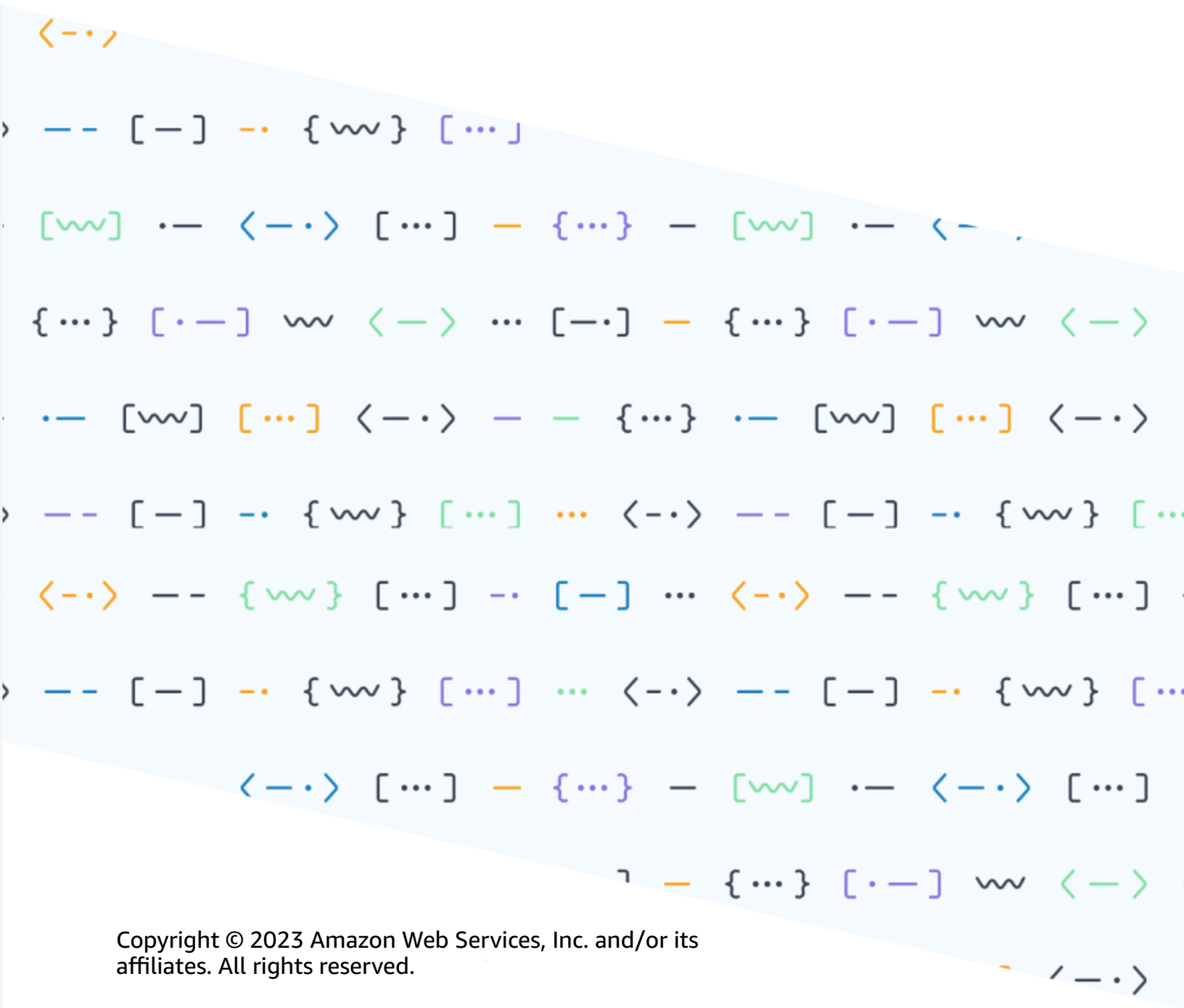
AWS Whitepaper

Machine Learning Best Practices in Healthcare and Life Sciences



Machine Learning Best Practices in Healthcare and Life Sciences: AWS Whitepaper

Table of Contents

Machine Learning Best Practices in Healthcare and Life Sciences

Publication date: **November 22, 2021** (*Document history*)

This whitepaper describes how AWS approaches machine learning (ML) in a regulated environment and provides guidance on good ML practices using AWS products.

This whitepaper takes into consideration the principles described in the Proposed Regulatory Framework for Modifications to Artificial Intelligence/Machine Learning (AI/ML)-Based Software as a Medical Device (SaMD) discussion paper.

This content has been developed based on experience with and feedback from AWS pharmaceutical and medical device customers, as well as software partners, who are currently using AWS products to develop ML models.

Introduction

The pharmaceutical industry, which is sometimes slow to adopt the latest technologies, is witnessing a massive change. The industry is looking to technologies such as artificial intelligence/machine learning (AI/ML), Internet of Things (IoT), blockchain, and other Industry 4.0 technologies. With this adoption of new technology comes regulatory challenges.

Machine Learning in particular has garnered some focus recently with the publishing of a discussion paper by the FDA. It explores the use of AI/ML in the context of medical devices but many of the same topics arise when bringing up ML adoption with executive leadership in any company: How do you trust ML models to not make important business decisions based on erroneous or unstable values? How do you know you have good hygiene in managing your ML environments? Are you prepared for (or capable of) a retrospective analysis if anything goes wrong?

One particularly strong example is seen in pharmaceutical companies, who must abide by mandatory reporting standards for any patient-relevant “Adverse Events” that are viewed by any employee or ingested by the company. This means that creating a new Twitter account for a particular market, or releasing a digital therapeutic app for patient health, can result in a surge of new natural language data that needs to be reviewed, adjudicated, and in some cases reported to the FDA.

This flood of data can completely overwhelm manual review teams and risk delays in reporting to the FDA within the mandated time limits, resulting in the potential for formal warnings or even legal action. ML has the potential to address the immediate needs of scale in these scenarios, and triage obvious problem cases from innocuous cases.

