Weights & Biases

MLOps:
A Holistic Approach

Authors
Darek Kłeczek | Bryan Bischof | Hamel Husain

Contents

Introduction

As machine learning (ML) capabilities continue to improve at an astounding rate, the discussion around operationalizing these systems (termed MLOps) is now front and center. Much of this discussion is focused on technology—however, technology is rarely why ML projects fail.

ML projects fail because of poor organizational structures and processes. Sometimes, a project's value does not justify its cost. Maybe stakeholders don't trust the ML solution. Perhaps data isn't available at the right quantity, quality, or cost. Maybe poor software development practices slow down iteration cycles and hinder collaboration. Which is all to say that there are plenty of reasons projects don't deliver value.

Our goal for this report is to describe a holistic approach to MLOps that drives business value, reduces risks, and increases the success rate for ML projects. We want to provide a structure to collect best practices and to facilitate conversations around MLOps problems—and solutions.

Sometimes, operational initiatives over-index on software, but there's more to MLOps than just your toolbox. We'll focus on three complementary areas:

- **People**

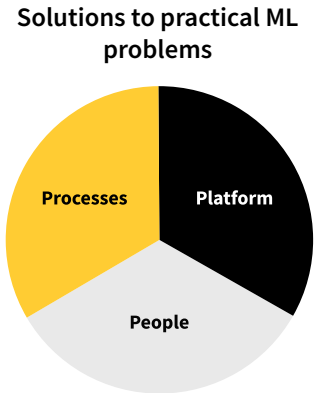
  Organizational structures and roles need to be thoughtfully designed to accommodate the unique nature of ML projects. While there is some overlap between ML and software projects, overlooking these differences is a costly mistake. We'll help you avoid this.

- **Processes**

  Well-designed processes ensure that ML teams have a positive impact on the business by enabling organizations to iterate quickly and effectively. Effective ML teams avoid taking on projects that are failures before they even start by assessing often overlooked areas of operationalization, such as the failure modes touched on above. We'll introduce you to some best practices and offer our suggestions.

- **Platform**

  With principles for people and processes in place, now is the right time to focus on having the right technology to fully operationalize ML. We'll discuss what you should consider when selecting tools, such as your team's skill sets, abstractions, and existing infrastructure.



Effective MLOps is achieved by aligning people, processes, and platforms toward the shared goal of solving a concrete business problem.

Above all else, it's critical that people, processes, and platforms are aligned towards a shared goal: **to solve a concrete business problem**. In the sections that follow, we'll share methods to validate and maximize your ability to create business value with ML before you embark on any specific project.

MLOps — The Hard Parts

Although a majority of businesses focus on purely technological aspects of MLOps, ML projects actually often stumble due to poor processes and organizational structures. To illustrate why a holistic approach is needed, we've categorized common failure modes below.

Consider going through the list below and marking problems or opportunities that are relevant to your organization:

People

Organization Design

- ☐ Insufficient or infrequent iteration, due to siloes between different teams and/or disciplines.
- ☐ Unclear or unrealistic responsibility boundaries. For example, your team has an expectation that everyone is "full stack" and should be able to perform platform engineering, data science, and product engineering tasks.
- ☐ Differences between the metrics the business uses to measure success and the model evaluation functions or experimental design objectives.

Project Management

- ☐ Goals and incentives do not allow data scientists to conduct experiments where outcomes are unknown a priori. Instead, the work is focused on predictable, low-risk—and thus often low-ROI—hypotheses.
- ☐ Management and direction of data science tasks do not properly reflect the R&D nature of machine learning.
- ☐ Usage of software development-centric management techniques (such as agile, scrum, or waterfall), despite their ineffectiveness in managing ML projects.[1]

Knowledge Sharing

- ☐ Duplicated work and poor information transfer across the team or organization.
- ☐ Onboarding new people onto an existing project is prohibitively onerous and time-consuming relative to other engineering projects.
- ☐ Poor internal documentation practices that reduce collaboration, increase siloing, and lead to different teams reinventing the same tools.

1. "Why Agile is bad for ML" Forbes

Processes

Scoping and Opportunity Sizing

- ☐ Approved projects are not suitable for machine learning, or the problems have not appropriately been framed as machine learning tasks.
- ☐ Theoretical value of approved projects does not justify the cost of development.
- ☐ Stakeholders don't trust or adopt the machine learning solutions that have been delivered.

Governance

- ☐ Ensuring that the data used to train or test models complies with data governance restrictions is difficult.
- ☐ Auditing undesirable model biases that may affect your company brand is challenging.
- ☐ External regulation requirements such as validation or documentation are not explicit priorities.

Data Discovery and Validation

- ☐ Data that is relevant to training the models that solve business problems does not exist.
- ☐ Relevant data that does exist is noisy and low quality.
- ☐ Improving data labeling is prohibitively expensive.

Data Engineering

- ☐ Complex and/or computationally expensive feature engineering requires expensive architectural exceptions in your infrastructure.
- ☐ No support for model operationalization with existing data orchestration tools.
- ☐ Working with online features requires significant effort and is not easily combined with offline feature stores.

Model Development and Evaluation

- ☐ Equipping developers with the right environment (ML-specific hardware and dependencies) and/or maintaining consistency between developer environments is challenging.
- ☐ Poor software development practices slow down iteration speed. For example, spending too much time refactoring code from notebooks to scripts and modules.
- ☐ Manual processes (e.g. manual experiment tracking) that lead to unreliable and irreproducible results, or results that are hard to share amongst the team.
- ☐ Getting ML engineers started with the compute and tooling they need is difficult.
- ☐ Migrating model prototypes to production requires significant effort, on the scale of months to years.

Decision Optimization

- ☐ Monitoring of technical metrics that do not reconcile with the business metrics used for decision-making.
- ☐ Usage of models without calibrating performance to decision-making criteria that could maximize business impact.
- ☐ Failure to articulate how a model will improve business outcomes in a way that resonates with stakeholders.

CI/CD and Testing For Machine Learning

- ☐ Software that is written to test workflows does not follow best practices (unit tests, integration tests, etc.)
- ☐ Manual and inconsistent tests.
- ☐ Manual copy and pasting of information from other platforms into code reviews (e.g. screenshots).

Deployment and Observability

- ☐ Model inference is unable to scale to meet production volumes.
- ☐ Production models do not perform well because the data in production is inconsistent with data during training.

Scaling, Orchestration, and Reproducibility

- ☐ Scaling from single to multiple models (e.g. one model for each customer, daily retraining, ensembles, shadow deployments) is not supported by the existing infrastructure or requires additional capabilities that the team realizes late in the project lifecycle.
- ☐ No good process or mechanism for automated testing
- ☐ Evaluation of ML models is not straightforward.
- ☐ Adding feedback loops from production causes unforeseen side effects.
- ☐ Changes in the external environment and production data frequently degrade model performance.
- ☐ Planning for scale is often overlooked, leading to high maintenance costs, poor user experience, and even cascading failures.
- ☐ Constant re-architecture of the tech stack as the solution scales.
- ☐ A machine learning platform that is always lagging behind the needs of the product.

Platform

Cost and Time to Value

- ☐ Hiring skilled engineers to build custom software is a common bottleneck to project completion.
- ☐ Internal tool development is suboptimal or overruns your budget.
- ☐ Custom software development delays realization of business value.

Maintainability, Interoperability, and Reliability

- ☐ Software maintenance is at risk due to developer churn.
- ☐ Monitoring, troubleshooting, and improving developer experience require specialized development.
- ☐ Integrating with various ML tools consumes significant development bandwidth.
- ☐ Hardware or software failures are disruptive to business operations.
- ☐ ML systems fail silently.

Fit for Purpose

- ☐ Employees seem unsatisfied with their tools and complain about negative impacts on productivity.
- ☐ Tools require team members to spend too much time developing skill sets that are orthogonal to their function.

