Model Performance Management with Explainable AI

Build High Performance Responsible AI

Amit Paka, Krishna Gade & Danny Farah
Build Trust with Responsible AI

Watch over AI behavior.
Explain how decisions are made.
Have confidence in results.
Ensure outcomes are fair.

“We automate transportation and the lives of people are in our hands, model explainability is a must-have.”
— Chief Technology Officer

Fiddler helps teams monitor and understand their AI systems with confidence in even the most high-value and high-risk use cases.

See how Fiddler builds trust into AI
Model Performance Management with Explainable AI

Build High Performance Responsible AI

Amit Paka, Krishna Gade, and Danny Farah
5. **Implementing MPM in Practice** ........................................ 51  
   Model Performance in Staging Versus Production (Offline Versus Online) 51  
   An Ideal MPM Framework 53

6. **MPM and Responsible AI** ........................................... 57  
   What Is Responsible AI? 57  
   The Challenges of Responsible AI 58  
   How Model Performance Management Solves These Challenges 61  
   Model Governance and How MPM Fits into the Life Cycle 62  
   The Future of Responsible AI and MPM 63
Every industry, company, and consumer has been impacted by artificial intelligence (AI). According to The State of AI 2019: Divergence, 1 in 10 enterprises currently use 10 or more AI applications. According to Gartner, 75% of businesses are expected to shift from piloting to operationalizing AI by 2024. How many AI applications does your company currently operate?

AI has the potential to provide productive, efficient, and innovative solutions to our everyday problems, but it comes with its risks. We’ve seen multiple examples in the past few years of alleged bias in AI. One high-profile example was the Apple Card/Goldman Sachs scandal in 2019, where what started as a tweet thread with multiple reports of alleged bias eventually led to a regulator opening an investigation into algorithm prediction practices at Goldman Sachs. And this isn't an isolated instance; there have also been reports about Amazon’s biased hiring algorithm, racial bias in healthcare algorithms, and bias in AI for judicial decisions.

These issues might have been avoided if humans had visibility into every stage of the system life cycle. Part of that life cycle involves training a machine learning (ML) model to help in making decisions. In the model validation stage, teams could have unearthed instances of unwanted model behavior. With visibility into model performance online and offline, these sorts of unwanted behaviors can be detected and managed early on.

For each high-profile case that comes under public scrutiny, there are probably many systems that are silently operating and negatively impacting lives. One of the main concerns with AI today is that issues are detected after the fact, usually when people have already
been affected by them. This is a foundational problem in AI that needs a foundational fix. That's where model performance management (MPM) comes in.

Technology teams often utilize DevOps principles and application performance management (APM) to rapidly iterate software development and ensure high performance, respectively. MPM is a framework that draws on these learnings and applies them to the unique challenges of working with ML models to provide teams with control and visibility over the entire ML workflow. MPM is to MLOps what APM is to DevOps: it's a framework that offers observability and control over model performance.
Before the rise of the internet, software development life cycles were longer due to the nature of distributing and accessing software. Remember the days of having to purchase a CD or DVD and insert it into your computer to install software? The life cycle was a waterfall, and it used to take months (if not years) to perform actions like pushing a patch or an update to consumers that are taken for granted in today’s agile world.

After the internet brought about more connected software systems, monitoring communications and interactions between systems as well as between people and systems became harder and harder. This led to the rise of two frameworks that work well with each other to enable rapid iteration while maintaining high-performing systems: development operations (DevOps) and application performance management (APM). As systems started to advance and become more intelligent, we saw the rise of machine learning (ML) and artificial intelligence (AI) systems. But with this fancy new technology comes many risks and questions about what might happen if we let these systems behave in uninterpretable or unexplainable ways.

The rapid emergence of these systems also led to the development of new practices focused on machine learning and data specifically. With traditional software, it is easy to test if it’s working: there is
usually a discrete set of possible inputs and outputs. When it comes to sophisticated ML models and AI systems, however, it becomes hard to fully monitor and observe what is happening inside. Some models may draw upon thousands if not millions of inputs to make a decision. This is why we need a new framework focused around ML and AI that is data-centric and aims at producing high-quality models and systems that work with the goals of rapid iteration of DevOps and MLOps.

Such a framework would be similar to APM but would apply to ML models and account for the nuances and differences between these models and traditional applications. This framework can be called model performance management (MPM). Model performance is not only reliant on metrics but also on how well a model can be explained when something eventually goes wrong. This is known as model explainability. Just as APM aims at providing ops teams with more visibility into the entire application development process, MPM allows those teams to monitor models as they are being trained, validated, and deployed. In this chapter, we will discuss the need for such a framework before delving deeper into the nuances and stages involved in the ML life cycle and how MPM and model explainability fit into it.

DevOps and Application Performance Management

Let’s take a step back and talk about why DevOps and APM came to be and what they are.

DevOps

DevOps, or development operations, is a term used to describe the combination of cultural practices, business processes, and tool-chains that are used to achieve rapid iteration and high velocity. The goal is to innovate quickly and catch problems and issues as soon as they arise. There are many DevOps frameworks and tools out there, and each has its pros and cons, but at the center of it all is an overarching philosophy that unifies the practices in some ways across companies and domains.
One tenet of DevOps is system observability and, in turn, monitoring. Monitoring is achieved through the use of metrics and logs that enable us to track what is happening inside an application. There are tools that do monitoring, and there are tools that offer full observability, but there is no one unified framework that aims at achieving full visibility into application systems. Thus, DevOps is often combined with APM.

**Application Performance Management**

APM provides a framework for monitoring and managing software application performance through measuring metrics about the application, with the aim of catching issues as soon as they appear, if not before. The exact metrics used can vary, but, with applications increasingly moving to the cloud, ensuring that APIs remain performant and meet users’ real-time expectations is getting more and more complicated. We need to make sure our applications are always available when needed and can handle an increase or decrease in load without losing performance.

Your business might have APIs it serves to customers, or internal APIs for applications used by employees. You might also be using an external vendor’s APIs for your own business needs. Regardless of the details, you will need to ensure that the applications using these APIs are performing with acceptable responsiveness and fluidness so that users aren’t left waiting for minutes for a page to load. This comes down to how fast the API can respond to user requests. If an API is suddenly slowing down, an APM framework will automatically alert the responsible team so they can investigate it, find the root cause, and address it. Having full observability allows teams to rapidly identify and fix problems before they cause a loss in revenue for your business.

The benefits of APM are:

*Broad visibility*

Prior to the rise of the DevOps tools that we have today, monitoring infrastructure was limited in functionality and difficult to use. Software and hardware were monitored separately, creating disconnected experiences. APM introduces broad visibility across all aspects of the deployment cycle, unified under one framework.
**Increased observability**

When monitoring triggers an alert, teams address the problem with a fix. However, locating the source can be tedious and repetitive, especially when an alert was actually expected rather than a surprise. Going beyond visibility, APM provides additional root-cause debugging support, called *observability*. For example, with observability, an application's drop in performance might be attributed to a node in the cluster failing to process requests and not a problem with the code itself.

**Clear ownership**

APM enables DevOps engineers to manage the end-to-end lifecycle of production code through observability and clear ownership. Clarity of ownership allows for the designation of specialized individuals who can respond to issues rapidly.

**Iteration speed**

Ultimately, a streamlined process with clear stakeholders and availability of monitoring tools for broad visibility enables your teams to deploy and identify and fix issues quickly, thus reducing the time between iterations and maximizing the impact of each iteration.

**Application Performance Metrics**

The typical metrics that are measured for application performance are:

*API latency*

The amount of time the system takes to perform an action, typically measured in milliseconds (ms)

*Queries per second (QPS)*

The amount of times the API is invoked every second

These metrics are vital for an application team to monitor the scale and load their application is able to handle. They are also the basis for any automated alerts or trigger-based events, such as system failovers, and can feed into an automated scale management solution such as cluster resizing or node pool resizing (a way of managing the number of computers the API is using to process its requests). With cloud computing becoming more and more accessible, companies are migrating many of their applications and workloads to the cloud, making monitoring these metrics ever more important.
There might be a day—say, Black Friday—when your systems get flooded with requests, causing the application to crash. By simulating such traffic using an APM framework and measuring the latency the system will have at any given QPS before the fact, your team can capture any issues and design ahead to prevent any failures or performance degradations. This is known as load testing and can aid in scaling up systems.

Real-time streaming data, historical replay, and great visualization tools are standard parts of APM solutions today. The popular APM products don’t just offer insights into simple statistics but also have API and component integrations that are business-ready.

**The Rise of MLOps**

As applications have started to adopt more advanced decision-making algorithms and models, the management and tooling required to ensure high performance has become more complicated. With code-centric applications, it’s fairly trivial to test whether they are doing what they’re supposed to do. But ML and AI models are data-centric, which adds many challenges that traditional DevOps teams are not used to dealing with (like ensuring data integrity and validity). This is where MLOps comes in.

Machine learning operations, or MLOps for short, is an amalgamation of cultural practices around data and its usage, data science, and tools that all aim at rapidly iterating through versions of a model or running experiments rapidly to test different hypotheses. MLOps does not exist in isolation from DevOps, and they need to work hand in hand to ensure that all bases are covered and that any issues can be addressed and resolved with little effort from your teams. We will cover the ML life cycle and some aspects of ML in Chapter 3 of this report, but for now let’s look at how MLOps compares to DevOps.

**MLOps Compared to DevOps**

MLOps and DevOps both come from the same foundations and are focused on rapid iteration and continuous improvement. Without frameworks based around these philosophies, teams will have a hard time establishing efficient cycles and will yield few fruitful results. Both MLOps and DevOps propose validating and testing any system before it is released. However, they lack when it comes to
observability and monitoring. This is where APM shines for DevOps, as it offers full visibility into an application, allowing teams to find the root cause of a problem as soon as it becomes apparent.

ML/AI models are inherently black boxes; data scientists do not always know why and how a model works. But this lack of visibility can’t be solved with a one-size-fits-all solution, because each model is unique. Therefore, the testing and validation you need to perform to properly monitor a model require a lot of planning and work.

**Model Performance Metrics**

There are many different types of machine learning (ML) models, each with its own nuances and metrics that it needs to track. ML models are highly data-centric and rely on the quality and consistency of the data being fed into them. Here are some examples of the sources of complexity in ML models today:

**Versatility**

ML serves a wide range of use cases—anti-money laundering, job matching, clinical diagnosis, and planetary surveillance, just to name a few.

**Architecture**

ML algorithms can take many forms, which are easier or harder to interpret; they range from simple logistic regression and decision trees to advanced neural networks for deep learning.

**Variety**

ML models come in many varieties (tabular, time series, text, image, video, and audio), and often these modalities are mixed for certain domains, further adding to the complexity of tracking the performance of the model.

**Volume**

The rise of cloud computing has enabled teams to train multiple models with more data in parallel, which makes it harder to keep track of the training data.