However, before deploying these models, you may need to evaluate a few requirements relevant to your stakeholders or regulators, such as model reproducibility, model explainability, decision support tooling, and how this all ties into templatized ML workflows.

Benefits of machine learning

Regulatory agencies such as the FDA acknowledge that ML-based technologies hold the potential to transform healthcare through their ability to derive new and important insights from vast amounts of data. One of the technology's greatest strengths is its ability to continually learn from real-world data, and its capability to improve its performance. However, this poses a regulatory challenge, as the model results change over time as the model learns, and this change happens after regulatory approval. This challenge was raised in an FDA discussion paper, and this whitepaper will endeavor to demonstrate how customers may be able to address that challenge using AWS services.

The ability for AI/ML software to learn from real-world feedback (training) and improve its performance (adaptation) makes these technologies uniquely situated among software as a medical device (SaMD). With appropriately tailored regulatory oversight, AI/ML-based SaMD has the potential to deliver safe and effective software functionality that improves the quality of care that patients receive.

To help customers realize the benefits of ML, this whitepaper covers the points raised by the FDA while also drawing from AWS resources.

The Life Sciences industry (encompassing, but not exclusively, bio-pharma, genomics, medical diagnostics, and medical devices) is governed by a set of regulatory guidelines called Good Laboratory Practice, Good Clinical Practice, Good Manufacturing Practice, and Good Machine Learning Practices (commonly referred to as "GxP").

The GxP Systems on AWS whitepaper provides information on how AWS approaches GxP-related compliance and security, and provides customers guidance on using AWS products in the context of GxP. The paper was developed based on experience with and feedback from AWS pharmaceutical and medical device customers, as well as software partners, who are currently using AWS products in their validated GxP systems.

The AWS Well-Architected Framework helps you understand the pros and cons of decisions you make while building systems on AWS. Using the Framework enables you to learn architectural best practices for designing and operating reliable, secure, efficient, and cost-effective systems in the cloud. It provides a way for you to consistently measure your architectures against best practices and identify areas for improvement.

Having well-architected systems greatly increases the likelihood of business success.

The Machine Learning Lens whitepaper focuses on how to design, deploy, and architect your ML workloads in the AWS Cloud. This lens adds to the best practices included in the Well-Architected Framework, but also covers ML scenarios and identifies key elements to ensure that your workloads are architected according to best practices. These include how to design, deploy, and architect your ML workloads in the AWS Cloud.

Life sciences at AWS

AWS started its dedicated Genomics and Life Sciences Practice in 2014 in response to the growing demand for an experienced and reliable life sciences cloud industry leader. At the time of this writing, the AWS Life Sciences Practice team consists of members that have been in the industry for over 17 years on average, and had previous titles such as Chief Medical Officer, Chief Digital Officer, Physician, Radiologist, and Researcher, among many others.

The AWS Genomics and Life Sciences practice serves a large ecosystem of life sciences customers, including some of the largest enterprise pharmaceutical, biotechnology, medical device, genomics, start-ups, university, and government institutions, as well as healthcare payers and providers. A full list of customer case studies can be found at Healthcare & Life Sciences Case Studies.

In addition to the resources available within the Genomics and Life Science practice at AWS, you can also work with AWS Life Sciences Competency Partners to drive innovation and improve efficiency across the life sciences value chain, including cost-effective storage and compute capabilities, advanced analytics, and patient personalization mechanisms. AWS Life Sciences Competency Partners have demonstrated technical expertise and customer success in building Life Science solutions on AWS. A full list of AWS Life Sciences Competency Partners can be found at AWS Life Sciences Competency Partners.

AI/ML in the life sciences industry fall under five main categories:

- **Research and discovery** — Use cases include molecular structure prediction, antibody binding affinity prediction, outcome prediction for patients with certain biomarkers, patient data enrichment and search, the classification of tissue samples, and so on.
- **Clinical development** — Use cases include forecasting and optimization of trial timelines, optimization of inclusion/exclusion criteria and site selection, protocol document enrichment, and so on.
- **Manufacturing and supply chain** — Use cases include predictive maintenance of equipment, computer vision for effective line clearance, optimization of manufacturing process staging, vial inspection, and so on.

- **Commercial** — Use cases include predicting healthcare providers with relevant patient bases, identifying next best action for sales and marketing, annotation and management of existing promotional materials, and so on.
- **Post-market surveillance and patient support** — Use cases include forecasting patient cost, automation of adverse event detection from social media or call centers, patient outcome prediction, and so on.

Current regulatory situation

Traditional Computer Systems Validation (CSV) is a point-in-time exercise where the resultant computer system was tested against the requirements to verify that the system satisfies its intended use. Whenever there was a change, the system went through revalidation.