The checklists above should help you narrow down your core areas of focus. How you prioritize these problems and categories depends on the specific details of your business, such as your scale, maturity, and technical capabilities. However, these risks are connected, and their nuance and variety necessitate a holistic approach to MLOps.

In the next section, we'll unpack these three categories—people, processes, and platforms—in more detail, and provide you with tools that you can apply in your own organization.

Leveling Up ML Capabilities In Your Organization

The MLOps discussion is often focused exclusively on tools. However, one critical aspect of successful machine learning operations is enabling people to succeed, which involves designing the right structure for your organization.

Additionally, you should consider the unique aspects of machine learning and how general software development principles may not be applicable to those projects.

Organization Design

Organizational structure influences how well people are aligned towards achieving business goals. Depending on the maturity of your organization, machine learning roles can be embedded in business teams instead of centralized in a separate organization. Some companies may also benefit from a dedicated group that is focused on managing the machine learning platform.

Defining clear roles is fundamental to organizational design. Well-defined roles allow people to focus on areas of interest without being spread too thinly; the surface area of machine learning systems is huge and it's often intractable for one person to cover all of the requirements effectively.

Below are some concrete areas of proficiency, along with associated job titles:

- **Data Scientist**: Explore data and find insights to influence business and product decisions.
- **Analytics Engineer**: Define and productionize business metrics.
- **Machine Learning Engineer**: Develop and evaluate machine learning models.
- **Machine Learning Researcher**: Experiment with new ML architectures and methods.
- **Data Platform Engineer**: Manage the data infrastructure.
- **MLOps Engineer**: Manage machine learning platforms and operations.
- **Machine Learning Auditor**: Control risks associated with ML solutions.
- **Product Manager**: Define what machine learning solutions to build.
- **Software Engineer**: Integrate ML capabilities into existing applications.

These slides from the Full Stack Deep Learning course provide an excellent overview of organizational design considerations for machine learning teams.

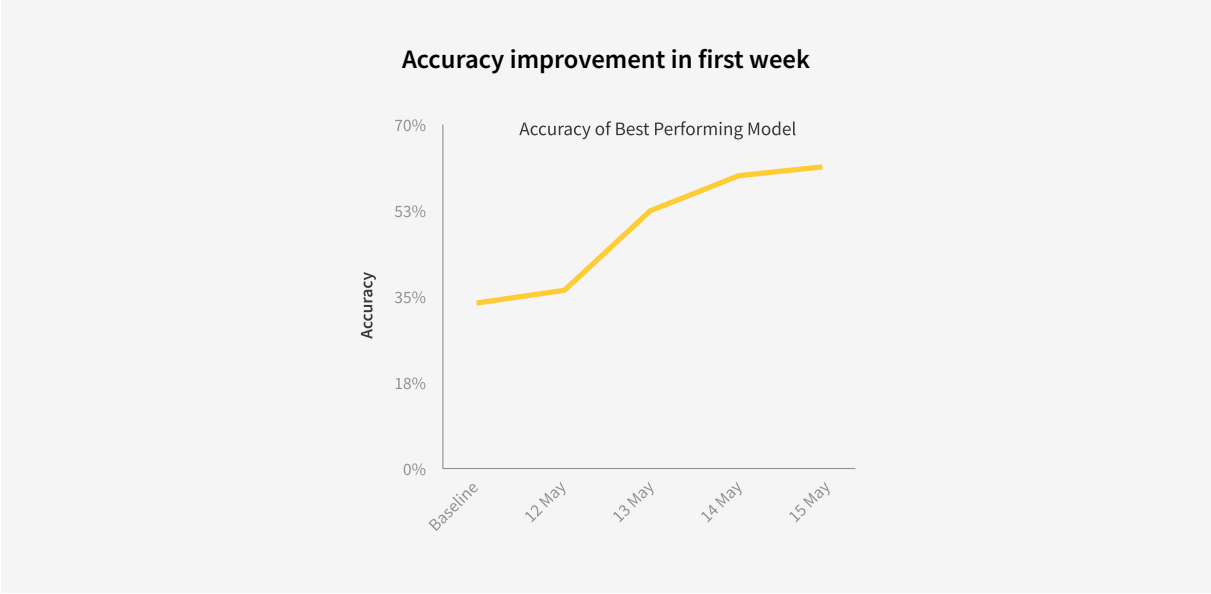
Managing Machine Learning Projects

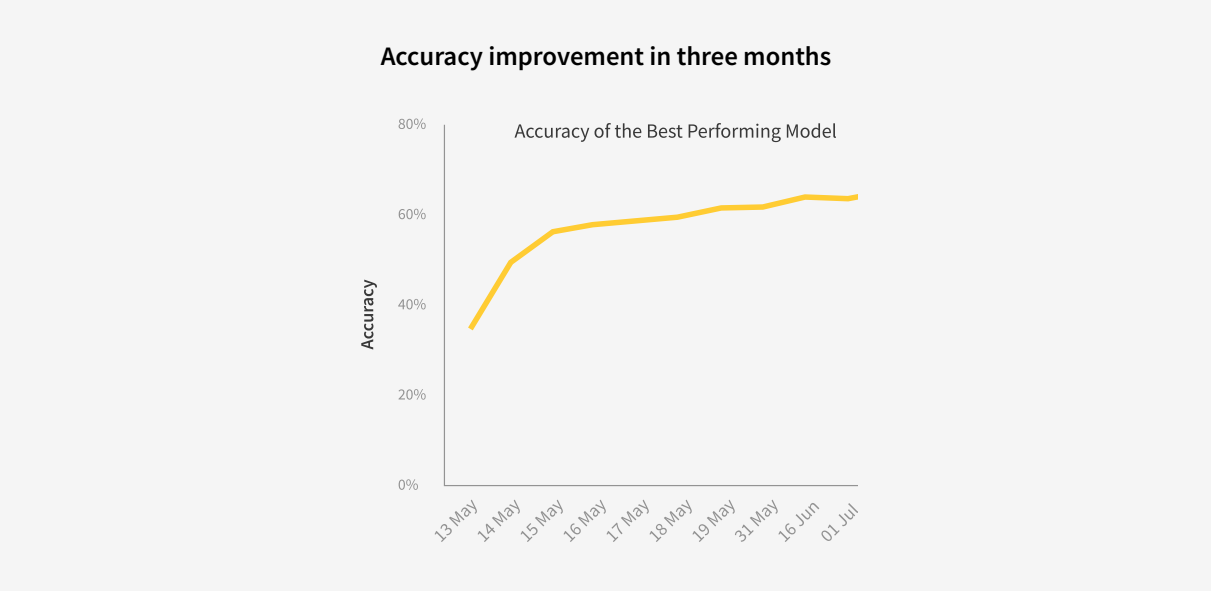
Classic project management approaches like waterfall, agile, scrum, etc. may not be the best fit for machine learning. Often, model performance and characteristics are unknown or highly speculative, making it challenging to estimate the business impact or timelines of some machine learning projects up front.

Progress on machine learning projects can plateau or have unexpected breakthroughs. The graph below, from this blog post by Lukas Biewald (CEO of Weights & Biases), illustrates this concept well.

Here, we can see the best accuracy achieved in a Kaggle competition in the first week vs. the first three months. Results like this are common. Sometimes ML projects see massive leaps in a short period of time and with little effort, while other times there is little progress despite significant effort. It can be very difficult to know which is more likely.

Uncertainty exists not only in how long specific tasks might take, but also in which tasks to perform at all. Later tasks are often highly dependent on the outcome of earlier ones. For example, the early efforts in a project are often dominated by experimentation and the results of those experiments dictate which paths to pursue.





Source: Why are machine learning projects so hard to manage?