Teams must be able to monitor sudden changes in data distributions that might lead to bias. Other issues might arise from the model being trained on too little or too much data, making it underperform. Tracking all the parameters that can affect a model’s performance is much harder than tracking code changes in a repository.
The metrics often used to track model performance are:

**Accuracy**
The percentage of total predictions that were actually correct

**Precision**
The percentage of positive cases that were correctly identified

**Negative predictive value**
The percentage of negative cases that were correctly identified

**Sensitivity/recall**
The percentage of actual positive cases that are correctly identified

**Specificity**
The proportion of actual negative cases that are correctly identified

Using some or all of these metrics, a model can be tracked from offline to online and have its performance measured against a simulated baseline. Since different models have different types of data and outputs, each will use different forms of these metrics. A tool that's commonly used to visualize them is the *confusion matrix*. Figure 1-1 illustrates this concept using a matrix with binary classification (the model only predicts either true or false). If the model has more output classes, then this matrix can be scaled to any number of rows and columns to analyze all the classes that are underperforming.

Using this matrix, we can see that:

- **Accuracy** = \( \frac{TP + TN}{TP + TN + FP + FN} \)
- **Precision** = \( \frac{TP}{TP + FP} \)
- **Recall** = \( \frac{TP}{TP + FN} \)

Although these metrics can offer a good measure of the performance of a model, it is not always possible to determine the “true” values of the predictions once a model is deployed. In production situations, models are often measured in relation to the business outcome they are trying to achieve. For example, a marketing model aimed at targeted emails might be measured by how many of the emails result in a purchase. In this scenario, the model is not measured by accuracy but by the business outcome it is able to
achieve. We will go into more detail about all of these metrics in later chapters.

![Ground truth (real)]

<table>
<thead>
<tr>
<th>Ground truth (real)</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Positive</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
</tr>
<tr>
<td>Positive</td>
<td>True positive (TP)</td>
</tr>
<tr>
<td>Negative</td>
<td>False negative (FN)</td>
</tr>
</tbody>
</table>

*Figure 1-1. A confusion matrix for binary classification*

**The ML Development Process**

Developing a machine learning model typically involves multiple steps, with various stakeholders working together and constantly iterating on data and models in the workflow. As the first step, data scientists define a business problem and turn it into a machine learning model, after gathering the necessary data.

Next, a model validator evaluates the model to make sure it works as intended and complies with company standards and government regulations. After the model is validated, it’s deployed into production by a machine learning engineer and starts to make predictions in business applications. Once it’s in production, the process of continuous monitoring, testing, and debugging starts, typically managed by a team of DevOps engineers, data engineers, ML engineers, and data scientists. And finally, teams analyze the model’s output in real time or in batches to check its performance, fix errors, and gather insights to train a better-performing model.

The stakeholders may vary from company to company because this field is still new and developing, but the core loop of iterating ML
models remains the same. Simply put, MLOps is this continuous feedback-driven loop of operationalizing machine learning models. To close this loop, MLOps requires a dedicated framework that can act as a centralized control system at the heart of the ML workflow. This framework, which tracks and monitors the model's performance through all the stages, is model performance management.

**Model Performance Management**

When compared to application performance, managing or overseeing model performance is a lot more data-intensive and requires a deeper understanding of the domain at hand. Without a unified framework or set of tools to use, your data science and machine learning teams might find it hard to deploy models at scale and maintain high performance. This is where MPM comes in. MPM is a framework for managing model performance throughout its life cycle and ensuring that teams have clear visibility into every stage of the ML life cycle. Figure 1-2 shows an overview of this life cycle. The feedback loop provided by MPM permits continuous model improvement.

![Figure 1-2. The life cycle of a model](image)
The MPM feedback loop augments each step of the life cycle in the following ways:

- While *training* a model and selecting features or data to train the model on, your teams can uncover hidden bias through under- or overrepresented groups in the data. Bias can also occur as a result of a change in the data coming in, known as *data drift*.

- After the model is trained, the MPM framework can *validate* and log any performance estimates to a database for tracking and auditing purposes. This can be used to generate summary reports for business, compliance, and AI ethics stakeholders.

- When a new model is ready to be *deployed*, it can be compared to a previous version using the same data to ensure that the new model is outperforming the old one. This is often called champion/challenger testing and validation.

- Once the model is launched to production, an MPM framework serves as a *monitoring* tool to observe model performance, drift, bias, and alerts on error conditions. An MPM framework can also be used to measure the performance of two models simultaneously, known as A/B or champion/challenger testing.

- Since MPM is a feedback loop, any insights generated from errors, performance issues, or *analyses* can be used to improve future models’ performance.

It is important to note that an MPM framework does not replace existing ML training, deployment, and serving systems. Instead, it sits at the heart of the ML workflow, capturing all the model artifacts, training and production data, and model performance metrics as well.

Because of the increased complexity within the ML space, teams often struggle with the following challenges, even when they have an MLOps framework in place:

*Inconsistent performance*

Since ML models are trained on historical data, there can be large differences between how the models perform offline (in staging) and how they perform online (in production, with previously unseen data).
**Lack of control**

Machine learning is often a black box. This makes it difficult to understand why and how the model is deriving an output from the given data to drive performance improvements.

**Amplification of bias**

Models can amplify bias in the data they are trained on, which makes identifying potential sources of bias very important. Furthermore, because of the lack of visibility, it can be difficult to be sure that the model is operating the same way on the test dataset or on real data as it did on the training data. Failure to identify discrepancies or imbalances between the historical and real datasets could amplify hidden biases, possibly violating corporate policy and resulting in customer mistrust.

**Lack of debuggability**

Inability to debug complex models could lead to low trust and model performance deterioration. It also prevents these models from being used in regulated industries such as finance and healthcare.

**Feedback loop**

Model degradation frequently drives the need to build a better model to represent the current business reality—but the operational insights that form the feedback loop are missing today.

**Difficulty in improving performance iteratively**

Challenges in tracking and comparing model behavior and performance across production versions make it difficult to understand how performance can be improved over time.

**MPM with Explainability for Full Coverage**

A high-performing ML model with around 80–90% accuracy might still have hidden biases and underrepresented groups in the data. In order to detect hidden biases in data, teams need to analyze the features being used and the distribution of data in those features. This should take place during the feature selection and model training stages, and usually leads to these issues being caught early on. However, as time goes on and models are set up to be automatically retrained, bias can start to creep in as certain groups of data become more or less prevalent. In order to account for this, an MPM
framework needs to monitor data distributions against a baseline to catch any arising drift that might cause bias.

Drift can occur due to the data being used to make the predictions changing or the data being used to train the model changing. In either case, MPM needs to track the distribution of data and account for the changes between what the model is expecting and what it is seeing. Any deviation above what is normal should be flagged for investigation and debugging. Chapter 2 will go into explainability in more detail and talk about the philosophy of Explainable AI.

**Benefits of Using an MPM Framework**

As mentioned previously, the MPM framework aims to bring visibility into the machine learning life cycle. It addresses several of the operational challenges in MLOps, such as inconsistent performance, loss of control, and lack of a feedback loop. But what does that mean for a business? Let’s take a look at some key business benefits that the MPM framework provides.

**MPM validates models before pushing them live**

Before deploying a model, it is crucial to validate it, whether required by law or not. This ensures that at launch, you have a high-performing, well-tested, and robust model. Since the model will have been built around a business hypothesis or problem, the validation process involves verifying with business stakeholders that it solves the intended problem once deployed and does so without hampering other aspects of the business. This requires the model to be explained in human-understandable terms. To answer questions like “How is the model making a prediction?” and “Are there any biases?” a model has to be validated. The validation can often be performed using techniques such as cross-validation to assess changes in the model's behavior. If the model passes validation, it gets deployed, and the business gets to start using machine learning with a clear understanding of the ML model's impact. However, this is just the beginning of the ML feedback loop.
**MPM continuously monitors model performance in production**

Machine learning models are unique software entities compared to traditional programs because they are trained for high performance on repeatable tasks using historical examples. As a result, their performance can fluctuate over time as the real-world data they receive as input changes. No company can know for certain exactly how a model will perform after it’s deployed. So, successful ML deployments require continuous monitoring to reassess the model’s business value and performance on an ongoing basis.

**MPM proactively addresses model bias**

Biases exist in the world, as a rule. Since ML models capture relationships from a limited set of training data, they are likely to propagate or amplify existing data bias and may even introduce new bias. Stories about alleged bias in credit lending, hiring, and healthcare AI algorithms demonstrate these risks. As a result, companies have been spending extra effort to measure and guard against bias by either ingesting or inferring protected attributes. But models can become biased after release as well.

Because bias can cause serious business damage, especially in highly regulated environments, identifying it in real time can save companies from costly penalties, such as financial fines or consumer outrage. As is the case with performance metrics, a data scientist typically needs to calculate bias metrics offline, which heightens the risk of errors (because they are not using production data). Moreover, bias issues can be correlated to other issues; for example, a spike in the number of male job applicants could cause bias in job candidate matching. Investigating causes of bias in a silo can cause data scientists to overlook such correlations, resulting in them spending time unnecessarily on debugging issues that could have been easily corrected.

**MPM detects training/serving skew**

Features used to train and serve the models in online and offline environments can differ. When this happens, the model will behave differently and needs to be debugged. By keeping the training data and prediction log in one place, MPM enables developers to detect these skews.
MPM explains past predictions

Imagine a bank is using ML to approve loans, and a customer files a complaint about a particular loan being denied. Because MPM tracks a model's behavior from training to serving, a risk operations officer can go back in time to reproduce the prediction along with the explanation (this is known as “prediction time travel”).

In the next chapter, we'll discuss how Explainable AI enhances the benefits of MPM even further.
Explainable AI (XAI) is a form of AI that aims at creating machine learning models that are, for the most part, explainable and/or interpretable by humans. XAI evolved out of the need to break open the black box of AI models to make them interpretable by humans, with the intent of minimizing the risk of unknown or unpredictable outcomes from those models. XAI is not only relevant for regulatory and legal reasons, but it is also an important tool for monitoring and managing model performance.

In this chapter, we will discuss who in your company (or outside it) might want an explanation of your models’ predictions, the many reasons you might want your models to be explainable and/or interpretable, and how different types of models can be explained. The focus will be on how explainability can be used to understand and thereby improve the performance of ML models.

### Explainability in Context

Before exploring XAI, let’s briefly discuss what AI is and its relationship to Responsible AI. AI is a form of intelligence demonstrated by machines, which is akin to natural intelligence demonstrated by animals and humans but without the ability to display emotions or consciousness. You might have heard of some advanced AI systems, such as the autopilot feature on planes or the autonomous driving capability of cars, that have been in the limelight of the AI community for the past decade.
AI systems can be as simple as the devices that turn on the lights as you walk into a room or as complex as an autonomous rover that can explore foreign planets. Think of FaceID on the iPhone as a form of AI that's used to verify your identity, or a fraud detection model that aims to catch fraudulent transactions on your bank account. The use of AI has expanded to touch nearly every aspect of daily life in unprecedented ways.