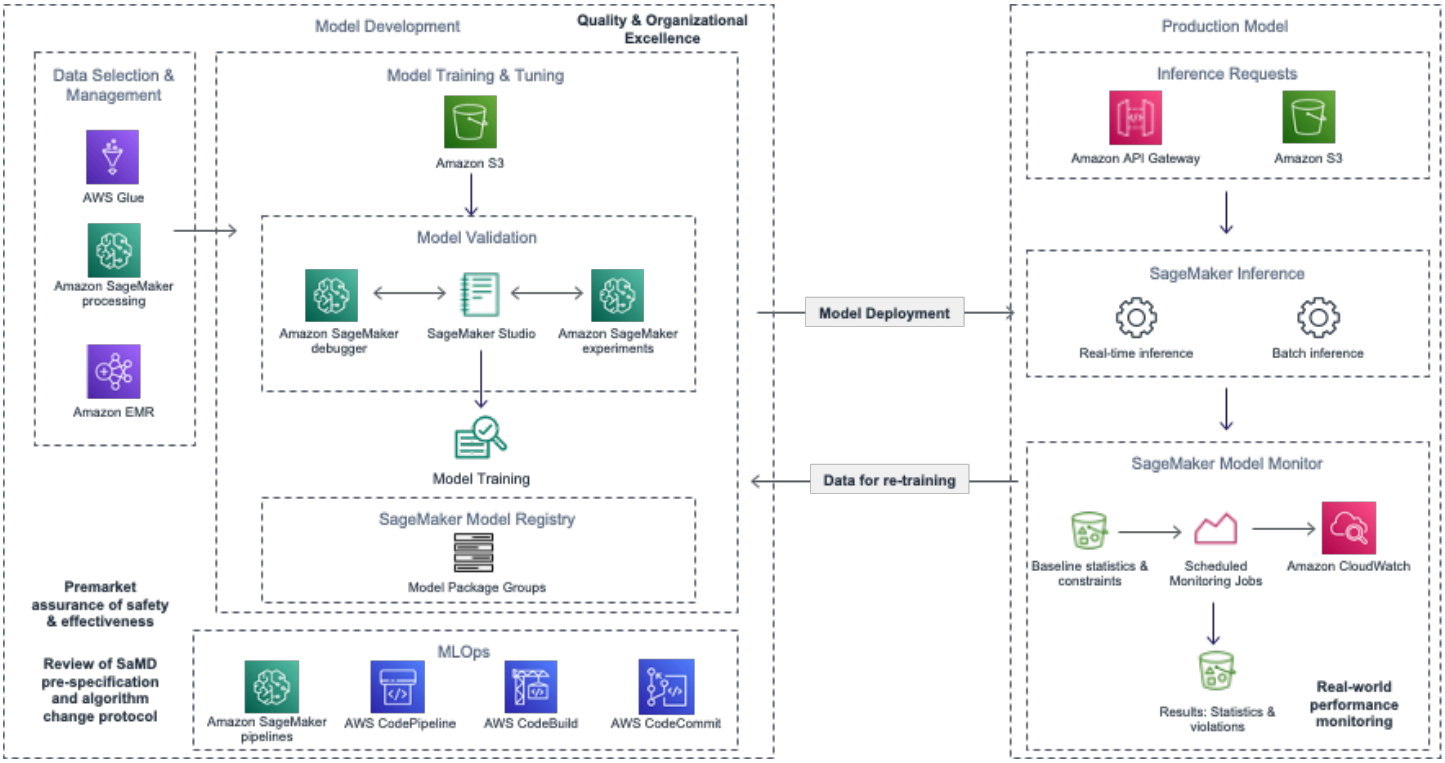
However, the FDA's view is that AI/ML-based SaMD exist on a spectrum, from locked to nearly continuously learning and changing.

"Locked" algorithms provide the same result each time the same input is provided. For these kinds of models, the static/point-in-time validation approach is still appropriate. However, an adaptive/nearly continuous learning algorithm changes its behavior using a defined learning process. Point-in-time validation does not work for such a model

The discussion paper outlines an approach of assessing the ML practices employed during the development and operation of an ML model, rather than the result of those practices (the model itself). The FDA proposes a Total Product Lifecycle (TPLC) approach which enables the evaluation and monitoring of a software product from its premarket development to postmarket performance, along with continued demonstration of the organization's excellence.

With this approach, the FDA will assess the following:

- Culture of quality and organizational excellence — to gain a reasonable assurance of the high quality of the organization's software development, testing, and performance monitoring capabilities
- Pre-market assurance of safety and effectiveness
- Review of SaMD pre-specifications and algorithm change protocol
- Real-world performance monitoring



Reference implementation of good machine learning practices

This framework relies on the principle of a "predetermined change control plan." The predetermined change control plan includes the types of anticipated modifications, (SaMD Pre-Specifications) based on the retraining and model update strategy, and the associated methodology (Algorithm Change Protocol) being used to implement those changes in a controlled manner that manages risks to patients.

- **SaMD Pre-Specifications (SPS)** — A SaMD manufacturer's anticipated modifications to "performance" or "inputs," or changes related to the "intended use" of AI/ML-based SaMD. These are the types of changes the manufacturer plans to achieve when the SaMD is in use. The SPS draws a "region of potential changes" around the initial specifications and labeling of the original device. This is what the manufacturer intends the algorithm to become as it learns.
- **Algorithm Change Protocol (ACP)** — Specific methods that a manufacturer has in place to achieve and appropriately control the risks of the anticipated types of modifications delineated in the SPS. The ACP is a step-by-step delineation of the data and procedures to be followed so that the modification achieves its goals and the device remains safe and effective after the modification. The preceding figure provides a general overview of the components of an ACP. This is how the algorithm will learn and change while remaining safe and effective.

Challenges to support AI/ML enabled GxP workloads

Developing AI/ML-enabled GxP workloads raise the following challenges:

- Building a secure infrastructure that complies with a stringent regulatory process working on the public cloud and aligning to the FDA framework for AI/ML
- Supporting an AI/ML-enabled solution for GxP workloads covering:
  - Reproducibility
  - Traceability
  - Data integrity
- Monitoring the Machine Learning model with respect to various changes to parameters and data
- Handling model uncertainty and confidence calibration
- Making AI/ML models interpretable

Provision a secure and GXP-compliant machine learning environment

For healthcare and life sciences customers, the security and privacy of an ML environment is paramount. It is therefore strongly recommended that you protect your environment against unauthorized access, privilege escalation, and data exfiltration. Common considerations when you set up a secure ML environment include:

- Platform qualification
- Compute and network isolation
- Authentication and authorization

These considerations are detailed in the following sections.

Platform qualification

The validated state of ML models may be brought into question if the underlying IT infrastructure is not maintained in a demonstrable state of control and regulatory compliance. Similarly, data integrity can also be affected by problems related to IT infrastructure. Therefore, to support a culture of quality and operational excellence, it is important to qualify your underlying platform and then maintain it under a state of control.

Details about platform qualification and the approach taken by many AWS customers are described in GxP whitepaper. However, you should adhere to your own internal processes.

Customers will often provide a qualified MLOps platform to their teams, along with a selection of pre-qualified building blocks to support their specific needs. These are often made available to development teams through services like Service Catalog.

Compute and network isolation

A well-governed and secure ML workflow begins with establishing a private and isolated compute and network environment. The virtual private cloud (VPC) that hosts Amazon SageMaker and its associated components, such as Jupyter notebooks, training instances, and hosting instances, should be deployed in a private network with no internet connectivity.