Knowledge Sharing

A key component in organizational effectiveness is ensuring that information and context are shared, in order to drive alignment on company goals and minimize duplicate efforts. The importance of this becomes acute with data science and machine learning, since much of the work requires experimentation.

Tools that help facilitate knowledge transfer and sharing of technical information, like W&B Reports, can uplevel the quality of data science and machine learning knowledge across organizations. W&B Reports allow you to display and aggregate information seamlessly across all of your W&B projects, without having to extract data into external systems. Reports also have a plethora of collaboration features, including commenting, shared editing, and the ability to be embedded in a wide variety of platforms.

Tools and platforms are important, but they are not sufficient to achieve success with machine learning projects alone. While some tools facilitate good processes, they cannot supplant them entirely. Below, we'll discuss the processes that are critical to successful ML projects.

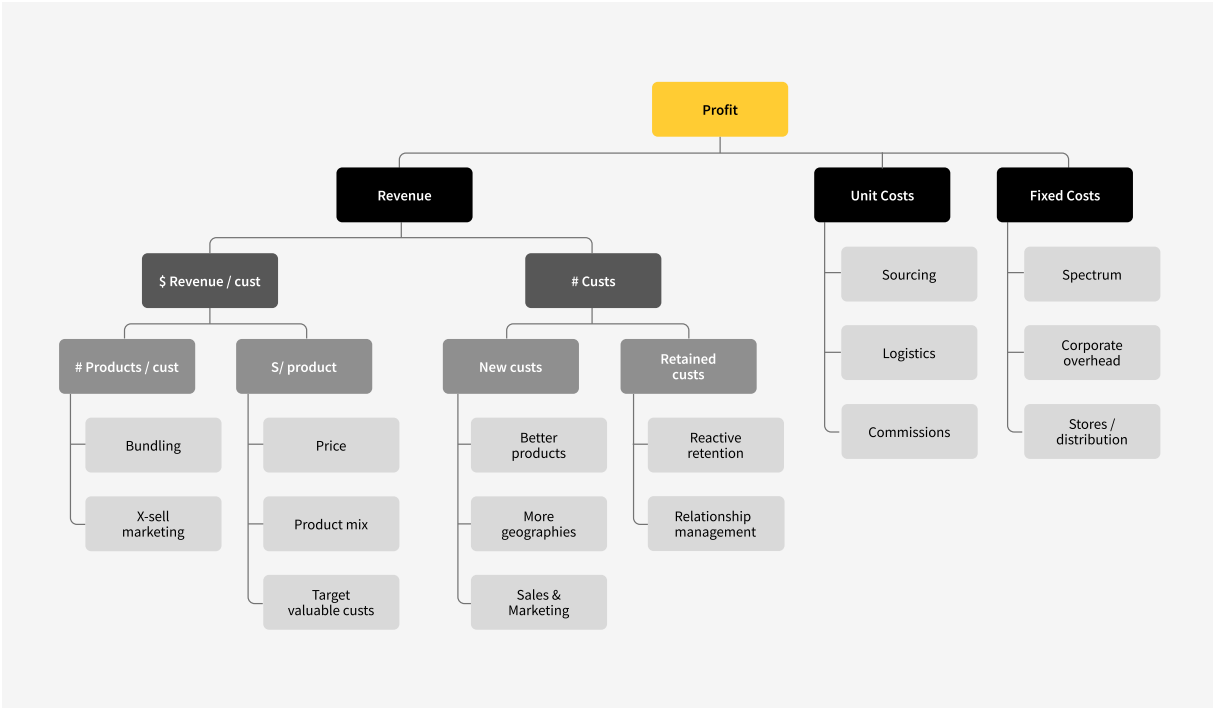
Designing Good Processes Around ML Projects

Scoping and Opportunity Sizing

Assessing which ML projects maximize business impact is not something that's part of most data science training. Business impact can come in many forms: efficiency, fairness, accuracy, or—most obviously—value.

This assessment is an essential skill for career success as a data scientist, but is most commonly a skill learned on the job through repetition and experience. Business partners often assume that machine learning is the answer to their problems, even when it may not be. Conversely, they may be skeptical of machine learning. Picking the right projects and assessing business value upfront is critical for long-term success.

Thankfully, there are concrete tools and approaches that can help teams assess the business viability of a project before embarking on it. Below is a diagram that illustrates some of the things you may want to consider when assessing the business impact of a proposed ML project:



A subset of possible business profit drivers that should be considered when sizing the opportunity of a potential machine learning project. Source: "Data Project Checklist", fast.ai

If you'd like a bit more information, the "Assessing The Feasibility of The Project" section of our blog post on optimizing business value with ML provides a concrete example of a structured approach to assessing ML projects.

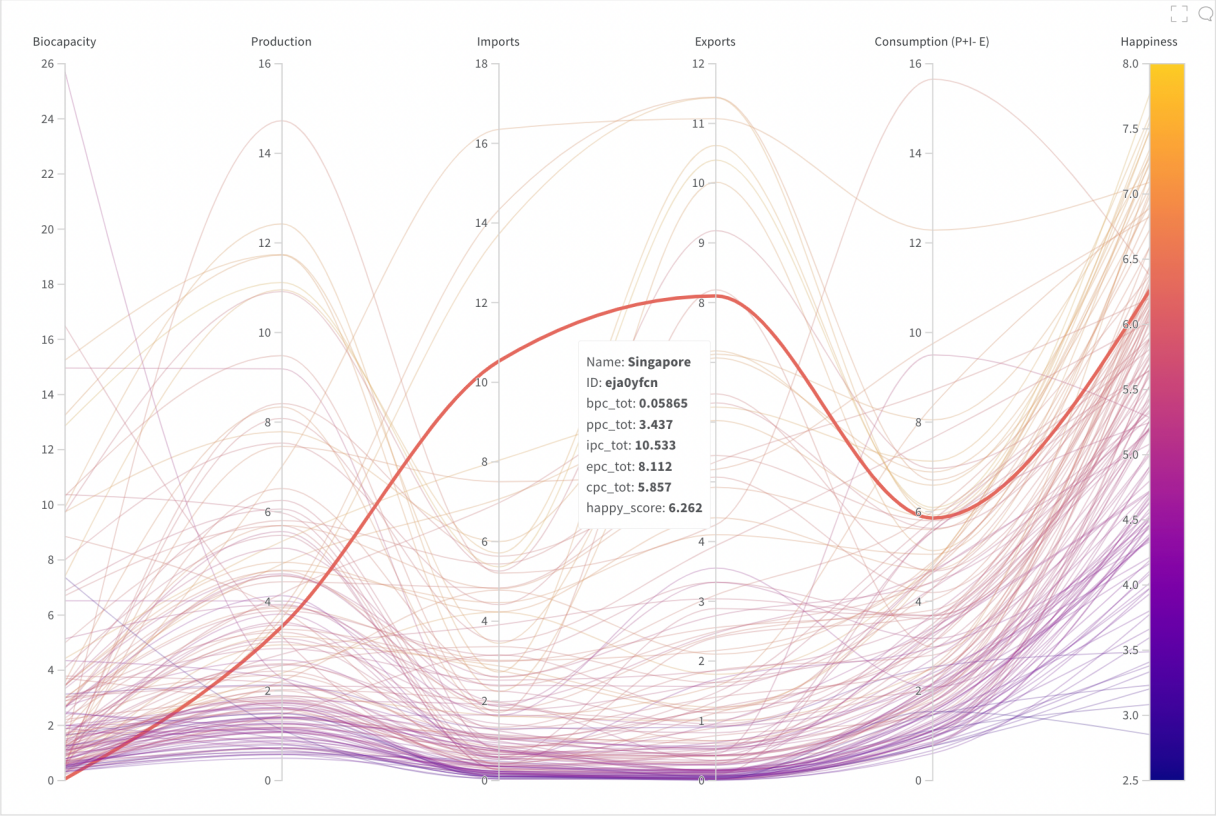
Data Discovery and Validation

When we first embark on an ML project, we must do a significant amount of exploration to identify if relevant data exists and where it exists, as well as how to extract, transform, and validate it. Often, the data will not already exist in an easy-to-retrieve SQL table. Furthermore, the veracity and freshness of the data may be unknown, and data engineers will be expected to validate and ensure continued data quality.[2]

Data quality insurance can take many forms including:

- Type-checking (no strings in your bool column)
- Distributional expectations (your average doesn't suddenly shift by an order of magnitude)
- Value errors (your color picker doesn't allow "dog")
- Diversity (your AV dataset has examples of firetrucks avoiding a redlight), and
- Consistent volume (today's new user signups is not down 99%).