How Does XAI Fit into Responsible AI?

We’ll discuss Responsible AI in more depth in Chapter 6, but let’s briefly introduce it here to discuss how XAI fits into it. Responsible AI is a branch of AI that aims at ensuring all AI systems are designed and built with fairness, privacy, security, and explainability in mind. If your models are complex, they might need to be explained by your data science team to understand (and try to eliminate or reduce) any potential biases. Your models might also need to be explained for regulatory reasons if you operate in banking, insurance, or other high-compliance industries.

XAI goes hand in hand with Responsible AI’s focus on ensuring fairness and limiting bias. XAI can be used to detect higher-weighted features in the model and either remove them altogether or reduce their weights through negative biasing. Reducing bias or noise in the data can make the model easier to explain or interpret.

Who Needs XAI?

There are many stakeholders that might want you to explain your models’ outputs. These stakeholders fall into three main categories:

- Internal technical and business stakeholders such as data scientists, machine learning engineers, product owners, or executives
- End users of the AI system or product
- Public stakeholders such as regulators or investors

Internal teams are interested in continuously improving the model’s performance through understanding why it predicted certain things. From management to low-level developers, it is important to comprehend what your model is doing and how it will behave in the real world before letting it run wild and free. Model explainability techniques can help stakeholders understand the effect each input had
on the output (its weight) and how different classes of data behave. Models come in many forms, and later in this chapter we will discuss the different forms of explainability across different model types (including models dealing with text, images, and tabular data).

End users might also want to understand why a model predicted a certain outcome for them. Depending on your use case, you may want to transparently expose this explanation. In the case of a dispute, the explanation can be used to justify or arbitrate the outcome of the model; it’s an especially important tool in high-risk models, such as those in the healthcare domain, where a model’s decision can mean life or death.

Finally, we have public interests such as regulatory bodies and investors. These stakeholders are interested in ensuring your models are compliant with all the rules and regulations that might apply to your company or industry. In Chapter 6, we will discuss in more detail how Responsible AI as a whole is used to manage these stakeholders’ interests.

**How XAI Can Be Used to Manage Model Performance**

The different types of explanations discussed in the next section offer insight into the practical application of XAI. This section will introduce how XAI can be used within the context of MPM.

**Interpretability Versus Explainability**

AI systems often involve machine learning models, which are essentially the decision-making algorithms that predict an outcome or recommend a decision. These predictions may or may not be fully interpretable or explainable by humans. Interpretability and explainability are often used interchangeably in the context of AI, but let’s briefly walk through the subtle differences between the two.

Interpretability is the ability for a cause-and-effect relationship to be mapped—that is to say, a defined set of inputs produces a defined and deterministic set of outputs every time, without necessarily knowing why that is the case. To put this in the context of machine learning, the model’s output can be predicted from the inputs, but there is no explanation as to why this output may have been produced. For example, a model that predicts your income based on
your age group alone will be fully interpretable, since for the same age it will always yield the same result.

Explainability, on the other hand, is the ability for the effect of each of a model's inputs on its output or prediction to be explained. Such explanations are important for complicated models such as neural networks where there are multiple inputs fed into a black box, with little insight into its inner workings. In one approach to explain a model's predictions, the input values are perturbed slightly to see if there's an effect on the model's output. Using such techniques, we can gain insight into why the model is behaving the way it is. An example of an explainable model would be a loan default prediction algorithm based on a number of inputs, such as debt-to-income ratio and credit score. With this model, it will be hard to map the inputs to outputs, since that would yield too many combinations. In this case, it might be difficult to interpret the model, but the model can be explained by using XAI techniques to determine how much each input contributes to the output.

**Models That Are Interpretable by Design Versus Explainable After**

With the slew of machine learning frameworks and toolsets to choose from, generating model interpretations and/or explanations becomes a nontrivial task. Not all tools are created equal, and not all models are built for explainability or interpretability. Depending on the type of model or framework used, the outputs or predictions may be fully explainable and/or interpretable from the get-go (as is the case with simpler models like logistic regressions or decision trees), or they may need to have someone explain them post-training (as is the case with deep learning models or pretrained models). Models that are interpretable by design are often viewed as the least risky, since they are easier to explain. So where possible, you might decide to stick with these simpler models that your stakeholders can understand rather than invest resources into building a more advanced model that might not fully be explainable or interpretable and thus harder to get buy-in on.
Understanding Model Predictions

Understanding why different models are predicting what they are predicting is very important in order to analyze model behavior, debug issues, track performance, and understand bias. If you suddenly notice a dip in your model's performance, the first place to look is the prediction outputs. If there is a problem with the model itself, then model explanations can be used to debug it or to account for how it deals with new data in the system. Often models experience what is known as data drift (or model drift), where over time the data provided to the model changes slightly (or new data emerges), causing its performance to degrade if it is not updated. It's important to observe model predictions after deployment and check that they remain understandable and do not deviate from expectations.

Offline Versus Online Explainability

Model explanations can be generated either offline or online. As the name suggests, offline explanations are typically used during the development cycles of a model in order to understand the model better and hopefully build the best version for production use. In contrast, online explanations are based on the model's predictions after it goes into use. Generating spot online explanations can help you debug and find the root cause of operational issues by providing an additional layer of model understanding, while generating them continuously can help you monitor how model behavior is changing (for example, how the top three most important features change over time). This is why it is vital to have a robust MPM process in place that can flag events as abnormal or track model confidence in order to alert the team when certain thresholds are passed.

XAI in Different Domains

XAI is used in many different domains, and it is used in different ways to explain models across these domains. This section will discuss the three main types of explainable models that you might encounter—models for text, images, and tabular/structured data—and the different ways they might be explained.
Tabular/Structured Data Models (Highest Contributing Features)

Structured data models are typically opaque and are not easy to interpret if there are more than one or two inputs. As these models become more complex and involve more variables, they become harder and harder to explain. But since they output predictions based on a series of inputs, it can be beneficial to know which inputs are contributing the most to the model’s output.

To explain a structured or tabular data model, the inputs are sorted in terms of their weight contribution to the model’s prediction, and color coding can be applied to help visualize what is happening and to pinpoint outliers or abnormalities.

Once a model explanation is available, the team can use it to understand what factors are contributing to the model’s predictions, whether it has any bias, or if any features can be removed without impacting the model’s output.

Suppose we have a model that works by analyzing a combination of features about a financial transaction with the intent of detecting fraud. These features might include some information about the customer, describing their typical behavior. The output of this model predicts how likely a transaction is to be fraudulent. Once a transaction is flagged, precautionary measures can be taken, such as locking the account until the user either calls the bank or visits a branch. If a transaction is incorrectly flagged as fraud, it is possible that the model doesn’t have enough features to understand the full context or was not trained on similar examples. In either case, a model explanation can be used to figure out which features contributed to the model underperforming or not behaving as expected, and new models can be trained to address this issue.

Text/Speech Models (Sentiment)

The majority of text models perform some form of natural language processing (NLP) and involve a corpus or dictionary of words that are used to generate a score for the provided text. This score can be used in many ways to create a picture of the sentiment or emotion in a given piece of text. Sentiment analysis is the most commonly used form of NLP, and these models usually display a binary positive or negative sentiment along with a score.
In order to explain such models, we attempt to identify the words in the text that contributed most to the text’s sentiment score. This can help pinpoint words that are missing from the corpus used to train the model to produce the scores, or words that are misclassified. Being able to understand the model’s output will allow your team to easily identify issues and work to constantly improve its performance.

Suppose we have a corpus that produces the following sentiment mapping: {“love”: +1}. Now let’s assume we provide the model with the following text: “I love how much you hate this subject.” If I were to ask you if that statement was negative or positive, what would you say? To me, that sentence is pretty neutral: love and hate effectively cancel each other out. If we were to ask our model, however, it would think that the sentence was positive since it only knows that “love” has a score of +1. An explanation of the model’s output might look something like this: “I love+1 how much you hate this subject.” To fix this issue, we would want to modify our corpus to look something like this: {“love”: +1, “hate”: –1}. This would result in a total score of 0 and a model explanation that might look like the following: “I love+1 how much you hate–1 this subject.”

This oversimplified example is essentially how NLP models work at scale, but models are usually trained on huge corpora and might have custom domain-specific scores.

A real-world example would be a Twitter analytics tool that is used to monitor the overall sentiment and/or emotion of a given hashtag or account. To ensure the model is performing as expected, samples of predictions can be run through XAI to highlight the words that are contributing the most to a given score. Let’s now assume that someone in QA discovers a word that is highly negative in sentiment but is not being highlighted. This word can be flagged as needing to be added to the corpus along with the negative sentiment. Because of the fast-paced nature of the Twitterverse and the constant evolution of the language, with new acronyms and slang terms being introduced all the time, this kind of adaptation is often required.
Image/Video Models (Heatmaps)

Image models typically fall into one of two major categories: image classification, where one or more labels might be the output of the model, and object detection, where a box is usually drawn around an object in the image along with some labels or tags. In either case, the underlying algorithms are usually of the same type—a form of deep learning algorithm called a neural network—and can be explained in similar ways.

Convolutional neural networks (CNNs) are a common type of neural network used in image models. Without going into the nitty-gritty details, these models break down the image into smaller chunks and then decide based on those chunks which pixels are important or not important.

Similar to how in text models the words that contribute the most to the model’s prediction are highlighted, in image models, pixels are highlighted to show which areas in the image the model decided to focus on versus ignore. The actual layers of the model might not be directly interpretable, however, as these models are usually complicated neural networks.

Figure 2-1 illustrates what this overlay might look like for a model that is identifying a cat.

![Figure 2-1. A heatmap showing the pixels where different model iterations focused their attention](image)
As you can see, the heatmap highlights the pixels of highest importance or relevance to the model and ones that may have been weighted higher than the others. This can help your team analyze where in the images the model is focusing and whether it is picking up unnecessary features or areas of an image.

Models that operate on videos work similarly to image models but generate a prediction for each frame. Thus, an explanation would include an overlay on the video showing areas of interest to the model. When unexpected behavior is observed, the video can be stopped or slowed down and the frame of interest can be analyzed in order to tune the model’s performance.