These SageMaker resources can be associated with your customer VPC environment. This enables you to apply network level controls such as security groups to govern access to SageMaker resources, and control ingress and egress of data into and out of the environment.

Connectivity between SageMaker and other AWS services, such as Amazon Simple Storage Service (Amazon S3), should be established using VPC endpoints or even AWS PrivateLink. Additionally, when creating a VPC endpoint, you can attach an endpoint policy to further control access to specific resources for which you are connecting.

Similarly, for training jobs and hosted models, SageMaker EC2 instances will communicate with Amazon S3 through your VPC when retrieving training data from Amazon S3. SageMaker does this by creating an elastic network interface in your specified VPC and attaching it to the Amazon EC2 infrastructure in the service account.

Using this pattern, the service gives you control over the network-level access of the services you run on SageMaker. When the SageMaker instances retrieve your trained model for hosting, they will communicate with S3 through your VPC. In this manner, you can maintain governance and control of the network communication of your SageMaker resources.

For details on setting up a private network environment, refer to the Amazon SageMaker Workshop for Secure Networking.

Authentication and authorization

After you create an isolated and private network environment, the next step is to ensure that only authorized users can access the appropriate AWS services. AWS Identity and Access Management (IAM) can help you create preventive controls for many aspects of your ML environment, including access to SageMaker resources, access to your data in S3, and access to API endpoints.

You can access AWS using a RESTful API, and every API call is authorized by IAM. You grant explicit permissions through IAM policy documents, which specify the principal (who), the actions (API calls), and the resources (such as S3 objects) that are allowed, as well as the conditions under which the access is granted.

Access to AWS Glue and Amazon SageMaker resources is also governed by IAM. While each organization has different authentication and access requirements, you should configure permissions in line with IAM best practices and your own internal policies and controls by granting least privilege access, or granting only the permissions required to perform a particular task.

Role-based access control (RBAC) is an approach used commonly by customers in financial services for ensuring only authorized parties have access to desired system controls. Creating roles based on job function policies, and using AWS Config to monitor and periodically audit IAM policies attached to users, is a recommended best practice for viewing configuration changes over time.

AWS Config offers conformance packs, which provide a general-purpose compliance framework designed to enable you to create security, operational or cost-optimization governance checks using managed or custom AWS Config rules and AWS Config remediation actions.

Conformance packs, as sample templates, are not designed to fully ensure compliance with a specific governance or compliance standard. You are responsible for making your own assessment of whether your use of the services meets applicable legal and regulatory requirements. AWS Security Assurance Services LLC (SAS) professionals designed a conformance pack to enable customers to align to a subset of FDA Title 21 CFR Part 11 design principles. Refer to Operational Best Practices for FDA Title 21 CFR Part 11 and Operational best practices for AI and ML.

This conformance pack contains AWS Config rules based on AI and ML that has been designed for compatibility with the majority of AWS Regions, and does not require the setting of any parameters.

Data encryption

Because ML environments can contain sensitive data and intellectual property, one of the considerations for a secure ML environment is data encryption. AWS recommends that you enable data encryption, both at rest and in transit.

For at rest encryption, AWS provides tools for creating an encrypted file system using open standard algorithms. For instance, customers can encrypt data stored in Amazon S3 and the data stored in Amazon Elastic Block Store (Amazon EBS) volumes. Similarly, AWS Glue supports data encryption at rest for Authoring jobs in AWS Glue and Developing scripts using development endpoints. You can also configure ETL jobs and development endpoints to use AWS Key Management Service (AWS KMS).

Some healthcare customers require the use of AWS KMS keys. Additionally, AWS recommends enabling Amazon S3 default encryption so that all new objects are encrypted when they are stored in the Amazon S3 bucket.

AWS also recommends using Amazon S3 bucket policies to prevent unencrypted objects from being uploaded. For data in transit, all internetwork data in transit supports TLS 1.2 encryption.

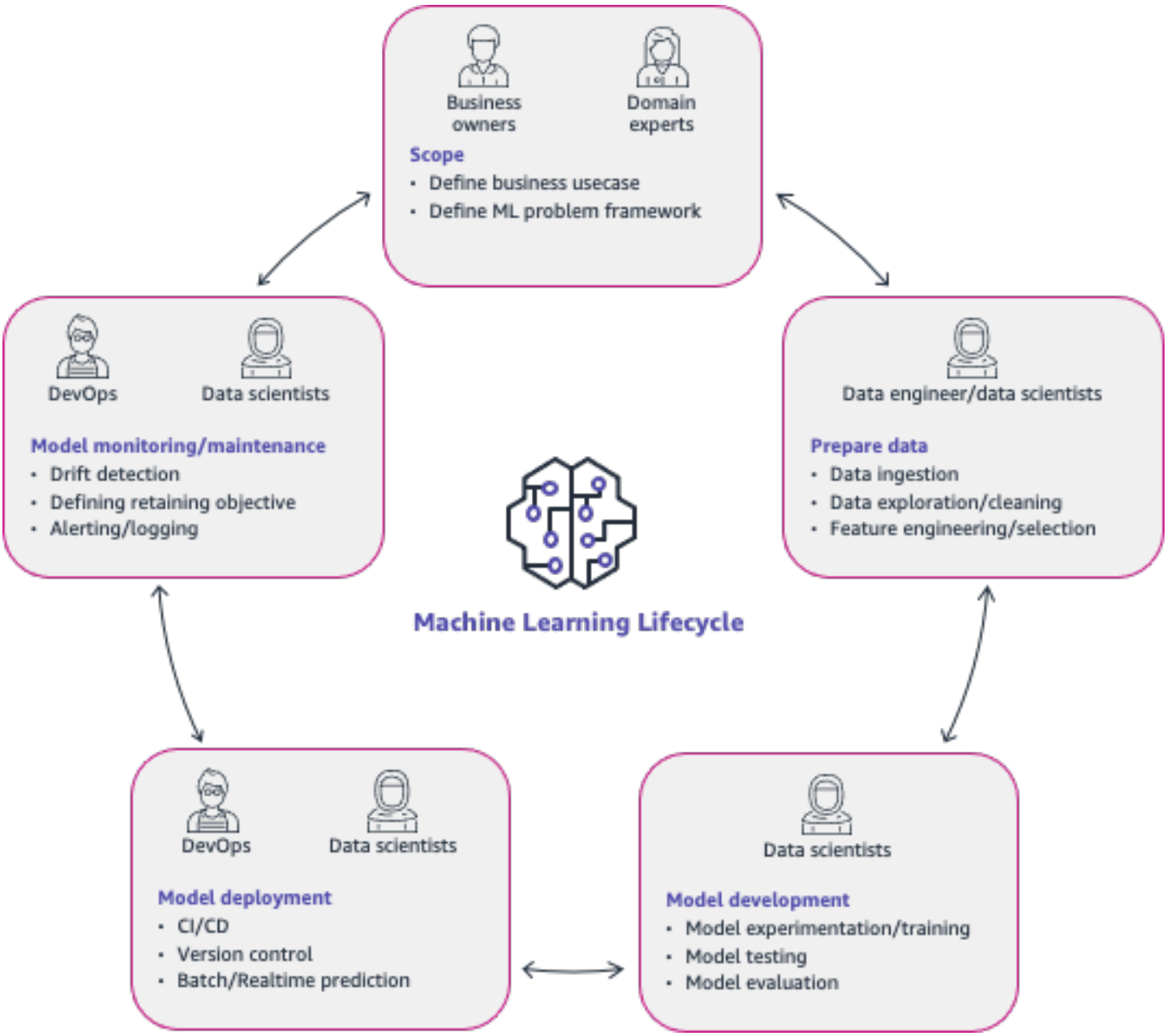
Finally, for data transmitted between instances during distributed training, inter-container traffic encryption can be enabled. However, note that this can increase the training time, particularly for distributed deep learning algorithms.