It is also important that this early-stage discovery process—usually called Exploratory Data Analysis (EDA)—is documented and reproducible. For example, this W&B Report demonstrates EDA for the World Happiness Report.



Using parallel coordinate plot in W&B for exploratory data analysis. Source: Planetary Well-Being Metrics

2. Data Quality Fundamentals, Barr Moses, Lior Gavish, and Molly Vorweck, 2022.

Model Development and Evaluation

Before we operationalize models, we need to conduct experiments to validate their feasibility and estimate their business impact. This often involves building a simple baseline, followed by collaborative and iterative experimentation and model evaluation. We'll discuss each of these stages below.

The first step of a new machine learning project should be understanding the business context, which we discussed in "Scoping and Opportunity Sizing". In parallel, we should engage in "Data Discovery and Validation" to explore and understand the data. Finally, we need to define a relevant model performance metric that correlates with our business objective. It's often beneficial to build a simple baseline and review it with subject matter experts before investing in advanced modeling.

Feedback from subject matter experts is critical in effective experimentation. Conducting error analysis with their input can help direct future investments. For example, we may decide to extend our training data, change our model's architecture, or improve our labeling process. It is important to establish tools that allow rapid iteration and collaboration with stakeholders.

We often need to look at metrics not only in the aggregate, but also on specific slices of data. For example, with self-driving cars, segmenting all classes from a camera feed correctly is important, but we also need to be able to look at the metrics for specific classes such as pedestrians. This W&B Report provides an example of error analysis on an autonomous vehicle task.

A robust model development and evaluation process will allow you to iterate fast, effectively communicate with stakeholders, and properly evaluate models. Weights & Biases Experiments and Reports can facilitate these tasks.

Development Tools

One prerequisite of effective model development and evaluation is setting up a development environment, which often involves configuring powerful remote compute environments rather than engineers' laptops. The features listed below are needed to create repeatable and reliable development environments.

- **Dependency Management**

  Most machine learning models and pipelines are developed in Python and require dependency management tools such as Poetry or Conda. Docker, virtualenv, or Pipenv can also simplify the process of developing and deploying machine learning models by ensuring a consistent environment.

Because there are such a wide variety of dependency and environment management tools, it is beneficial to standardize on a few, since disparate systems across a team reduce collaboration and effectiveness. Additionally, creating paved paths for using dependency management across prototyping and production generally pays significant dividends down the line.

W&B offers excellent support for a wide variety of dependency management frameworks, including Docker, conda, or pip.

- **Sandbox Environments**

  Many companies use cloud providers or on-prem hardware for their compute needs. However, developers need the ability to create sandboxed environments with all of the tools that are consistent with your production environment. This may require creating a locked-down control plane to limit the surface area of what can be utilized. Minimizing the discrepancy between the development and production environment is key for enabling rapid iteration.

- **General Development Tools**

  There are many tools that help ML practitioners be more productive while experimenting. These may include terminal tools (e.g. tmux), IDEs (e.g. VSCode, PyCharm, Jupyter Lab), code versioning tools (e.g. git), and debugging tools[3]. It is important that your sandbox environment supports these general development tools, in order to enable a minimum viable developer experience for ML engineers.

- **Jupyter Notebooks**

  Jupyter Notebooks are central to a machine learning engineer's ability to rapidly prototype and experiment with ideas. However, there is much debate in the community about how to use notebooks effectively, with some even claiming that notebooks should not be used at all.

  Nevertheless, there are tools and processes that enable teams to use notebooks effectively while still maintaining software engineering best practices, including distributing Python packages, writing tests, and documentation. Tools like nbdev, Metaflow, and Quarto offer facilities for developing and deploying production-grade software and sharing knowledge with notebooks.

  W&B also allows you to leverage the full power of W&B features from a notebook with its excellent set of notebook integrations.

Decision Optimization

ML is often used to optimize operational decisions. However, models only provide predictions, not business decisions. We need to calibrate how to use our model to make decisions. For example, in the context of classification, if the cost of false positives is very high, we might tweak our decision threshold to prioritize fewer false positives at the cost of more false negatives, and ultimately greater business impact. Failure to calibrate[4] decision thresholds may result in negative business value, despite having predictive models.

The "Tuning The Model For Business Value" section of this blog post provides a decision optimization framework, demonstrated with a practical example that shows you how to maximize business value.

---

3. W&B provides optional code snapshotting to tie back versions of the code used to generate experiments logged to W&B.

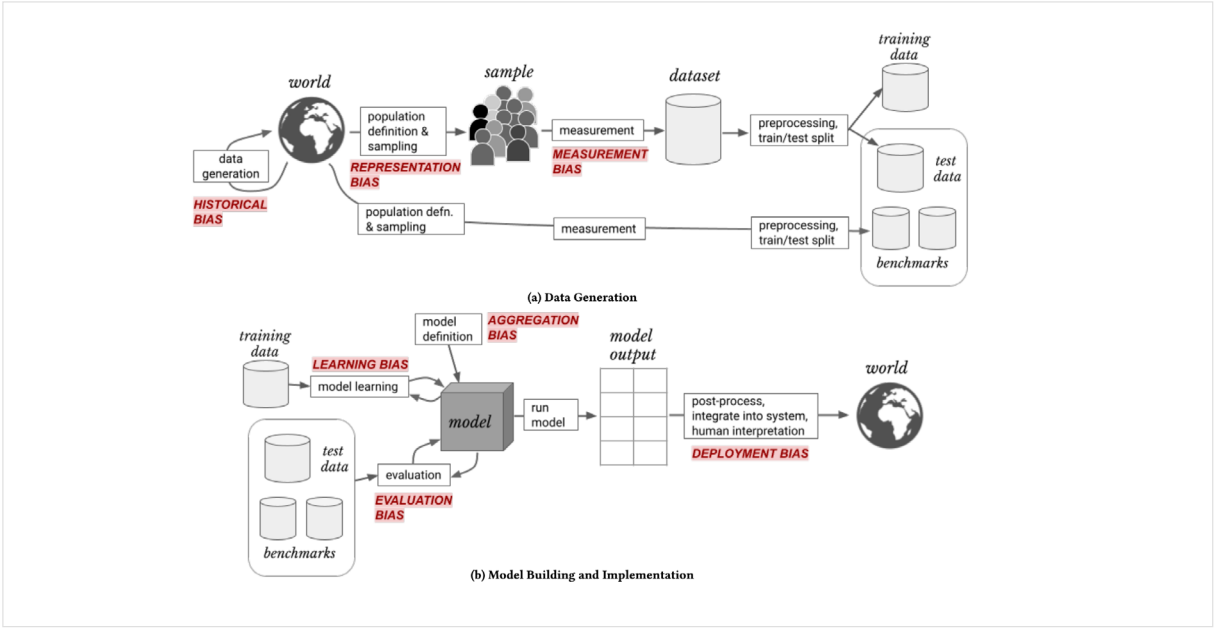
4. "Classifier Calibration", Bryan Bischof, 2022

W&B also facilitates error analysis of models. For example, this Report illustrates where a computer vision model struggles to discern amongst similar animal types, enabling an ML engineer to analyze and improve on model errors.

image	guess	truth	score_Amphibia	score_Animalia	score_Arachnida	score_Aves	score_Fungi	score_Insecta	score_Mammalia	score_Mollusca	score_Plantae	score_Reptilia
	Amphibia	Amphibia	0.814	0.0057	0.0776	0.0006	0.0116	0.0235	0.0059	0.0198	0.0012	0.0401
	Amphibia	Amphibia	0.7901	0.0045	0.0619	0.0013	0.0104	0.0187	0.0073	0.0219	0.0025	0.0813
	Amphibia	Insecta	0.2716	0.002	0.1361	0.0018	0.0328	0.2425	0.0304	0.0211	0.011	0.2507

Governance

There are many sources of bias that can occur in machine learning. It is important to understand what each of these types of biases are, their root causes, and how to implement systems and processes to mitigate them. Here is a diagram of seven common types of biases, as illustrated in the paper "A Framework for Understanding Sources of Harm throughout the Machine Learning Life Cycle":



Seven types of biases introduced at various stages in the machine learning project lifecycle.