Let’s go over how XAI can be applied in the context of a manufacturing defect model that is used to remove defective parts from the assembly line in a factory. The model would use an image of the part as input and the output would simply be a label, defective or not defective, alongside a score that shows how confident the model is in its prediction. These scores can be monitored over time to see if there is a sudden dip in confidence or whether the model’s confidence is misplaced (for example, if there is suddenly a huge spike in the number of defective parts that are later deemed not defective). In this case, as mentioned earlier, the model might be experiencing drift and will need to be updated or tuned. Using XAI, the team can analyze what parts of the image the model is focusing on and whether it is either picking up on unnecessary features within the images or not picking up on important features.
Now that we have discussed the importance of explainability, let’s dive into the machine learning life cycle to shed some light on what a model goes through on its journey from conception to production. There are various stages in the ML life cycle, and based on the complexity of your domain and/or the maturity of your system, you might already have incorporated many of these stages into your workflow. Depending on the size of your teams and the seniority of the individuals on those teams, each stage may be the responsibility of one team, or the entire life cycle may be managed by one team, or anywhere in between. MPM fits nicely into the model development, deployment, and monitoring stages and can help with monitoring and managing models that are being deployed. We’ll discuss this in more detail in Chapter 4; for now, we’ll focus on the various stages in the ML life cycle and what each one is about. First, though, we will briefly walk through the different types of analytics to illustrate how machine learning has evolved over time.

The Three Types of Analytics

As you can see in Figure 3-1, there are three types of analytics: descriptive analytics, predictive analytics, and prescriptive analytics. Machine learning models can be used for all three.

Descriptive analytics, as the name suggests, usually comes in the form of reports or charts where insights can then be derived by a human. This is the most commonly used of the three types, and you are probably using it in your company already. The aim of
descriptive analytics is to tell you what happened in the past, not to provide any recommendations for the future beyond what a human can interpret from the data. Machine learning models can be built to describe data too; this type of analytics is not limited to just charts and reports.

To take this a step further, a machine learning model can be built to predict outcomes of certain situations, also known as predictive analytics. This usually follows on from descriptive analysis, where a specific scenario can be modeled and the inputs can be mapped to create a virtual representation of that one real-world scenario in order to predict a certain output. Future outcomes can be predicted based on the incoming data at any point in time. Any previously unseen scenarios might not be able to be predicted accurately, though, and this is one shortcoming of machine learning today. This is also why MPM is crucial for staying on top of changes in the data or the model.

The final form of analytics is known as prescriptive analytics, and it is the least commonly used as it is the most difficult to implement. However, it potentially provides the most value to your business. A
prescriptive model will provide a prescription for a set of actions in order to achieve a desired outcome or target. Such models can be used to automate decisions or improve customer experiences, and they usually require multiple inputs to provide a prescription. Think of a scenario with multiple steps, where the prescriptive model tells you what to do at every step of the way. Depending on the outcome at each stage, the model can readjust and predict a new optimal path.

**Life Cycle Stages**

Now that you understand the different types of analytics to which machine learning can be applied, let's tie this back to the machine learning life cycle and MPM. Figure 3-2 illustrates the different stages of the ML life cycle when it is applied to a new problem. Note that different models involve varying levels of complexity and might not need to go through all of these steps.

![Figure 3-2. The stages in the ML life cycle](image)

As you can see, there are nine stages, which can be summarized as:

1. Problem definition
2. Data collection
3. Data processing and storage
4. Metrics definition
5. Data exploration and analysis
6. Feature extraction and engineering
7. Model training and offline evaluation
8. Model integration and deployment
9. Model release and monitoring
6. Feature extraction and engineering
7. Model training and offline evaluation
8. Model integration and deployment
9. Model release and monitoring

The life cycle starts when a problem is identified, and then it feeds back into itself. This is the iterative nature of model development at play. Once you or your team have been through this cycle a few times, the data that needs to be analyzed or collected might already be available, and it may be possible to skip some steps. The cycle repeats itself to keep up with constantly changing data and requirements. As we'll see, MPM closes the loop between the training, deployment, and monitoring of a machine learning model and can aid in detecting these changes and drifts and enable your teams to rapidly iterate.

Before we consider how it fits in, let's look at each stage of the ML life cycle in a little more detail.

**Problem Definition**

The very first stage in any machine learning model's life cycle is identifying and defining the problem it aims to solve. This problem can come in many forms and may be related to internal or external stakeholders' interests or needs. The problem needs to be defined with enough detail to be able to be understood by your engineering and data science teams. Something like “customers are not happy” is not a well-defined problem, but something like “customers have been having difficulty locating items they recently viewed” is. Depending on the goal of your team or company, the problem can be as simple as improving team collaboration or as complicated as predicting customer churn and prescribing actions to avoid it.

Let's consider an example problem in which we can follow through the different stages of the ML life cycle. Suppose you are the owner of an ecommerce startup and realize that you have no insight into what customers are doing on your site other than your bottom line and maybe some basic traffic metrics. You notice a dip in overall year-on-year sales, and you think it would be a good idea to check out any feedback customers have provided. You see a lot of comments from users saying that it’s not easy for them to find products they’ve viewed recently, and that it would be nice to have some sort
of recommendations based on the products they are viewing to help them find other relevant items easily. So the problem is that your company is experiencing a drop in sales due to a lack of user tracking and intelligent features. You think to yourself, “This can’t be that hard. I’ll just ask my team to integrate this into the application.”

At this point, you go to your team and ask them what needs to be done in order to understand customers’ behavior better and offer them a better experience through recommendations and personalization. They say they need more data in order to give you insights and build models to improve the customer experience, with the hopes of increasing sales. This is where data collection comes in.

**Data Collection**

Now that you have defined the problem and identified the data you need to collect, it’s time to implement all the necessary pieces to start collecting, processing, and storing the data. Before beginning to collect any data, it is important to set some policies around how the data will be used, who will have access, and what parts of the data will be accessible. This is usually the role of a CIO or a data governance officer. These policies will need to be transparent in order to instill confidence in your users about how their personal data is being used by your business.

Your data collection and privacy policies might include a mandate to obfuscate any personally identifying information (PII) so it is not visible to the analytics teams, or to only allow access to production data to privileged users such as team leaders or directors. Once all the policies are in place, the necessary tools can be selected and implemented in order to begin collecting the data for analysis and modeling. This will usually involve some changes to the application, such as adding frontend and backend components to track what your users are doing or adding functionality to use any future models into the workflow or website (such as the ability to identify a user using a stored cookie or other unique identifier).

Let’s go back to the example of the ecommerce startup. Your team now needs to implement a frontend tracker that sends an event to a backend service whenever a user performs some action on the site. This might be a search, a click on a product, or a click away from the site. No matter what the user is doing, it should be possible to track and store that information. A cookie can be stored on the user’s
browser in order to track the same user across multiple events on the backend.

Let’s say, for the sake of this example, that there are an average of 100 people online at any given second of the day clicking through your website or app. That would mean you would have to process and store around 100 events every second. Assuming each event is about 10 KB in size, the traffic would generate $10 \times 100 = 1$ MB of data every second. Given that there are 86,400 seconds in a day, this would mean you would need to process $1 \times 86,400 = 86.4$ GB every single day. If you tried to store that in an Excel file on your computer, you would probably run out of space within a week. You can see how that amount of data can easily add up and become harder and harder to manage. This is where data processing and storage solutions come into play, bringing the power and scalability of the cloud into your grasp.

**Data Processing and Storage**

You may have heard of the terms “data warehouse” or “data lake”—these are examples of data storage solutions. A data warehouse is a structured store of data (think of an SQL database), while a data lake is an unstructured file store, usually in the form of a filesystem-like service called blob storage. Data warehouses are used for analytical purposes such as sales reporting. The incoming data might be unstructured but will be converted into a structured form that is then stored in the data warehouse. Data lakes are useful for storing files such as images or raw text files that can be used as part of a machine learning pipeline.

In order to process large amounts of data, or “big data” as it is colloquially known, cloud computing tools such as batch or stream processing frameworks can be used to ingest the data into the warehouse or lake. *Batch processing* is when data is processed in batches at regular intervals or based on the batch size (think baking cookies in an oven that can only fit 10 cookies at once). *Stream processing* is when data is processed as it comes in and is usually implemented in cases where data needs to be analyzed in real time (think cookies cooking on a conveyor belt for mass production). A variant of stream processing is microbatch processing, where events are processed in small batches at very short intervals (say, every few seconds).
There are two common patterns for data ingestion for stream and batch processing: Extract, Transform, Load (ETL) and Extract, Load, Transform (ELT). In ETL, the data is transformed in flight (a common pattern in stream processing systems) and then loaded into its final resting place, either a data warehouse or a data lake. ELT, on the other hand, favors loading the data in its raw form and then using built-in or third-party tools, transforming it, and inserting it into its final location.

Depending on your requirements and the speed at which the data will need to be used, both ETL and ELT can be used. Loading the data in its raw form will preserve it in case there are any errors in the transformation. Some data might not need to be aggregated or analyzed in real time, while other types of data might need to be aggregated or analyzed in real time for tracking or compliance reasons. (Having redundancy doesn’t hurt in the world of data; you could always use both approaches for the same data.)

Let’s tie this back to our ecommerce example. You have around 1 MB of data being ingested every second, and around 86 GB each day. To handle this, you’ll need an ETL or ELT processing system (or a combination of both!) with some sort of obfuscation (removing any personal information) and some data quality checks. Any data that fails the check should be forked to another location for analysis or reprocessing. Once the good data has been processed and stored, it is ready for analysis, exploration, and (the exciting part) building some machine learning models!

Data that is processed in real time can feed into an analytics engine for the data science and analytics teams to consume. Data that is aggregated from the raw data every few hours, days, or weeks can be used by the business to track overall health and the key performance indicators (KPIs) of the different lines of business. But before you start doing all that, you should probably define some requirements for any analyses and potential models that are going to be built or developed by your team.

**Metrics Definition**

Now that you have started collecting data in order to better understand your business or problem, let’s talk about the requirements for metrics. It is good practice to prepare for any unforeseen circumstances by defining good metrics early on. These metrics might also
serve as the accuracy or performance measures for any models that will get deployed to production. This will not only aid in selecting which models to deploy and which to shelve, but also serve as a way for you and your team to compare different models against each other. They provide the basis for any benchmark or evaluation, and can come in the form of business metrics such as revenue or customer satisfaction (usually based on survey scores) or technical metrics such as model accuracy. Depending on your use case, you should spend some time with your team early on defining these metrics to help everyone align on the goals of the analysis or model.

To tie this to the ecommerce example, let’s recall the problem at hand. You noticed a dip in total sales due to some customer experience features that were missing. Any model that is built with the intention of improving the customer experience should in turn hopefully lead to more sales as well. Using this as a metric to track any model changes can be a good proxy to measuring how well customers are responding to new models. This is a useful business metric, but it does not help your team develop the model offline (they can’t tell which model will lead to more sales).

To remedy this, another requirement would be to have the ability to measure the accuracy of a recommendations engine. This could be used to predict how likely a user would be to click on any product that would be recommended to them if there were a recommendations engine present. This is a good metric to use offline: your team can study real user flows and measure how many of your users would be helped by the introduction of a recommendations or personalization model. This is a technical metric that can aid in developing and evaluating the model offline before it is deployed and the sales generation can be measured.