Machine learning lifecycle

Building and operating a typical ML workload is an iterative process, and consists of multiple phases. We identify these phases loosely based on the open standard process model for Cross Industry Standard Process Data Mining (CRISP-DM) as a general guideline. CRISP-DM is used as a baseline because it's a proven tool in the industry and is application neutral, which makes it an easy-to-apply methodology that is applicable to a wide variety of ML pipelines and workloads. The end-to-end ML process includes the following phases:

- Business goal identification
- ML problem framing
- Data collection
- Data integration and preparation
- Feature engineering
- Model training
- Model validation
- Business evaluation
- Production deployment (model deployment and model inference)

This section presents a high-level overview of the various phases of an end-to-end ML lifecycle, which helps frame our discussion around security, compliance, and operationalization of Amazon Web Services ML best practices in healthcare and life science.



The machine learning lifecycle process

The preceding figure describes the ML lifecycle process, along with the subject matter experts and business stakeholders involved through different stages of the process. It is also important to note that ML lifecycle is an interactive process.

Phase 1

Phase 1 is to define business problem and goals. Domain experts and business owners are most involved in this part, determining success metrics. KPIs and determining compliance and regulatory

requirements also fall under this phase. Data scientists typically work with the SMEs to frame the business problem in a way that allows them to develop a viable ML solution.

Phase 2

Phase 2 involves gathering and preparing all relevant data from various data sources. This role is often performed by data engineers with expertise in big data tools for data extraction, transformation and loading (ETL). It is important to ensure that the data is versioned and the lineage of the data tracked for auditing and compliance purpose. Once the raw datasets are available, data scientists perform data exploration, determine input features and target variables, outlier analysis, and necessary data transformations that may be needed. It is also important to ensure any transformations applied to training data can also be applied in production at inference time.

Phase 3

Next is the model development and model evaluation phase. Data scientists determine the framework they want to use, define out-of-sample, out-of-time datasets, and experiment with various ML algorithms, hyperparameters, in some cases, add additional training data.

Phase 4

Next, you take the trained models and run them on out of time and out of sample datasets, and pick the model or models that return the best results close to the metrics determined in Phase 1. Model artifacts and any corresponding code must be properly versioned and stored in a centralized code repository or in an artifact management system.

Note that this stage of the process is experimental, and data scientists may go back to the data collection or feature engineering stage if the model performance is consistently poor. More details on data and ML artifact lineage are available in the Traceability section of this document.

Data scientists are also required to provide reasons or explain feature/model influence on predictions. Model interpretability is discussed in later sections.

Phase 5

The next phase is to deploy the models into production. This is often the most impactful and difficult step because of the gap between technologies and skillsets used to build and deploy

models in production. A large part of making this successful requires intense collaboration among infrastructure professionals such as DevOps engineers, data scientists, data engineers, domain experts, end users, and business owners during the decision making process. There should be standardized metrics, and all decision makers should be able to interpret them directly.

In most organizations, lifecycles of ML models end with the deployment phase. There is a need for some form of shadow validation where models are deployed but not integrated in the production workflow to capture differences between training and live data. This ensures that the model continues to perform as expected when receiving data from production systems. Once this validation proves successful, the model's predictions can be used in production workflows.

However, for ML models to be effective in the long run, continuously monitoring the model in real-time (if possible) to determine how well it is performing is necessary, as the accuracy of models can degrade over time. If the performance of a model degrades below a certain threshold, you may need to retrain and redeploy your model. Further details concerning model monitoring are discussed in the Model monitoring section of this document.

Best practices for ML lifecycle stages

Topics

Data collection

The first step in the development of ML workloads is identification of data that is needed for training and performance evaluation of an ML model. In the cloud environment, a data lake usually serves as a centralized repository that enables you to store all structured and unstructured data regardless of scale.

AWS provides a number of ways to ingest data, both in bulk and in real-time, from a wide variety of sources. You can use services such as AWS Direct Connect and AWS Storage Gateway to move data from on-premises environments, and tools like AWS Snowball and AWS Snowmobile for moving data at scale. You can also use Amazon Kinesis to collect and ingest streaming data. You also have the option to use services such as AWS Lake Formation and AWS HealthLake to quickly set up data lakes. The following best practices are recommended for data collection and integration:

- **Detail and document** various sources and steps needed to extract the data. This can be achieved using AWS Glue Data Catalog, which automatically discovers and profiles your data, and generates ETL code to transform your source data to target schemas. AWS also recently announced a new feature named AWS Glue DataBrew, which provides a visual data preparation

interface that makes it easy for data analysts and data scientists to clean and normalize data to prepare it for analytics and ML.