Let's quickly define the biases in the diagram above:

- **Historical bias:** How data is originally generated.
- **Representation bias:** How the data sampling strategy is defined.
- **Measurement bias:** How the data sampling strategy is carried out.
- **Aggregation bias:** How models are defined.
- **Learning bias:** How models are trained on sampled data.
- **Evaluation bias:** How models are validated against benchmarks.
- **Deployment bias:** How the final implementation of the machine learning system is experienced by the world.

W&B Reports can be leveraged in your overall governance process to identify biases in your data and models, as well as to share that knowledge across the organization. For example, this W&B Report outlines various tactics you can use to discover bias and fairness issues in your models.

Furthermore, W&B can assist you with data privacy management, like detecting issues such as inadvertently training models on private data. W&B's Artifact lineage provides visibility into all of the data used by your models and metrics, which can be used to satisfy compliance audits related to data access. For example this Report shows examples around regulatory audits.

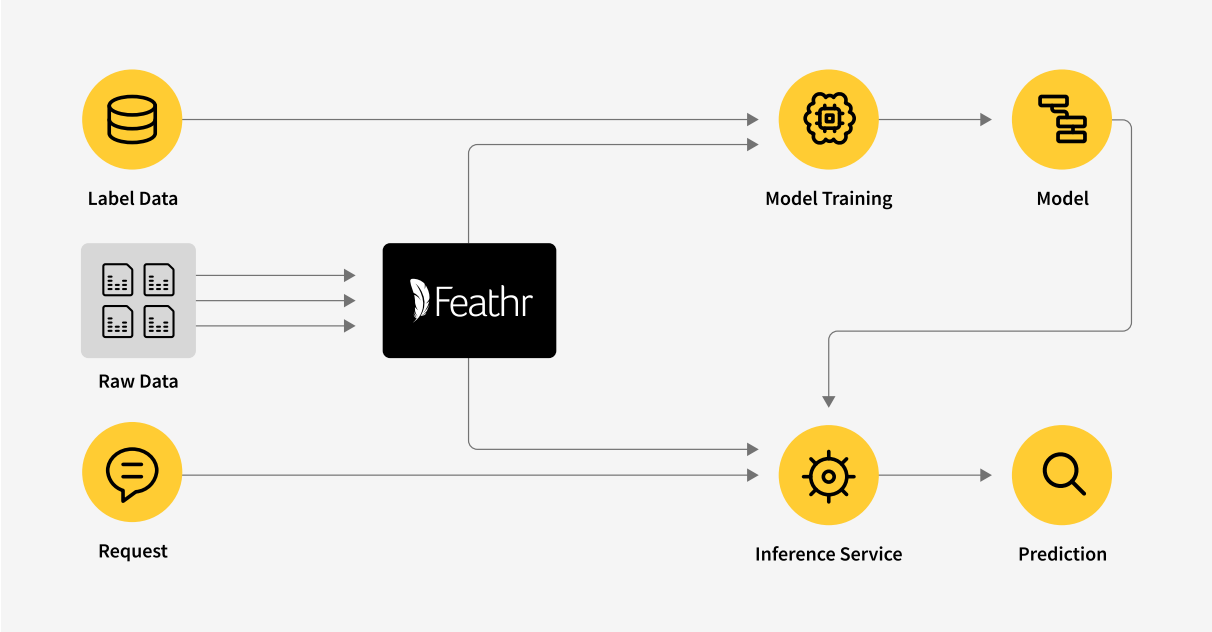
Data Engineering

A prerequisite for training machine learning models is data preparation. This makes data engineering a central component of any ML operationalization task.

In "Data Discovery and Validation", we discussed how to most effectively discover, validate, and prepare data. However, this data transformation and validation needs to be repeatable and consistent in order to operationalize models.

Data pipelines allow us to automate the sourcing, transformation, and validation of data for model training and inference. A feature store can also help with consistency and reproducibility by organizing your features in a transparent way. Feature stores can also provide features on demand for high-availability and low-latency environments.

Most feature stores also ensure that your batch data and your streaming data stay in sync. For example, Linkedin built Feathr to promote consistency of data across the training and serving of models:

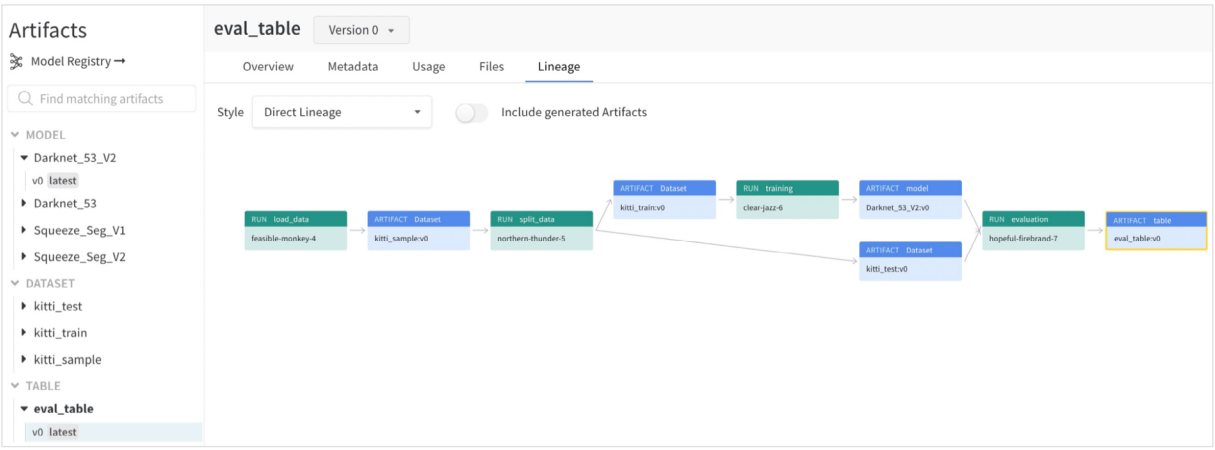


The purpose of Feathr, and feature stores in general, is to unify data processing for both “offline” processes, such as model training, and “online” processes, such as model inference. Source: Linkedin Engineering Blog

Furthermore, if your dataset requires frequent labeling or human review, it may be helpful to use existing models for assistance. For example, you can use a model's feedback to prepare labels—to be verified later by an expert annotator—and select the most valuable examples for labeling (active learning).

Even though data engineers may shoulder some of this orchestration and pipelining work, we find that having an understanding of data engineering techniques is critical for ML practitioners to be effective when designing an ML product. W&B Artifacts can help you couple data engineering tools with your model pipelines by snapshotting and tracking data lineage associated with models.