**Data Exploration**

Now that there are some metrics for the business side and your technical team, it’s time to start exploring the data. It is also possible at this stage to start developing hypotheses for predictive or prescriptive models that can validate or invalidate the possibility of using the collected data for a given model. This is usually done by subject matter experts, such as specialized data analysts or data scientists, and relationships between different data points can be mapped to create a better picture of the system.
During this stage of the life cycle, it is usually descriptive analytics that is the most useful, as it is easier to translate findings from the technical side to the business side this way (think dashboards and reports). Reports can be generated to understand different types of data groups (such as customer types) and their intrinsic characteristics. Descriptive analytics can also be used to inform areas of improvement that can be addressed by a machine learning model that understands the domain and can eventually evolve from descriptions to making predictions or even offering prescribed actions.

In the context of our ecommerce startup, to show users items they have recently viewed, the data for each user can be looked up based on their identifier. Once you understand what a few users are doing individually, you can try to create relationships between different users to draw out some patterns.

You can generate user personas for the different types of users that are using your site based on the products they are looking at and buying, as well as other features (such as frequency and location, to name a few). This information can then be used to provide more tailored experiences to different users. The term “features” here is used to describe a feature of a person shopping on the site, and in the context of an ML model, it means almost the same thing. A model feature is a data point that is known to the model at the time it is trained and can be used to represent how the input to the model would look at the time of making a prediction. (Think of knowing the persona of a new user—that’s not possible, so would this be a good feature for a model?) This brings us to the next step, feature extraction and engineering.

**Feature Extraction and Engineering**

The features that are fed into the machine learning model (user personas, for example) can be selected by the data science team according to the inherent nature of the domain, or they can be automatically generated by some optimization that aims at maximizing a set target outcome (such as accuracy or performance). This is the role of feature selection, extraction, and engineering.

Once you’ve identified the features and/or targets, the next step is to set up an automatic way to generate these features and store them. This can be done in real time, as the data is coming in, or
retroactively in batches every few hours, days, or weeks. It depends on what your domain is and what cadence makes sense for the type of model being used. If the feature is the average monthly spend of a customer type, then it only makes sense to generate that feature once every month. If, however, you want to see the daily spend, then this feature would need to be extracted once a day. Features are usually the inputs to the machine learning model, and depending on how complex your system is, you might have a handful (say, 5–10) or an incomprehensible amount (say, a few million) of different types of features.

As described in the previous chapter, these features may be known up front, or they might be determined by the model itself and require some explanation or interpretation after the model is trained. No matter how complex or simple your model is, you will need to train and evaluate different hypotheses to produce the best possible model. With advanced models, like deep neural networks, the features might be images with labels, or bodies of text with some tags. These are fed into the model in order to show it some examples so it can form an understanding of the system. The model can use the examples provided to select the best features that identify what the target label is. Think of a model that is identifying cats in pictures, where the shape of the ears can be picked up as an identifying feature. This is where model training and offline evaluation come in and can help in building models that will hopefully make it into production one day.

**Model Training and Offline Evaluation**

Once you understand your data (for the most part) and have some features that represent your system, you can build and evaluate some models that can help you predict outcomes or analyze data. For this, you’ll need to set up a collaborative environment where your data science and analytics teams can work together to create the best possible models.

The first step in model training is to split your data into two sets, for training and testing (80/20 and 70/30 are common train/test split ratios). This can be done by randomly separating out some of the data into a test set that the model will not see during training, but that has the target outputs labeled to allow you to measure the model’s performance. The target can be in the form of a label or a prediction. Once the training and test sets are ready to go, you can start...
training models and evaluating them against the test set. The metrics defined in the requirements stage can be used to compare the performance of different models or frameworks. Generating multiple training and test splits, which is commonly referred to as cross-validation, can help in testing for model sensitivity or identifying bias in the model that may arise from over- or underrepresented data. Once a stable model with desirable results is developed offline, it can move on to integration and deployment.

Let’s take the example of building a recommendation system model for our hypothetical ecommerce startup. A recommendation system is usually a banner that sits somewhere on the page (or along the user journey) that shows other products the user might be interested in or has recently viewed, or that others have also looked at or bought. A common method used to create a recommendation system is collaborative filtering—without going into too much detail, this method works by creating a sort of hive mind that can understand patterns and relationships between different items listed on the website based on users’ behaviors.

For simplicity, let’s assume there are only three products (A, B, and C) listed and only two customers (1 and 2) shopping. If customer 1 looks at products A and B but customer 2 only looks at B, then customer 2 will be recommended product A because customer 1 also looked at A. Now if customer 2 also looks at product C, then customer 1 will be recommended product C as well. You can see how slowly, with a few million users and a few hundred or even thousand products, the model would start to understand what products are intrinsically related based on user behavior alone.

Going back to our ecommerce example, using the data we have collected from users so far, we can split the users into a training set and a test set. The model can be trained on the training set and evaluated against the test set based on how many of the items that would be recommended were actually actioned on by that user. Using this benchmark, the model can be trained and tuned in order to achieve a desirable level of accuracy. As mentioned earlier, not all metrics can be evaluated offline (in this case, sales generation, for example) and some will need to be live tested, which is usually done through an A/B test or a champion/challenger test. That means moving on to the model integration and deployment stage.
Model Integration and Deployment

Once a model has been developed and evaluated against offline metrics to show its viability, it will need to be integrated or deployed into your application, workflow, or website. Integration involves adding code or support for the model to operate within your website, application, or workflow. Deployment involves launching the model in a state where it is ready to be released. Depending on the architecture of your system and where the model will be called from (edge versus cloud, for example), the specifics of how this is done may vary. This integration and deployment process might also be referred to as a continuous integration/continuous deployment (CI/CD) pipeline in some contexts.

In some cases model integration and deployment is as easy as merging a pull request (CI) into a branch on a repository, which will then trigger a deployment job (CD) contingent on some automated tests passing. It can also be as easy as pushing a button in your cloud provider or service provider's console (e.g., AWS SageMaker or Google AI Platform) that will update the parameters or version number through a UI. In other cases, however, model deployment might involve releasing new versions of software to your users or device fleet. If downloads aren't automatic, then some users might be left out of the updates until they decide to update the application manually on their devices.

Let's take a look at how we would integrate and deploy the shiny new recommendation system we just built for our ecommerce startup example. First we would need to add support for our frontend to talk to the model on the backend. We would also need a mechanism to split traffic between users to allow for an A/B test or a challenger/champion experiment. We would need to test whether having a recommendations banner would help increase sales or even just user views of items. Once we have the integration ready, we can deploy our model and stage it for release and monitoring, which is the final stage of the journey to production.

Model Release and Monitoring

After a model has been integrated and deployed, it can be released and monitored. The release process might involve changing its configuration to point to the latest deployment of the model or simply pushing a button to update to that version. However complex or
simple this process is, it should be done by someone who has the authority and responsibility to perform this task.

After a machine learning model is released, there is no guarantee that the underlying system it is representing isn’t going to change or there isn’t going to be new behavior that needs to be accounted for. This is where monitoring comes in and ties in to MPM. MPM can offer a lot of insight into how models are performing and which models need to be updated. The model version, parameters, and target user base can all be modified during the release stage according to the identified splits in the previous stages. Since the model can target different groups for experimentation, monitoring can be applied to these different sets, based on their identifiers, in order to monitor the performance across each. Experimentation can help validate or invalidate certain theories your subject matter experts might have about your domain.

Let’s say you were looking to increase the sales in your organization and wanted to test out different models to see which would generate the most profit. Monitoring would be the most important part of that experiment, as it would aid in tracking the different models and how the users are reacting to them.

To close off the life cycle of ML models, let’s talk about what happens once you have the recommendation system on your ecommerce site. Once the model is released to a portion of the users, you start to notice that the users with the recommendations banner are interacting with the website a lot more and in turn end up buying more products as a percentage of their group. This is great news! You have validated that a recommendation system would increase your sales.

But the journey isn’t over yet. Now it’s time to roll out the model to everyone else and start thinking about a better version—perhaps a version 2.0 that challenges this model or maybe a version that is a little more personalized, say, using deep neural networks. Multiple parallel iterations of this life cycle can be occurring at any given time, and this is a good sign of a healthy data-driven organization.
How MPM Fits into the Life Cycle

Let’s close this chapter by discussing where MPM fits into the ML life cycle and what it can and can’t do. MPM is a philosophy and a set of tools and frameworks (analogous to APM within DevOps) that aims at measuring models in terms of their performance. The management comes in the form of intervention, such as a new model being built, or a model being decommissioned and sent to the model graveyard. As shown in Figure 3-3, MPM starts at either stage 6 or stage 7 of the ML life cycle, depending on whether the features are relevant to the model’s performance or not. As mentioned in the “Data Processing and Storage” subsection, any data quality issues should be captured long before the data gets to the model training stage.

![Figure 3-3. The stages in the ML life cycle](image)

If the data being used to predict an outcome is causing issues in the model, then MPM suggests that the incoming data be stored for the team to analyze and dig deeper. This analysis might include considering parameters such as who the user or caller of the model was, what the input data was, and what the model predicted. Using XAI techniques, the model can be explained to understand any undesirable behavior in order to fix or correct it.

Undesirable model performance would usually originate at the user level, where a bug or issue is flagged by your QA team or automatically detected. Once this happens, the team in charge of the model
may need to escalate and fix the issue, depending on how mission critical it is. If the issue does not appear to originate with the model, they can trace the path back up the pipeline until the issue is detected and resolved. If the problem happens to be that the model is out of date or is not accounting for new data types, then it might be necessary to retrain or update the model. As more and more models start making their way into production, you can see how it gets harder and harder to manage and monitor them.

Having an MPM solution in place can help with monitoring and tracking model performance, and it gives you almost full observability into your models out in the wild. Combining MPM with the previously discussed XAI gives you a full picture of what is going on with your models and why. This will never replace the know-how and expertise of subject matter experts, but it can serve as a useful tool to help them manage models efficiently at scale and help you ensure your models are built responsibly.
CHAPTER 4

MPM in the ML Life Cycle

MPM serves as the glue that binds the model training, deployment, and monitoring stages together. Without an MPM process or system in place, your team will have a hard time finding the root cause of model performance issues. Whether the problem is caused by bad data or by the model being out of date, your team should be able to pinpoint its source. In this chapter, we'll look at how MPM can help provide full visibility into any issues that may arise when training, deploying, and monitoring models.

The ML Feedback Loop

The machine learning life cycle is cyclic, due to the ever-changing nature of the world. Since models serve as general representations of real-world situations or scenarios, they need to constantly be monitored and updated to deal with new or previously unseen data and circumstances. Closing the feedback loop is a concept taken from control theory where the expected outputs and measured outputs of a system are compared in order to tune the system's performance and achieve optimal results with minimal or zero error. To explore this concept further, let's first take a quick look at what control theory is.

Brief on Control Theory

Control theory is about the control of dynamic (engineered) systems, typically using feedback mechanisms, in order to achieve optimality. Optimality is based on the context of the system being...
controlled and can be deterministic and exact or within a margin of acceptable error. There are two types of controllers, open loop and closed loop. To relate this to MPM, the expected outputs and metrics of a model need to be tracked with their actual outputs and metrics over time to ensure that the model is achieving optimal results.