- **Define data governance** — Who owns the data, who has access, the appropriate usage of the data, and the ability to access and delete specific pieces of data on demand. Data governance and access management can be handled using AWS Lake Formation and AWS Glue Data Catalog.

Data integration and preparation

An ML model is only as good as the data being used to train it. Bad data is often referred to as "Garbage in, Garbage out". Once the data has been collected, the next step is to integrate, prepare and annotate data. AWS provides a number of services that data engineers and data scientists can use to prepare their data for ML model training.

In addition to the services such as AWS Glue and Amazon EMR, which provide traditional ETL capabilities, AWS also provides tools as part of Amazon SageMaker, designed specifically for data scientists. These include:

- **Amazon SageMaker Ground Truth**, which can be used for data labeling
- **SageMaker Data Wrangler**, which simplifies the process of data preparation and feature engineering
- **SageMaker Feature Store**, which enables you to store, update, retrieve, and share ML features

Additionally, SageMaker Processing allows you to run your pre-processing, post-processing, and model evaluation workloads on a fully managed environment. We recommend implementation of the following best practices for data integration and preparation:

- **Track data lineage** so that the location and data source is tracked and known during further processing. Using AWS Glue, you can visually map the lineage of their data to understand the various data sources and transformation steps that the data has been through. You can also use metadata provided by AWS Glue Data Catalog to establish data lineage. The SageMaker Data Wrangler Data Flow UI provides a visual map of the end-to-end data lineage.
- **Versioning data sources and processing workflows** — Versioning data sources processing workflows enables you to maintain an audit trail of the changes being made to your data integration processes over time, and recreate previous versions of your data pipelines. AWS Glue provides versioning capabilities as part of AWS Glue Data Catalog, and AWS Glue Schema

Registry (for streaming data sources). AWS Glue and Amazon EMR jobs can be versioned using a version control system such as AWS CodeCommit.

- **Automate data integration deployment** pipelines — Minimize human touch points in deployment pipelines to ensure that the data integration workloads are consistently and repeatedly deployed, using a pipeline that defines how code is promoted from development to production. AWS Developer Tools allow you to build CI/CD pipelines to promote your code to a higher environment.

Feature engineering

Feature engineering involves the selection and transformation of data attributes or variables during the development of a predictive model. Amazon SageMaker Data Wrangler can be used for selection, extraction, and transformation of features. You can export your data flow, designed in Data Wrangler, as a Data Wrangler Job, or export to SageMaker Pipelines.

ETL services like Amazon EMR and AWS Glue can be used for feature extraction and transformation. Finally, you can use Amazon SageMaker Feature Store to store, update, retrieve and share ML features. The following best practices are recommended for feature engineering:

- **Ensure feature standardization and consistency** — It is common to see a different definition of similar features across a business. The use of Amazon SageMaker Feature Store allows for standardization of features, and helps to ensure consistency between model training and inference.
- If you are using SageMaker for feature engineering, you can use SageMaker Lineage Tracking to store and track information about the feature engineering steps (along with other ML workflow steps performed in SageMaker).

Model training

The model training step involves the selection of appropriate ML algorithms, and using the input features to train an ML model. Along with the training data (provided as input features prepared during the feature engineering stage), you generally provide model parameters to optimize the training process.

To measure how well a model is performing during training, AWS uses several metrics such as training error and prediction accuracy. Metrics reported by the algorithm depend on the business problem and the ML technique being used.

Certain model parameters, called *hyperparameters*, can be tuned to control the behavior of the model and the resulting model architecture. Model training typically involves an iterative process of training a model, evaluating its performance against relevant metrics, and tuning the hyperparameters in search for the most optimal model architecture. This process is generally referred to as *hyperparameter optimization*. AWS recommends the application of the following best practices during the model training step:

- **Follow a model testing plan and track your model experiments** — Amazon SageMaker Experiments enables you to organize, track, compare, and evaluate ML experiments and model versions.
- **Take advantage of managed services for model turning** — SageMaker Automatic Model Tuning and SageMaker Autopilot help ML practitioners explore a large number of combinations to automatically and quickly zoom in on high-performance models.
- **Monitor your training metrics to ensure your model training is achieving the desired results** — SageMaker Debugger can be used for this purpose, which is designed to profile and debug your training jobs to improve the performance of ML models.
- **Ensure traceability of model training as part of the ML lifecycle** — SageMaker Lineage Tracking can be used for this purpose.

Model validation

After the model has been trained, evaluate it to determine if its performance and accuracy will enable you to achieve your business goals. Data scientists typically generate multiple models using different methods, and evaluate the effectiveness of each model. The evaluation results inform the data scientists' decision to fine-tune the data or algorithms to further improve the model performance.

During fine-tuning, data scientists might decide to repeat the data preparation, feature engineering, and model training steps. AWS recommends the following best practices for model validation:

- **Keep track of the experiments performed to train models using different sets of features and algorithms** — Amazon SageMaker Experiments, as discussed in the Model training section, can help keep track of different training iterations and evaluation results.
- **Maintain different versions of the models and their associated metadata such as training and validation metrics in a model repository** — SageMaker Model Registry enables you to catalog

models for production, manage model versions, manage approval status of the models, and associate metadata, such as the training metrics of a model.

- **Transparency about how a model arrives at their predictions is critical for regulators who require insights into how a model makes a decision** — AWS recommends that you use model explainability tools, which can help explain how ML models make predictions. SageMaker Clarify provides the necessary tools for model explainability.
- **Biases in the data can introduce bias in ML algorithms**, which can significantly limit the effectiveness of the models. This is of special significance in healthcare and life sciences, because poorly performing or biased ML models can have a significant negative impact in the real-world. SageMaker Clarify can be used to perform the post-training bias analysis against the ML models.