For example, W&B Artifacts will build a dependency graph of data for you as data flows through your machine learning pipeline, as well as allow you to version and track how your datasets change over time. Artifacts also link registered models to their associated data artifact:



W&B Artifacts let you track the dependencies of your data across multiple versions.

Scaling, Orchestration, and Reproducibility

After developing a baseline, we often need to prepare our model for production along the following dimensions:

- **Scale**

  You may need to train additional (potentially many more) models for specific customers or products. Often, you may also want to increase your model's capacity so that it can learn from a larger dataset. You may also need to compress your model so that it can meet your organization's constraints (latency, memory, etc.).

  But how does a machine learning engineer or data platform engineer scale their systems to trillions of inferences per day? Some canonical methods borrowed from software engineering include horizontal and vertical scaling, load balancing, localization, replication, and asynchrony.

  While these software engineering methods are important and valuable, MLOps has some additional considerations with respect to scale. Two important data processing regimes in ML are **batch** and **streaming**.

  For applications that utilize data that is known ahead of time, pre-processing that data as a batch can make the application much more performant. Data that is not knowable ahead of time may require a streaming approach. A combination of batch and streaming approaches are often necessary, which requires a system for processing data from both regimes in a consistent and performant way.

- **Orchestration**

  In order to ensure that your models do not become stale, you may want to regularly retrain them on recent data. You may also want to automate model testing and evaluation to ensure that retraining actually improves model performance. Lastly, you should ensure that downstream or upstream dependencies are refreshed appropriately when your model is retrained.

  Generally, creating these workflows starts by expressing your pipeline as a directed acyclic graph (DAG). These pipelines need to be integrated with compute resources and be able to track artifacts and metrics associated with each execution. The orchestration engine (e.g. Dagster, Airflow, Prefect) takes care of executing the pipeline based on a schedule or a specific trigger (for example, when new data arrives).

- **Reproducibility**

  People on a team should be able to debug and collaborate on machine learning workflows so that they can maintain them. Workflows should be reproducible, which involves not only snapshotting code but also data and any user inputs for running the model.

  Another important aspect of reproducibility is portability. Portability is achieved when you can easily integrate components of your workflow into other workflows and quickly add new components in a modular way.

  W&B Launch provides tools that can get you started with scaling, orchestrating, and reproducing experiments. You can use it to execute runs in reproducible containerized environments as well as queue and launch jobs across your own local or cloud clusters.

CI/CD and Testing For Machine Learning

In traditional software development, continuous integration/continuous deployment (CI/CD) is used to automate many tasks including testing, building, and deploying software. CI/CD can also be adapted to ML projects. For example, in addition to unit tests for your code, you can perform smoke tests on your models (i.e. validating that your models can predict and learn from example data). You can also have CI/CD systems retrieve data from W&B and display relative metrics in a pull request.

It is important that you establish some level of testing and automation when you change code associated with your ML workflows. Using CI/CD tools where available will also help you build trust with your engineering peers by being deliberate about software tests and making those results visible to everyone on your team.

Below is a demo that uses Kubeflow, W&B, and GitHub Actions to demonstrate CI/CD for ML:
Machine learning operations with GitHub Actions and Kubernetes - GitHub Universe 2019

Deployment and Observability

In order to realize business value from our machine learning models, we need to actually deploy them. When designing a model deployment system you will have to consider latency (batch, real-time, streaming), scale, and security requirements.

Generally speaking, batch predictions are simpler than real-time or streaming because you can asynchronously write raw data and predictions to a database without having to worry too much about latency, memory constraints, or web servers being able to handle prediction requests.

When running inference at scale, it may be important to optimize the model to reduce latency and infrastructure costs. Traditional models, such as decision trees, can be compiled (optimized for inference execution) to improve efficiency. For deep learning models, we can perform distillation or use lower floating point precision to reduce their size, as well as use frameworks to serialize and optimize models to run on specific hardware[5].

Moving a model to production requires following a defined workflow. For example, we often want to define criteria to promote a model from development to staging, and from staging to production. Additionally, we need to be able to track model lineage so that we can associate models with data and evaluation metrics at various parts of the workflow. This type of observability can be enabled through W&B Models.

Another kind of observability is monitoring our model's performance in real time. We will want to monitor the distribution of input data to detect and react to data drift. Furthermore, it often makes sense to have a human in the loop looking at a fraction of model predictions and verifying model performance, especially on new information or edge cases. New, freshly labeled data can then be fed into a live training loop to improve model performance[6].

Finally, we want to make sure we are monitoring business metrics associated with ML projects. If we deploy an improved ML model, we want to verify that it improves our business metrics.

5. One such example is model quantization.
6. This is often referred to as "continuous learning".

Before deploying your models, it can be useful to conduct A/B tests to verify that your new model is better than an incumbent model. Utilizing randomization, we can build statistical confidence that observed differences in metrics between two subpopulations are due to the variation between their exposure to one model or the other. Experimentation can also come in the form of reinforcement learning, but the A/B testing framework is the most common approach to evaluate the effect of a new model.

For example, Airbnb leveraged A/B testing tools (pictured below) to ensure that recommendation models were affecting business metrics:

> **"We conducted an A/B test to compare the new setup with 2 models with Personalization features to the model from Stage 1. The results showed that Personalization matters as we were able to improve bookings by +7.9%..."**
>
> Source: Machine Learning-Powered Search Ranking of Airbnb Experiences

Airbnb leverages the powerful A/B testing platform pictured here to make decisions that improve customer satisfaction.
Source: Machine Learning-Powered Search Ranking of Airbnb Experiences

To enable A/B tests and observability of your deployment workflows, it is necessary to version and track your model deployment candidates. A model registry can help you track and version models from staging to production, as well as give you visibility into a model's history.

W&B Models provide you with robust tools to version, track, and organize your models for the purposes of deployment.

ML Platforms

It is often hard to decouple processes from technology. In an ideal world, we would start with the process and pick the right technology to support it. In reality, technology often influences or constrains our processes.

It is therefore important to build a platform that is flexible enough to support your workflows, while also integrating well with other solutions in your stack. We have highlighted many tools throughout this article that you can use to compose an end-to-end ML Platform, which we'll summarize later in this section.

From a technological perspective, there are three main categories of tools relevant to ML:

- **Frameworks**

  While low-code and no-code platforms are gaining some traction, most ML practitioners prefer code to train and evaluate models. There are many frameworks (often open-source) at various levels of abstraction that support this, including TensorFlow, Keras, PyTorch, Lightning, fastai, XGBoost, and JAX. Similarly, there are frameworks focused on particular ML tasks like YOLOv5, Detectron, and GPT-3.

- **Ops tools**

  Performing every step of the ML workflow manually is not efficient. Adopting developer tools can make ML practitioners more productive, promote best practices, and improve business results. Some popular tools in this category help with tasks like experiment tracking, handling features, labeling, versioning, serving, monitoring, and orchestration.

- **Compute and Storage**

  In order to execute ML workflows, we need storage and compute provided through the infrastructure layer. Both are most often sourced through cloud computing (AWS, GCP, Azure) but some companies choose to manage this on their own (on premises).

For reasons discussed below, we believe good ML platforms offer maximal flexibility with regards to ML frameworks, leaving the data scientists free to work with any framework they choose. Furthermore, your compute and storage decisions are likely solidified well before you start to engage in building an ML Platform. Therefore, this section will focus primarily on Ops tools.

Considerations When Selecting Tools

Some important considerations when building an ML Platform are: skill sets, abstractions, existing infrastructure, monoliths vs. point solutions, and build vs. buy. We'll discuss some of these below.

