeavor that is highly iterative and experimental. Supporting this unique aspect of machine learning to enable data scientists to quickly experiment, evaluate, and compare results, collaborating with other data scientists, will require an easy way to track the metadata about the experiment, such as used parameters, performance result metrics, model lineage, data, and code.

The benefits not only include reproducibility, but more importantly the traceability. In addition, ML model exploration and iteration costs both money and time. Therefore, anything that can be done to reduce money and time will bring efficiency to organizations.