Auditability

Another consideration for a well governed and secure ML environment is having a robust and transparent audit trail that logs all access and changes to the data and models, such as a change in the model configuration, or the hyperparameters.

AWS CloudTrail is one service that will log, nearly continuously monitor, and retain account activity related to actions across your AWS infrastructure. CloudTrail logs every AWS API call, and provides an event history of your AWS account activity, including actions taken through the AWS Management Console, AWS SDKs, command line tools, and other AWS services.

Another service, AWS Config, enables you to nearly continuously monitor and record configuration changes of your AWS resources. More broadly, in addition to the logging and audit capabilities, AWS recommends a defense in depth approach to security, applying security at every level of your application and environment. AWS CloudTrail and AWS Config can be used as Detective controls responsible for identifying potential security threats or incidents.

As the Detective controls identify potential threats, you can set up a corrective control to respond to and mitigate the potential impact of security incidents. Amazon CloudWatch is a monitoring service for AWS resources, which can trigger CloudWatch Events to automate security responses. For details on setting up Detective and corrective controls, refer to Logging and Monitoring in AWS Glue.

Traceability

Effective model governance requires a detailed understanding of the data and data transformations used in the modeling process, in addition to nearly continuous tracking of all model development iterations. It is important to keep track of which dataset was used, what transformations were applied to the data, where the dataset was stored, and what type of model was built.

Additional variables, such as hyperparameters, model file location, and model training metadata also need to be tracked. Any post-processing steps that have been applied to remove biases from predictions during batch inference also need to be recorded.

Finally, if a model is promoted to production for inference, there needs to be a record of model files/weights used in production, and model performance in production needs to be monitored.

One aspect of traceability that helps ensure you have visibility of what components or artifacts make their way into production, and how they evolve over time in the form of updates and patches, is the use of *versioning*. There are three key components that provide versioning for different types of components involved in developing an ML solution:

- Using software version controls through tools such as GitHub to keep track of changes made to processing, training, and inference script. AWS provides a native version control system in the form of AWS CodeCommit that can be used for this purpose. Alternatively, you can use your own GitHub implementations.
- Using a model versioning capability to keep track of different iterations of models being created as part of iterative training runs. SageMaker Model Registry, which natively integrated with the wider SageMaker features, can be used for this purpose.
- Using a container repository to keep track of different container versions, which are used in SageMaker for processing, training, and inference. SageMaker natively integrates with Amazon ECR, which maintains a version of every container update.

Reproducibility

Reproducibility in ML is the ability to produce identical model artifacts and results by saving enough information about every phase in the ML workflow, including the dataset, so that it can be reproduced at a later date or by different stakeholders, with the least possible randomness in the process.

For GxP compliance, customers may need to reproduce and validate every stage of the ML workflow to reduce the risk of errors, and ensure the correctness and robustness of the ML solution.

Unlike traditional software engineering, ML is experimental, highly iterative, and consists of multiple phases that make reproducibility challenging. It all starts with the data. It's important to ensure that the dataset is reproducible at each phase in the ML workflow. Variability in the dataset could arise due to randomness in subsampling methods, creating train/validation/test splits and dataset shuffling.

Variability could also arise due to changes in the data processing, feature engineering, and post-processing scripts. Inconsistencies in any of these phases can lead to an irreproducible solution. Methods that can help ensure reproducibility of the dataset as well as the data processing scripts include:

- Dataset versioning
- Using a fixed seed value across all the libraries in the code base
- Unit testing code to ensure that the outputs remain the same for a given set of inputs
- Version controlling the code base

The core components of the ML workflow are the ML models, which consist of a combination of model parameters and hyperparameters, which need to be tracked to ensure consistent and reproducible results.

In addition to these parameters, the stochastic (uncertain or random) nature of many ML algorithms adds a layer of complexity, because the same dataset along with the same code base could produce to different outputs. This is more pronounced in deep learning algorithms, which make efficient approximations for complex computations. These results can be approximately reproduced with the same dataset, the same code base, and the same algorithm.

In addition to the algorithms, the underlying hardware and software environment configurations could impact reproducibility as well. Methods that can help ensure reproducibility and limit the number of sources of nondeterministic behavior in ML modeling include:

- Consistency in initializing model parameters
- Standardizing the infrastructure (CPUs and GPUs)
- Configuration management to ensure consistency in the runtimes, libraries and frameworks

When the solutions aren't fully deterministic, the need for quantifying the uncertainty in model prediction increases. Uncertainty quantification (UQ) plays a pivotal role in the reduction of uncertainties during optimization and decision making, and promotes transparency in the GxP compliance process. A review of uncertainty quantification techniques, applications, and challenges in deep learning are presented in *A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges*.

Few methods for uncertainty quantification include:

- Ensemble learning techniques such as Deep Ensembles, which are generalizable across ML models and can be integrated into existing ML workflows.
- Temperature scaling, which is an effective post-processing technique to restore network calibration, such that the confidence of the predictions matches the true likelihood. Refer to a reference paper on calibrating neural networks.
- Bayesian neural networks with Monte Carlo dropout.

For more information about these methods, refer to Methods for estimating uncertainty in deep learning.

Amazon SageMaker ML Lineage Tracking provides the ability to create and store information about each phase in the ML workflow. In the context of GxP compliance, this can help you establish model governance by tracking model lineage artifacts for auditing and compliance verification. SageMaker ML Lineage Tracking tracks entities that are automatically created by SageMaker, or custom created by customers, to help maintain the representation of all elements in each phase of the ML workflow.