Skill Sets

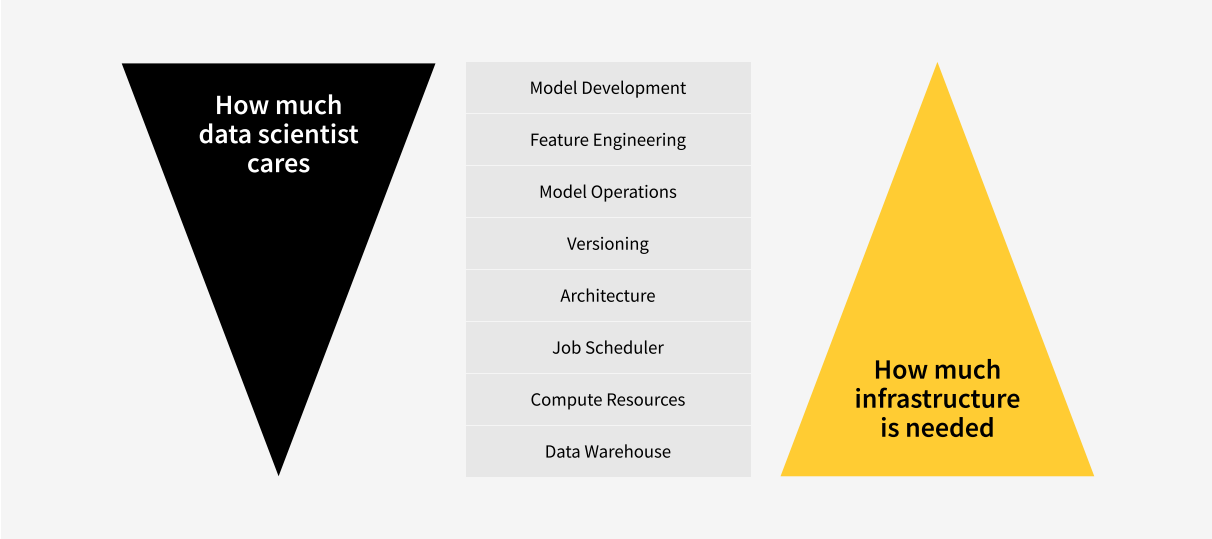
When deciding which tools to use as part of your platform, it is important to select tools in the context of your organization's skill set. For example, if a tool assumes knowledge of Kubernetes, does someone with sufficient Kubernetes experience work on your team?

If not, does it make sense to hire a person if this would be the only use of their skills? You should also make sure to understand if there might be people on your team who are already familiar with certain tools from previous jobs or roles that you can leverage right away.

Another consideration that is often overlooked is how your team wants to grow professionally in their careers. Engineers on your team with a growth mindset may view the opportunity to use new tools as a strong incentive to keep working at your company. Similarly, prospective new hires often screen companies based on the tools they use. You must balance leveraging the skills you already have vs. learning new tools.

Abstractions

It is important to build the right abstractions to help your team focus on high-value activities. This can be a tricky balance because there is a fundamental mismatch between how much infrastructure is needed at different parts of the stack and what data scientists care about:



Data scientists tend to care less about capabilities that require more technological infrastructure. This hypothesis underpins much of the MLOps discussion today. Source: Effective Data Science Infrastructure

The more data scientists care about a particular area, the more flexibility they will want. As a rule, it is often a good idea to create an ML Platform that gives data scientists a wide range of freedom on model development and feature engineering tools, while giving them high-level abstractions that allow them to reduce time spent configuring compute resources, even if that limits their flexibility in those areas.

Finally, it is important to try to meet your developers where they are. For example, for data scientists, this can mean offering them familiar APIs such as Python or SQL, and supporting development environments they're comfortable with, like Jupyter Notebooks.

Existing Infrastructure

You may have existing data or other infrastructure prior to building an ML Platform. If so, it is useful to determine whether the set of tools you are considering complements your existing infrastructure. In other words, if you are using Kubernetes, it might be desirable for you to deploy newly adopted tools on Kubernetes in order to leverage paved paths that already exist in terms of skill sets, security, or observability. Additionally, if your pipeline is not easily schedulable, you may need to create custom integrations with your existing platform.

Point Solutions vs. Monoliths

Point solutions tackle one area of the ML workflow really well. Because point solutions are focused, they can be feature-rich for a specific task. On the other hand, monoliths provide a broad range of infrastructure in one package but may not focus very deeply in any one area.

It is important to note that most tools are not strictly point solutions or monoliths but exist on a spectrum. Also, point solutions often drift towards monoliths over time as more features are added. When evaluating tools in this context, it is important to pay attention to this dynamic and assess the features you care about most.

Lastly, point solutions also tend to offer more flexibility in terms of interoperating with other tools and are easier to onboard, whereas monoliths offer a more opinionated set of tools that you do not have to glue together yourself. You should consider these tradeoffs when selecting tools for your ML Platform.

Build vs. Buy

When designing your ML platform, you need to decide between in-house development and buying the ML tools. Here are some factors you should consider in your decision-making process.

- **Business value of ML projects**

  You'll want to maximize the business value of your ML projects, and having tools that support your data scientists to get better outputs faster will increase the top line. You'll get better outputs by using tools that are familiar to your engineers and offer great user experience.

- **Total cost of ownership**

  Hiring engineers to develop your own tools can be expensive. IT projects also often overrun initial estimates, which you should factor into your budget if pursuing in-house development.

- **Maintenance and scalability**

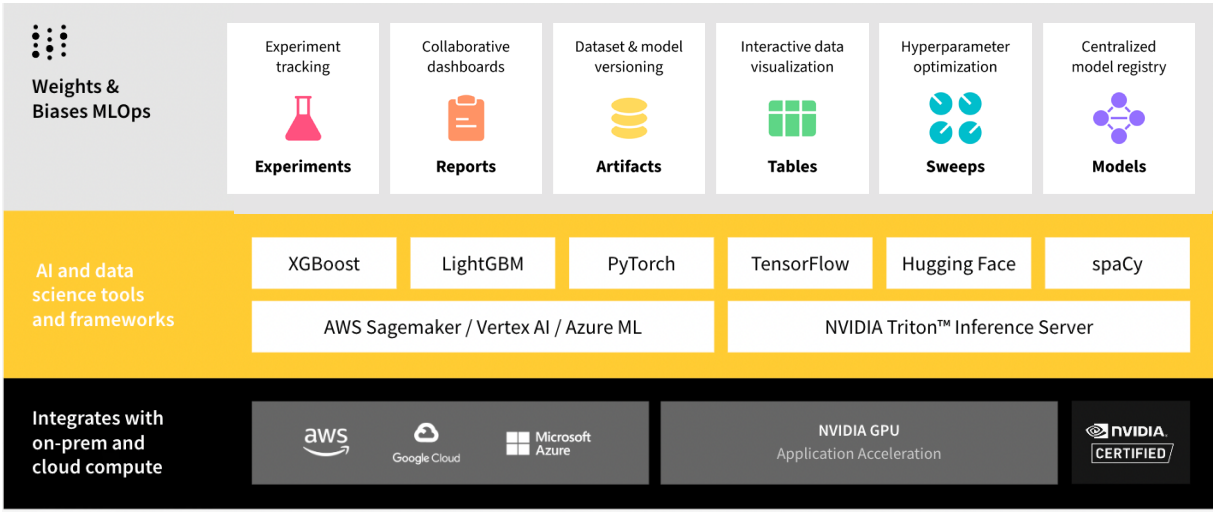
  There are costs associated with maintaining and monitoring in-house solutions. You may get downtimes that also impact your business value. As your solution scales, the maintenance requirements often grow exponentially.

Other Considerations

There are many other considerations such as cost, support, and maturity. However, in our experience, the categories we explored above are particularly important—yet often overlooked—with respect to machine learning.

How Weights & Biases Fits In

Weights & Biases offers a machine learning platform for developers with a focus on improving observability and iteration speed. Using W&B is extremely lightweight, as you can get started with just a few lines of code. W&B allows you to track experiments, version and iterate on datasets, evaluate model performance, reproduce models, visualize results, identify errors, and share findings with colleagues.