MPM closes the loop of outputs in order to correct any errors in the future and maintain a state of optimality (something we will discuss later in this chapter). Often in control systems, the analysis is done in the frequency domain as opposed to the time domain. This means that instead of analyzing when something happened, a control system analyzes how often and how intensely something happens. Model error is a good thing to measure, as it can show how often something is not going as expected rather than when this is occurring. This measurement helps in pinpointing the highest-priority issues to fix (i.e., those that have the highest impact on your business).

Open-loop versus closed-loop controllers

An open-loop controller is one that does not rely on the measured output for regular operation. This is the case in highly predictable systems that have a discrete set of outputs based on the inputs. Looking at Figure 4-1, we can see how the flow of inputs to outputs is linear and easy to follow.

![Figure 4-1. An open-loop controller](image)

Think of the timer on a microwave that heats up food. It only knows how long it needs to go for and has no need to detect the temperature of whatever’s inside it to adjust that time. The reference here would be the time the food needs to be heated for, and the system input would be to turn on or keep on the microwave based on how much time is left. Once the reference signal runs out (the timer ends), the controller will turn off the microwave.

A closed-loop controller, as can be seen in Figure 4-2, is one where the system relies on the output to achieve its desired state. The reference signal and the measured output are compared in order to tune the controller’s output (which would be the system input) and try to
minimize the error between the reference and the measured output. An error function is used to compare the two signals, and it sends the error as an input into the controller. The controller then, using its internal model, tweaks the system inputs to try and achieve the desired output.

Figure 4-2. A closed-loop controller

In Figure 4-3 we see an example of a closed-loop controller: the power steering system on a car. The desired direction of the car is implied by the direction the steering wheel is being turned in, and any undesired resistance is given as feedback to the user in the form of the steering wheel rotating in the direction of the undesired motion. The controller has an internal model mapping the rotational distance of the steering wheel to the desired direction of the wheels to turn the car.

Figure 4-3. A closed-loop controller in a car’s power steering system

There might be some background noise that causes the steering wheel to turn without any input from the driver, maybe due to a bump or a hole in the road. The power steering system receives the reference signal (in this case, the rotation of the steering wheel by the driver) as well as signals from sensors in the car (the measured output) and uses these to determine how to steer the car to achieve the user’s desired result, as well as to return some feedback to the driver in case any undesired movement is detected.
Control Theory and MPM

Similar to how control theory operates on feedback using sensors, MPM acts as a feedback mechanism for your ML system. In the context of machine learning, the sensor in charge of giving feedback to your ML team is the MPM framework that is tracking the outputs of the model. The user (or human-in-the-loop) is the data scientist or ML engineer constantly monitoring and trying to update the model. The feedback is all the data collected from the model, as well as some metadata. The error function would be part of the MPM system that can flag any errors based on the reference inputs from the team or system. If the error function shows that the model is producing undesirable results, the ML system can then be investigated with the aim of improving the performance.

Thinking of the machine learning life cycle as a dynamic system allows us to understand the need for closing the feedback loop and tracking error (divergence from the desired references). References in the machine learning world are often the target labels used for training the model, which might not exist in incoming data since the expected prediction of the model is not known at the time of the prediction. Tracking the frequency of changes, as well as deviations in the model's output, might help reveal any model drift or data drift issues. These can then trigger the team to investigate and try to correct the error in the system. Often this process of validating the model's performance relies on some user input, such as ground truth labels or expected outputs. This validation process can be achieved through sampling a small set of the data and applying these labels, then comparing the output of the model to this expectation or ground truth.

MPM in the Training and Deployment Stage

Now that you understand what model performance is, let's talk about how MPM fits into the ML life cycle stages and offers an easy way to manage processes to continuously monitor and improve model performance. MPM can start at the time the features are being extracted or engineered, or it can start at the moment the model is trained. This depends on the complexity of the input data and the effect of data drift on the model's performance: a model that has very few inputs that rarely deviate from the norm would not need its features tracked since the inputs will always be predictable.
and have a low risk, whereas a complex model with multiple inputs that are completely variable and can change over time would need to be tracked.

**Using XAI to Understand Model Bias**

During the training stage of a model, bias might be hard to detect or prevent without trying to understand how the model might behave with inherent bias. In order to mitigate possible bias, XAI (discussed in Chapter 2) can be used to explain why results might be biased to a certain subset of the data. This can be caused by unbalanced training data or by the model being set up with inherent bias. Detecting unbalanced training data can be done by analyzing the distribution of the different labels and input data points to detect any underrepresented or overrepresented data. The data can then be undersampled or oversampled in order to reduce or eliminate any bias. If, however, the bias occurs due to the model itself being biased, this would show up through testing the model and using XAI to understand the model’s predictions.

Once the bias has been reduced to a negligible amount or eliminated completely, the candidate models can then be evaluated against the target labels to find the best one to release. Sometimes the best-performing model is also the most biased, and using XAI in conjunction with MPM is key to detecting these issues early on. (When you develop models that are not sensitive to bias, of course, you do not need to worry as much about detecting bias.)

**Choosing the Best Model to Release**

Choosing which model to integrate into your application or workflow and release to your users after optimizing performance and dealing with bias might be the most difficult decision, as models can sometimes be unpredictable, and any unpredictable behavior may translate to business risk and possible liability (depending on your line of business). The best-performing model offline might not prove to be the best-performing model online. This can be due to a high level of sensitivity—any model that you choose to release should therefore be analyzed for its sensitivity both to the training data and any unseen prediction data. As discussed already, this is done through splitting the historic data into training and test splits, where the test set is not seen by the model during training, but the expected prediction is stored alongside the test data.
Once a model is deemed to be ready for production, it can be versioned, integrated, and released. This versioning will help with many downstream analytics dependent on tracking the lineage of the model as well as its outputs. Any discrepancies between the model in offline and online environments can then be identified, and the performance of the model can be optimized to achieve the desired results.

**MPM in the Release and Monitoring Stage**

After a model has been approved for release, there are a number of measures to take to ensure that it can be confidently rolled out to all your users. MPM suggests that the model log all its predictions and input data alongside the model version and other metadata to allow the team to monitor it. Any issues that happen in the model can be detected by tracking the predictions being made and whether they begin to deviate far enough from the norm to trigger an investigation (known as model drift), and the root cause can be investigated using XAI. Models can also be compared in a live setting (known as A/B testing or challenger/champion testing). More advanced models might need to have more than two variations compared in order to ensure the highest-performing model is promoted; this form of testing is called multivariate testing.

**Detecting Model Drift**

Model drift occurs when the data feeding into the model begins to change or deviate from what the model expects, resulting in changes in the model output. When this occurs, it might be hard to detect since the model outputs are not usually stored alongside some metadata for analysis. Using an MPM system or process ensures that the model outputs are stored and accessible by your data science or operations teams. On an average day, the expectations of distributions across the different input groups should be within an acceptable margin of error. If these values begin to deviate over time, the analysis of the outputs can show any drift and allow your team to pinpoint and fix any issues.

One issue might be that the model needs to be seasonally updated due to the seasonality of your business. Another case would be that the model itself was not trained to account for all scenarios and is therefore producing unexpected outputs for unseen data. This can
be caused from underrepresentation or overrepresentation of certain subsets of the data. To remedy this, your team can add in paths for unseen data, such as triggering a customer support ticket or referring the user to documentation in order to try to remedy the problem. Whether the issue is detected by the system or by users contacting your support team, the MPM system can be used to replay and analyze the predictions in order to update the model to remedy the issue.

Finding the Root Cause Using MPM and XAI

Once an issue has been identified by either the MPM system, a customer, or your QA team, it’s time to find the root cause and address it. Your team might be tempted to hastily attempt to remedy the issue by applying a quick fix or patch at the application layer without looking into the black box that is the machine learning model. But using MPM, they can identify and fix the root cause of the problem by isolating the issue and determining an appropriate remedy that has no impact on other parts of your system.

In order to mitigate the risk of customer satisfaction or compliance issues, your team can test out changes or fixes on a small portion of the data, or replay the past few days’ worth of live data in a simulated environment to check if the changes would have resulted in the expected behavior. Without an MPM system to store all the predictions alongside the metadata, it would be hard to replay data and run historical tests on any patches or fixes. Once the fixes have been validated on historical data, they can be released to your users. In order to mitigate any unforeseen risk, the model can be released to a small portion of your users first, and the results can be monitored to ensure the fix is safe to roll out to the rest of your users.

Live Experiments

Validating whether a new model (or an updated model) is going to accomplish the desired outcomes might be hard to achieve without mechanisms to mitigate unforeseen risk. There are many methods you can use to identify potential issues or defects early on in the process. Private testers or QA teams can capture these issues and log them for your engineering teams to address. No process is 100% perfect, though, and there will always be unknowns. So, to further mitigate risks, live experimentation can be done to minimize any damage by targeting specific subsets of incoming data for the new
model to receive. By splitting the data in this way, the person track‐
ing the experiment can monitor the results to validate any hypothe‐ses or fixes that might have been applied. Once the model is
validated on a small portion of the data, the ratio can be increased,
either straight to 100% or gradually in order to account for edge
cases or low-volume data points.

**Random sampling versus targeted splitting**

When conducting an experiment, it is important to keep in mind
the data split that will be done in order to achieve the desired ratios.
Sometimes you might need to validate that a very specific edge case
has been fixed and that this fix does not cause any other issues. In
these cases, you might want to target the fix for the users that fit into
that edge case and validate that it works before going on to validate
that it does not cause any issues for other users. Random sampling
can be used for testing all users by ensuring an even split of samples
across the two (or more) experiment targets.

There are many methods to randomly split data to ensure no biased
targets occur. A common method is to use a hash of non-identifying
fields in the data (metadata) and randomly split that set of hashes
into target groups. Since the data being used to generate the hash is
not unique to that row, the samples will be randomly distributed
across the groups. This ensures that the “randomness” of the selec‐
tion process for the target groups contains no bias toward selecting
one specific group over the other based on any of its identifying fea‐
tures. This method applies to cases where the individual predictions
are not related to a specific user, but sometimes you might want to
split the users into groups.

If you want to target users rather than data points, you can first
assign a random universally unique identifier (UUID) to each user
and then sample the IDs using the same method as mentioned pre‐
viously. You might need to generate these UUIDs for each user, or
you might be able to use an existing field (e.g., a user ID field). In
either case, these IDs are usually stored on the user’s device or
attached to their login information in order to better track the per‐
formance of the different models across the different target user
groups.
A/B testing

Now that we’ve seen how the models can be split into target groups, let’s talk about the simplest type of live experiment we can perform: the A/B test or challenger/champion test. The A and B in A/B testing just refer to two models; they can be two completely different models or slightly modified versions of the same model. In either case, there is usually one model that is already in production (assuming this isn’t the first experiment you are running), which would be the champion in a challenger/champion test. The challenger is tasked with outperforming the champion and taking over its role as the champion model. Once a challenger that has been shown to be superior to the champion is promoted to that role, it awaits challenges from future models.

Multivariate testing

Sometimes you might have many different variables that need to be tested at the same time, or many different variations of models that you want to experiment with. In these cases, you’ll need to have multiple target groups vying for the glory of champion. Once the different groups and variables have been released, without an MPM system in place that is keeping track of all the different things happening at the same time, it might become extremely hard to analyze the results. Ensuring that the MPM system also tracks the metadata of each model version, as well as the users’ UUIDs, becomes a full stack effort requiring the integration of these metric trackers into your application or system.
Closing the feedback loop of ML using MPM is an important step in ensuring constant monitoring and improvement of your AI systems. There are a number of methods or tools that can be used to implement an MPM system or framework to achieve this in practice. Before we look at the different options, let’s start by discussing what makes a good MPM framework.

Model Performance in Staging Versus Production (Offline Versus Online)

As illustrated in Figure 5-1, MPM serves as a way to ensure that you have full visibility into your model and that any predictions it makes are stored along with the input data and the model version. This allows you to replay the predictions later in order to try and fix any performance issues, possibly using XAI to help understand them.
When training a model offline, as was mentioned in the previous chapters, the data is split into training and test sets and the performance is measured with the objective of tuning it to achieve the optimal model. During training, the target objective (label or expected output) is used to measure how accurate or precise the model is and thus how well it is performing. These target metrics are usually identified and set early on in the machine learning life cycle. The offline evaluations serve as a measure to estimate how well the model will perform in the real world.

Offline metrics can only show how well a model will perform on historical data, however, and might not be a good measure for how well the model will perform online. It’s possible that bias that exists in the historical data might not be accounted for, and the problems might only become apparent after the model has been deployed. This can have a large impact on your business. In order to prevent
such scenarios from happening, it is important to continuously monitor model performance issues during training and to stress-test the model in order to catch any problems before it goes live.

**Online Model Performance**

Once a model does go live, it becomes harder to track its performance as the target labels or outputs from the model are not known ahead of time. One way to track the performance is to measure any changes in the predictions being made and whether they suddenly vary outside a normal range. If that does happen, it could trigger a model performance issue. Another way to monitor performance in a live environment is to use indirect measures, such as increases in sales or customer satisfaction. These proxy measures might not reveal any direct issues in the model but are a good way to track complex models that are not performance critical. For models that are performance critical, a QA sample can be taken and a human evaluator can measure the accuracy or performance of the model and track any issues for the data science and machine learning teams to look into. Issues might also arise due to customers reporting a bug or asking for an explanation as to why a certain result was given to them.

**An Ideal MPM Framework**

When thinking about an ideal MPM framework that will help in managing models, we need to start with the KPIs and metrics that will help us in tracking performance. An MPM framework implements the MPM process in order to provide continuous feedback to your team and allows for transparency into the process. This framework should track the performance of the model from the time it is trained until it is deployed and released.

An ideal MPM framework needs to be able to aid in pinpointing the root cause of model performance issues. As such, it needs sufficient visibility into your system and the outputs it is producing. Here are some of the criteria that define a good MPM framework:

- It tracks the *model version* alongside the *training data* used.
- It stores all *metadata* about predictions alongside the *predictions* themselves.
- It provides *automated alerts* when set *thresholds* are reached.
• It has the ability to *replay predictions* in order to *compare models*.

• It has a *graphical interface* to allow monitoring of the health of the system.

• It allows the team to *analyze the raw data and metadata* to uncover undetected issues.

• It *utilizes XAI* to *uncover hidden bias* in the data.

This list is far from exhaustive and can be expanded upon based on your needs. Once the MPM system requirements have been identified for your team and/or organization, it is time to select the tool(s) to implement such a framework. The following sections offer some examples of tools, frameworks, and platforms that can be used to implement MPM in practice.

### Integrated Tools Provided by Cloud Platforms

There are many large cloud providers and platforms today offering a range of products and services in the space of data science and data engineering. The three biggest players in the cloud computing space—Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP)—all have some model tracking and performance management capabilities built into their offerings or available as supplementary add-ons. The systems for tracking model performance are usually integrated into their AI or machine learning services and give users advanced visibility into model performance. If you already have a cloud provider, using the tools and services they offer will allow your team to integrate them into an MPM framework with little to no overhead.

If your team is considering moving to the cloud (or changing providers), you should evaluate the different platforms and sets of tools depending on your specific needs and use cases. One platform might offer superior model tracking compared to another, and each should be evaluated on its merits. Going with a multicloud approach might also be of benefit, as it can offer the best of all worlds with direct comparisons of which platform meets your needs best. This industry is still developing, and there will be something new offered by a competitor every year, so it's best to stay agile and have the ability to transition if need be.
Specialized MPM Software

If you’re not yet ready to move to the cloud, or the offerings of the cloud providers do not meet your team’s needs, then adopting some specialized software for MPM might be a suitable option. If your system is sitting behind a private network or firewall with no access to the internet, then it might be hard to integrate with some services that operate over the internet. There are both open source and software-as-a-service (SaaS) MPM offerings. Open source offerings can be run within your infrastructure and offer more configurability; examples include MLFlow and Seldon. SaaS offerings, on the other hand, rely on an active internet connection in order to interface with the provider’s services or your account within its environment. An example of a full-featured SaaS MPM solution is Fiddler, a system that allows users to track its models’ performance from training all the way through to deployment, redeployment, and finally retirement. It allows model bias to be caught early on in the training and validation stages in order to prevent unwanted behavior after deployment. Different versions of the model can be tracked and analyzed, and model predictions can be replayed to simulate traffic.

Whether your team chooses an on-premises or cloud native solution will come down to a multitude of factors. If latency and scale are concerns for your business, then it might be beneficial to adopt cloud native systems. If cost and security are more important to you, then an on-premises solution using an open source offering might be the right choice. The decision will impact the system’s performance in a production setting, and care should be given to your choice. A hybrid approach can also be used if data residency is a requirement in your industry or business.

Custom-Built Systems

If the needs of your team are not met by any of the cloud platforms or specialized tools you evaluate, it might be worthwhile considering building your own MPM framework. There are many open source tools and frameworks that can be leveraged for this task. One software stack that can meet the needs of MPM is the Elasticsearch ELK stack (Elasticsearch, Logstash, and Kibana), which is a full suite of logging, storing, and analyzing tools and services that can be used in a multitude of use cases and in different configurations. This
is open source software with options for managed services provided by many vendors, including Elasticsearch itself. This means that you can choose to manage your own infrastructure or allow a vendor or service provider to manage the infrastructure for you. Managing your own infrastructure adds a lot of overhead in terms of resources needed to maintain the MPM system but offers you full configurability.

ELK is just one example of a stack that can fit the needs of MPM. When possible, using prepackaged solutions or services might reduce overhead and turnaround time, and it has the advantage of a dedicated team (the vendor or service provider) being available to deal with any defects or issues. Using services also allows your team to focus on building workflows or model performance checks rather than solving the problems of tracking model performance.

You might already have a set of tools and services that can be used as the building blocks for an MPM framework but lack the end-to-end coverage. In this case, building a workflow that implements MPM within your team’s processes might be the best solution. Integrating the data feeds from the various steps into a central data store would allow for an analytical layer to be built on top of that data. Integrating the analytical layer into an interface would offer your team an MPM framework that centralizes performance tracking. Automated alerting and model flagging can also be implemented to allow for integration into tools such as PagerDuty or directly into your messaging platform (e.g., Slack).
One of the reasons that many companies are now considering MPM
solutions is that it helps them get to an important end goal: not just
implementing AI systems that are high quality and drive the bottom
line of the business, but doing so responsibly. According to Accenture,
corporate social responsibility and ethics are increasingly
important to consumers—and if that isn’t enough, companies
worldwide face a growing landscape of regulations around how AI
can be implemented and used. But what is “Responsible AI,” and
how can MPM help? Let’s take a look.

What Is Responsible AI?
Responsible AI is the practice of building AI that is transparent,
accountable, ethical, and reliable. When AI is developed responsibly,
stakeholders have insight into how decisions are made by the AI sys-

tem, and the system is governable and auditable through human
oversight. As a result, outcomes are fair to end users, stakeholders
have visibility into the AI post-deployment, and the AI system con-
tinuously performs as expected in production.

Transparency ensures that stakeholders have visibility into what the
AI system is doing. They understand the “why” behind the decisions
and predictions made, and build systems with privacy and security
built in. Accountability provides much-needed checks and balances
for AI systems with guardrails, guidelines, and governance frames-
works in place. Most importantly, there is human oversight and the
ability to override decisions where needed. Ethics broadly refers to
ensuring that AI is doing well by all impacted, with a focus on fairness and inclusion. Reliability is one of the most important principles because it is what ensures that responsible models are maintained over time, continuously.

**The Challenges of Responsible AI**

Why aren’t our existing practices for software development sufficient for deploying AI systems responsibly in production?

The reason is that AI systems are not like traditional software. With normal software, you develop to a specification, run tests, and rest assured that your program will not suddenly change its behavior. Furthermore, you can document, inspect, and debug how the software is working and what leads it to return a certain result.

AI systems are different. They are based on data, which is constantly changing. The production data that the model sees might look very different from its training data, causing its predictions to miss the mark. And AI systems are much more complex than traditional software, with layers of deep neural networks that operate in a dimensionality that is much higher than the human brain can comprehend.

When we talk about Responsible AI, we typically hear teams mentioning several different kinds of challenges, centered around explainability, reliability, fairness, and cultural change. Let’s take a look at some of the issues involved.

**Explainability**

Explainability—that is, being able to understand why the model generated a certain result—is an essential component of Responsible AI. To be accountable to your users, you need to be able to explain why a certain prediction was made, in a way that is humanly understandable. Furthermore, to assess what went wrong when decisions were unfair or harmful, and make sure this doesn’t happen again, you need to be able to inspect the model and see why it behaved the way it did.

Explainability is a major challenge with AI systems, because of the “black box” problem. Whether it’s the data science practitioner or the compliance team reviewing the model, no one really knows why it made a certain decision. That’s because the model is making
decisions based on hundreds or even thousands of factors, which are transformed and used to generate the outcome in nuanced ways, not unlike how neurons work in the brain.

These systems are so high-dimensional that humans can’t interpret them by looking at patterns in the output or at the code itself. They need special explainability algorithms and tools, which is a burgeoning field of AI research.

**Reliability**

When you have developed a model and launched it into production, it’s important to know how the model is performing. Unlike traditional software, the behavior of an AI system can change at any time when the data in production starts to look quite different from the data the system saw in training. As an example of this kind of data drift, during the COVID-19 pandemic, there was a massive shift in behavior worldwide. People’s purchasing patterns completely changed, causing once-commonplace items to become hard to find. Demand for plane tickets virtually disappeared overnight. Individuals were taking out loans when they’d never had to before.