The platform consists of the following features:

- Experiments

  Experiments enable you to track and visualize experiments in real time, compare baselines, and iterate quickly on ML projects. The W&B Python library is lightweight, compatible with workflows of any shape and size, and can be easily integrated into your training and evaluation pipelines.

- Reports

  Reports let you organize and embed visualizations, describe your findings, and share updates with collaborators. As a tool for project management and collaboration in machine learning projects, Reports combine the power of interactive, dynamic visualizations with text and rich media.

- Artifacts

  Artifacts make it easy to get a complete and auditable history of changes to your files by enabling dataset versioning, model versioning, and the tracking of dependencies and results across machine learning pipelines. Store entire datasets directly in Artifacts, or use Artifact references to point to data in other systems like S3, GCP, or your own bucket.

- Tables

  Tables allow you to log, query, and analyze tabular data in order to understand higher-level patterns in your data. Compare changes precisely across models, epochs, or individual examples, understand datasets, visualize model predictions, and share insights in a central dashboard.

- **Sweeps**

  Sweeps help automate hyperparameter optimization and explore the space of possible models. Optimize models more quickly and free up time for teams to focus on other parts of the machine learning pipeline.

- **Models**

  Models provide a central system of record for models. Create registered models to organize your best model versions for a given task, and track models as they move into staging and production.

W&B doesn't address the entire ML workflow, but rather focuses on providing a set of best-in-class tools for several areas. This is why W&B offers integrations with a wide variety of data science infrastructure, such as Databricks, Metaflow, Kubeflow, Prodigy, Ray, SageMaker, Lightning, MosaicML, and more. This flexibility enables you to compose an end-to-end ML Platform that fits your needs.

Addressing MLOps Opportunities With W&B

As discussed throughout this piece, effective MLOps is characterized by putting the right processes in place and organizing the right people appropriately.

Revisiting our original checklist, here are some of the ways W&B can help you build a platform that's right for your unique organization:

People

Organization Design

- ☑ W&B Reports facilitate collaboration and communication across a wide variety of disciplines in your organization. This allows people in various roles in your organization to participate in an end-to-end workflow.

Project Management

- ☑ W&B Experiments and Reports give project managers unparalleled visibility into the status of machine learning projects, while complementing their existing tools and processes.

Knowledge Sharing

- ☑ W&B Reports facilitate knowledge sharing with features like commenting, shared editing, and the ability to be embedded in a wide variety of platforms.
- ☑ You can use W&B with other tools that facilitate knowledge sharing like nbdev and Quarto.

Processes

Scoping and Opportunity Sizing

- ☑ W&B Reports allow you to share analysis and presentations, including those related to scoping and opportunity validation.

Decision Optimization

- ☑ W&B Tables allow you to perform analyses on your model's metadata including model calibration and decision optimization. Furthermore, these tools can be used in Jupyter Notebooks for additional flexibility and integration with other tools.

Data Discovery and Validation

- ☑ W&B Tables allow you to explore and visualize your data, and can be used alongside other tools in Jupyter Notebooks.
- ☑ W&B Reports allow you to document and share findings from data discovery and validation with your team, as illustrated in this example.

Governance

- ☑ W&B Reports can be used to augment your governance process as illustrated in this example, which includes various tactics you can use to investigate model fairness.
- ☑ The W&B API can be used to integrate with governance-related tools such as fiddler.ai.

Model Development and Evaluation

- ☑ W&B Experiments and Reports help facilitate model development and evaluation.
- ☑ W&B Sweeps automate hyperparameter tuning, improving the productivity of both your engineers and your GPUs.
- ☑ W&B Tables allow you to evaluate model predictions in an interactive and visual way.
- ☑ Furthermore, W&B offers integrations with popular ML frameworks that are focused on development and evaluation such as: scikit-learn, Keras, PyTorch, fastai, XGBoost, TensorFlow, SpaCy, and many more.

Data Engineering

- ☑ W&B Artifacts can be used to couple data engineering tools with your model pipelines by snapshotting and tracking data lineage associated with models.

Scaling, Orchestration, and Reproducibility

- ☑ W&B Launch provides tools that can get you started with scaling, orchestrating, and reproducing experiments.
- ☑ Furthermore, W&B integrates with relevant tools such as Metaflow, Kubeflow, SageMaker, and Databricks.

CI/CD and Testing For Machine Learning

- ☑ The W&B API and CI/CD systems like GitHub Actions can be used to enable workflows like those demonstrated here:

  Machine learning operations with GitHub Actions and Kubernetes - GitHub Universe 2019

Deployment and Observability

- ☑ W&B Models provide you with robust tools to version, track, and organize your models for deployment purposes.
- ☑ With the W&B API, you can retrieve models from the registry for A/B testing and serving with the tools of your choice.

Platform

Cost and Time to Value

- ☑ W&B's user-based pricing model is easy to calculate and budget.
- ☑ W&B is easy to set up and start using quickly, while also being capable of running thousands of experiments at scale to build models collaboratively.
- ☑ W&B tracks hardware usage for machine learning experiments, which allows for optimization while avoiding wasting expensive resources.

Maintainability, Interoperability, and Reliability

- ☑ W&B performs monthly high-quality releases while constantly improving the platform, ensuring an uptime greater than 99.95% and a high NPS over 75.
- ☑ W&B is integrated into every popular ML framework and instrumented in thousands of popular ML repos.
- ☑ Technology partner integrations are a smooth experience with W&B and Amazon Sagemaker, Vertex AI, Kubeflow, etc.

Fit for Purpose

- ☑ W&B keeps investing in a platform that is co-designed with top-notch institutions such as OpenAI and used by hundreds of companies around the world.
- ☑ Over 300,000 ML practitioners love using W&B.

Get In Touch

Operating machine learning systems requires a comprehensive design that consists of people, processes, and platforms. We believe this guide provides a holistic overview of the components of highly functional machine learning operations.

If you would like to learn more about W&B and how we can partner with your organization to drive business value with ML, sign up or request a demo.

Further Reading

- This in-depth survey describes challenges and opportunities of operationalizing ML. The survey was conducted across a broad range of industries and domains.
- Full Stack Deep Learning is a course that dives deeper into some of the topics discussed in this article, such as considerations when building ML infrastructure as well as how to organize ML teams.
- Designing Machine Learning Systems by Chip Huyen goes into great depth regarding the various components of ML systems and associated best practices.

Darek Kłeczek is a Machine Learning Engineer at Weights & Biases, where he leads the W&B education program. Previously, he applied machine learning across supply chain, manufacturing, legal, and commercial use cases. He also worked on operationalizing machine learning at P&G. Darek contributed the first Polish versions of BERT and GPT language models and is a leader in the Polish NLP community. He's a Kaggle competition winner and 3x Kaggle Master.

Dr. Bryan Bischof is an adjunct professor of Data Science at Rutgers University, and formerly the Head of Data Science at Weights & Biases. He's worked in time series signal processing at scale, demand forecasting, global optimization and logistics, and personalized recommendations at companies like Stitch Fix, Blue Bottle Coffee, and IBM. He's obsessed with math, and has a dog named Ravioli. You can find him on Twitter (@BEBischof) and LinkedIn.

Hamel Husain is currently an entrepreneur in residence at fast.ai, where he builds tools for data scientists and machine learning engineers. Hamel has previously worked at Airbnb, DataRobot, and GitHub where he built a wide array of machine learning products and infrastructure. Hamel has contributed to data and infrastructure tools in open source such as Metaflow, Kubeflow, Jupyter, and Great Expectations. Hamel was also a consultant for over 10 years, and used data science to improve business outcomes in the restaurant, entertainment, telecommunications, and retail industries. You can learn more about Hamel on his website.