We’ve heard countless stories of teams who had to completely retrain their models to account for the fluctuations in data. And those are just the teams that caught the problem—often, reliability issues with AI go undetected. The reason for this is, again, that these systems are complex and entirely dependent on data, which means they need new monitoring techniques. AI is also prone to unique security issues, like attackers trying to send misleading data to the model to manipulate its results.

**Fairness**

Fairness is the absence of bias for or against an individual or group based on their characteristics. At a fundamental level, the world is, sadly, an unfair place. Bias is inherently present in the world around us and encoded into our society. We can’t directly solve the problem of bias in the world at large, but Responsible AI is about making sure that we take all the measures possible to weed out bias from our data, our models, and our human review processes.

The challenge is that AI systems are particularly prone to bias. Traditional software algorithms can be biased too, but machine learning models are more likely to hide the underlying biases in the data, and
they might introduce specific, localized discrimination. There are just more ways for bias to seep in, particularly in the data chosen for training. For example, the data used to train Amazon’s facial recognition system was mostly based on white faces, leading to issues with detecting darker-skinned faces.

Figure 6-1 shows the many kinds of bias that can enter an AI system at each stage of its development.

![Bias Diagram](image)

*Figure 6-1. The kinds of bias that can be found in an AI system*

**Cultural Change**

Finally, an additional challenge is the cultural buy-in at all levels of the company that it takes in order to slow down, or even stop, to implement Responsible AI practices. Many teams are already feeling like they’re behind when it comes to taking advantage of AI. They’re looking for a fast solution that they can put in production, preferably yesterday. But moving too fast, without thinking about the potential for harm, is extremely risky—not only to the end users, but to the company’s reputation.

This is one area where tech companies would do well to take a page out of a financial institution’s playbook. The financial industry is heavily regulated, and for many years the use of models (which were traditionally simpler statistical models) has been accompanied by strict compliance rules. As a result, banks have had model risk
management teams for over a decade, overseeing and validating models. This role is naturally expanding to AI as well.

**How Model Performance Management Solves These Challenges**

MPM goes hand in hand with implementing Responsible AI. In fact, it would be very difficult to overcome the challenges we’ve just described without MPM. It solves these roadblocks with a few key innovations.

**Explainable AI**

MPM solves the AI “black box” problem by introducing explainability into every stage of the operational ML life cycle. With any good MPM solution, explainability is part of the package. During model development and validation, practitioners can use MPM to inspect how features contribute to the model, helping find places for improvement or even eliminate features that could be contributing to bias. Explainability tools with MPM can help teams prepare models for regulatory audits and provide transparency to end users as well.

We looked at Explainable AI in Chapter 2, but how does explainability work? Early explainability approaches were either simpler interpretable models that approximated the models being explained or applications of a game theory concept called Shapley values. Today we have many more explainability solutions, including gradient-based techniques like Integrated Gradients that work better for models using unstructured data. The original idea behind Shapley values was to fairly distribute gains to a coalition of players in a cooperative game based on their contribution. When applied to the machine learning context, the ML model becomes the game, the inputs are the players, and the model prediction is the gain. The Shapley values approach probes the model in a combinatorial way to see what its output would have been if the inputs had been changed. It does this in the context of individual and groups of features to account for feature correlations. This creates a unique set of contributions for each given input for a given model prediction output to help determine how much it contributed to the output.
Monitoring in Production

For an AI system, production can be full of issues such as data drift, broken data pipelines, latency problems, or computational bottlenecks. Just as we prepare for planes to crash, it’s important to prepare for models to fail.

That’s where MPM comes in. When a model is in production, it needs continuous monitoring to watch out for operational challenges like data drift, data integrity, model decay, outliers, and bias. By combining traditional performance monitoring techniques with model explainability, MPM gives teams access to actionable monitoring alerts that show them the “why” behind these operational challenges, so they can quickly solve any issues that arise.

A Single Source of Truth

To enable cultural change and address the foundational issues that contribute to “irresponsible AI,” MPM provides visibility into the model for the entire team, throughout every stage of the model’s development. This helps everyone, from engineers to business stakeholders, get on the same page. When you give teams the tools to examine and explain AI models, everyone is empowered to be accountable for the model’s decisions.

Model Governance and How MPM Fits into the Life Cycle

Outside of monitoring a single model with MPM, teams need to manage the upstream and downstream dependencies between their models. However, they often lack good tools to manage their models. Model ownership is not always clearly defined, and owners may not know who all their users are. All too often, a change in one model can break another without anyone noticing. Model governance answers questions like: Who is responsible for the model? Who is held accountable for errors in the model? Is the model operating according to industry standard regulations?

With MPM, similar to DevOps, models have clear owners and are managed in one central location, providing visibility to the entire team. Let’s take a look at how the AI life cycle works with MPM from start to finish. First, teams document the desired behavior of the AI systems and the larger product that they fit into. For example,
what are the KPIs that will show whether the model is behaving as desired? These KPIs are then built into the MPM system.

Next, the model is trained and validated in a test environment. Validation helps teams quality-check the model before moving it to production; the model’s creators can access individual explanations for each prediction to better understand what influenced that prediction. It’s important to proactively look for what can go wrong. If a certain attribute is contributing a lot to the model’s predictions (say, if someone’s income has a 20% effect on whether they are approved for a loan), does this have the potential for negative outcomes and harm? Explainable AI with MPM helps teams develop these insights.

Finally, an internal auditing and reporting process is put in place, which can be accessed by all stakeholders. MPM ensures that everyone has access to a shared source of truth, complete with monitoring and alerting of outliers, data drift, or any other type of production failure.

**The Future of Responsible AI and MPM**

The influence of AI will only continue to grow, and Responsible AI is more important than ever. Here are some of the ways we expect Responsible AI and MPM to change and develop over the next few years.

**New Regulations**

In 2019, Democratic lawmakers proposed the Algorithmic Accountability Act. The Pentagon adopted Ethical AI principles in 2020. There’s a trend here, and with China’s race to AI dominance and constant AI issues in the news, the current administration will most likely be compelled to approve the first federal regulations in AI accountability. While some regulations have come out of the EU and other countries, a well-drafted regulation from the US will be key to AI’s adoption and (responsible) growth over the next decade.

**A New Role: Chief Ethics Officer**

In 2020 there were significant breakthroughs in AI, from GPT-3 to a solution for the decades-long protein folding problem. This growth is both exciting and alarming, causing many to think about putting more guardrails in place for AI systems. Unless companies take a
more human-centric approach to developing AI, individuals working on these kinds of innovations will increasingly blow the whistle on moral issues with AI. The companies that succeed will be those that hire chief ethics officers to proactively address this problem by implementing Responsible AI with MPM.

More Tools to Test Bias

National protests about racial inequity put a powerful spotlight on bias in society in 2020. Against this backdrop, the multitude of AI bias issues at prominent companies made ML teams increasingly aware of the need for MPM and Responsible AI. No company introduces bias in its products deliberately—it's a result of inadequate tools and processes. Consumers are overwhelmingly demanding change. Fortunately, 2020 saw the release of several open source ML fairness tools, such as Fairlearn from Microsoft. With stronger ecosystem support, ML teams will embrace bias testing, even for nonregulatory use cases, as part of their MPM systems. The adoption will start small. We expect the initial focus to most likely be on assessment rather than mitigation.

MPM with Monitoring Will Become Mission Critical

The pandemic caused a dramatic shift in consumer behavior that impacted models and caught teams off guard. A lack of real-time operational visibility into production models resulted in delayed team responses, lost revenues, and reputational risks. As AI accelerates from labs into the real world, business leaders are seeing the need for visibility into deployed AI systems to ensure their metrics are continuously monitored and no inadvertent liabilities are introduced. Just like DevOps was established to create more reliable software, ML teams will introduce monitoring with MPM.

ML Model Validation Spreads Beyond Banking

The Federal Reserve and the Office of the Comptroller of the Currency mandated validation of banking models in the aftermath of the financial crisis of 2008. With AI models replacing quantitative ones, banks are applying the same rigor and process to ensure AI models are sound. After years in research and development, Explainable AI products are finally mature enough for broad adoption in financial services, giving rise to new roles like AI validator on
an AI governance team. With a model validation step, banks have been able to successfully limit inadvertent AI issues and increase their AI-driven top line. Other verticals, like insurance, retail, healthcare, and recruiting, will adopt some aspects of this model validation process to not only ensure that their models are robust but also bring ML transparency to their partner teams in 2021.

2021 is shaping up to be a seminal year in Responsible AI, where companies finally begin to adopt key practices due to undeniable political and social pressure. Companies that invest in MPM are likely to see that investment pay off in added consumer trust, fewer regulatory hurdles, and more opportunities for growth.
About the Authors

Amit Paka is the cofounder and CPO of Fiddler.ai, a mission-driven company with the goal to build trust into AI. Prior to founding Fiddler.ai, Amit led the shopping apps product team at Samsung and founded Parable, the Creative Photo Network, now part of the Samsung family. He also led PayPal's consumer in-store mobile payments launching innovations like hardware beacon payments and has developed successful startup products, particularly in online advertising—paid search, contextual, ad exchange, and display advertising. Amit has passions for actualizing new concepts, building great teams, and pushing the envelope. He aims to leverage these skills to help define how AI can be fair, ethical, and responsible.

Krishna Gade is the cofounder and CEO of Fiddler.ai, a mission-driven company with the goal to build trust into AI. Prior to founding Fiddler.ai, Gade led the team that built Facebook's explainability feature ‘Why am I seeing this?’. He's an entrepreneur with a technical background with experience creating scalable platforms and expertise in converting data into intelligence. Having held senior engineering leadership roles at Facebook, Pinterest, Twitter, and Microsoft, he’s seen the effects that bias has on AI and machine learning decision making processes, and with Fiddler.ai his goal is to enable enterprises across the globe to solve this problem.

Danny Farah is a senior consultant based out of Toronto, Canada, and specializes in architecting DataOps and MLOps systems to empower organizations to execute on their mission and vision. He has expertise in data science, data engineering, and machine learning, both through academic knowledge and professional experience. He holds a master of engineering degree from the University of Toronto, where he focused on advanced analytics and data science principles within the Industrial Engineering department.

Throughout his professional experience, Danny has worked with various clients across multiple industry sectors and helped propel them past their goals. Danny has worked with all three major cloud providers—Google, Amazon, and Microsoft—as well as other third-party technology providers within the machine learning and data engineering space. Working across technology providers has allowed him to gain a deep understanding of the MLOps ecosystem as it stands today and how it may be applied within the context of a
given provider or technology stack. This hands-on experience has allowed Danny to relate high-level business requirements to technical needs and understand how to architect and build scalable systems that meet those ever-increasing business requirements.