Model interpretability

Interpretability is the degree to which a human can understand the cause of a decision. The higher the interpretability of an ML model, the easier it is to comprehend the model's predictions. Interpretability facilitates:

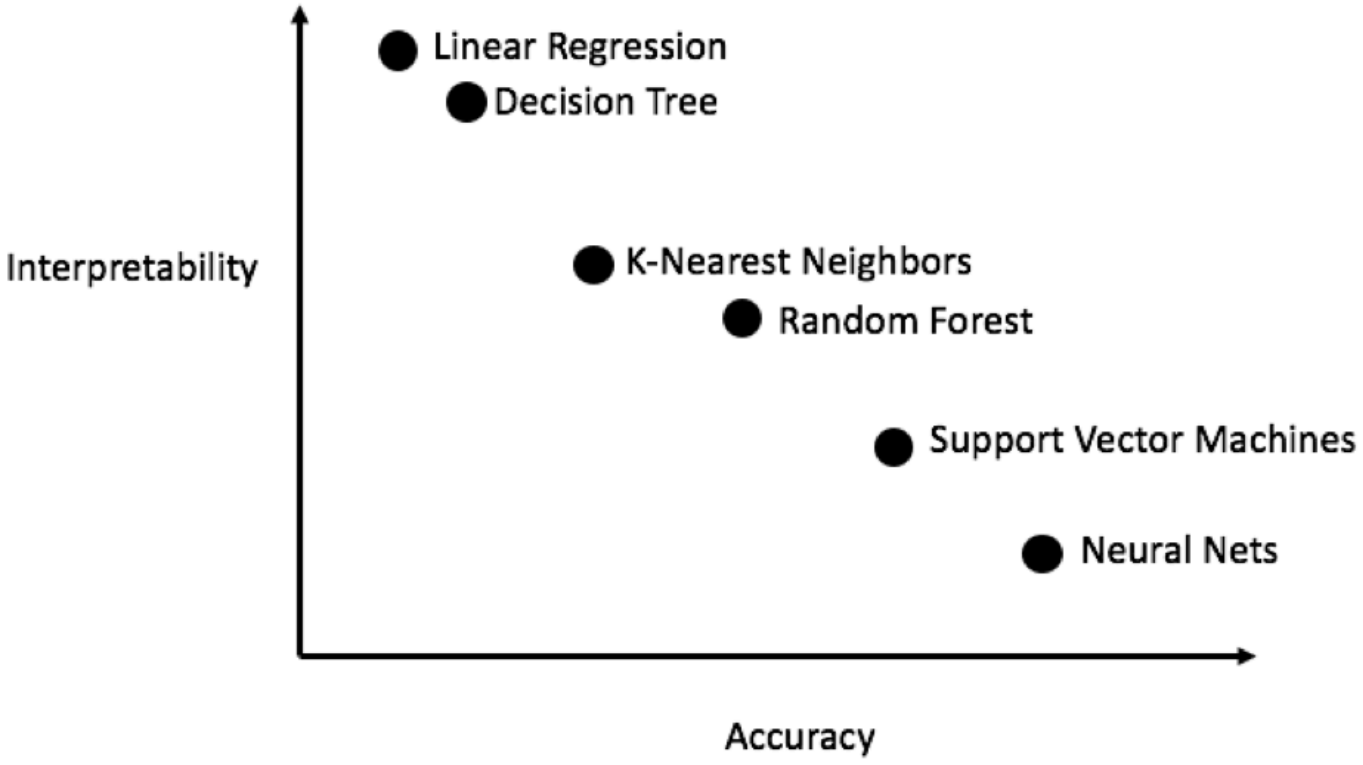
- Understanding
- Debugging and auditing ML model predictions
- Bias detection to ensure fair decision making
- Robustness checks to ensure that small changes in the input do not lead to large changes in the output

- Methods that provide recourse for those who have been adversely affected by model predictions

In the context of GxP compliance, model interpretability provides a mechanism to ensure the safety and effectiveness of ML solutions by increasing the transparency around model predictions, as well as the behavior of the underlying algorithm. Promoting transparency is a key aspect of the patient-centered approach, and is especially important for AI/ML-based SaMD, which may learn and change over time.

There is a tradeoff between *what* the model has predicted (model performance) and *why* the model has made such a prediction (model interpretability).

For some solutions, a high model performance is sufficient; in others, the ability to interpret the decisions made by the model is key. The demand for interpretability increases when there is a large cost for incorrect predictions, especially in high-risk applications.



Trade-off between performance and model interpretability

Based on the model complexity, methods for model interpretability can be classified into *intrinsic analysis* and *post hoc* analysis.

- **Intrinsic analysis** can be applied to interpret models that have low complexity (simple relationships between the input variables and the predictions). These models are based on:
  - Algorithms, such as linear regression, where the prediction is the weighted sum of the inputs
  - Decision trees, where the prediction is based on a set of if-then rules

  The simple relationship between the inputs and output results in high model interpretability, but often leads to lower model performance, because the algorithms are unable to capture complex non-linear interactions.
- **Post hoc analysis** can be applied to interpret simpler models, as described earlier, as well as more complex models, such as neural networks, which have the ability to capture non-linear interactions. These methods are often model-agnostic and provide mechanisms to interpret a trained model based on the inputs and output predictions. Post hoc analysis can be performed at a *local* level, or at a *global* level.
  - **Local methods** enable you to zoom in on a single data point and observe the behavior of the model in that neighborhood. They are an essential component for debugging and auditing ML model predictions. Examples of local methods include:
  - **Local Interpretable Model-Agnostic Explanations** (LIME), which provides a sparse, linear approximation of the model behavior around a data point
  - **SHapley Additive exPlanations** (SHAP), a game theoretic approach based on Shapley values which computes the marginal contribution of each input variable towards the output
  - **Counterfactual explanations**, which describe the smallest change in the input variables that causes a change in the model's prediction
  - **Integrated gradients**, which provide mechanisms to attribute the model's prediction to specific input variables
  - **Saliency maps**, which are a pixel attribution method to highlight relevant pixels in an image
- **Global methods** enable you to zoom out and provide a holistic view that explains the overall behavior of the model. These methods are helpful for verifying that the model is robust and has the least possible bias to allow for fair decision making. Examples of global methods include:
  - **Aggregating local explanations**, as defined previously, across multiple data points
  - **Permutation feature importance**, which measures the importance of an input variable by computing the change in the model's prediction due to permutations of the input variable
  - **Partial dependence plots**, which plot the relationship and the marginal effect of an input variable on the model's prediction

- **Surrogate methods**, which are simpler interpretable models that are trained to approximate the behavior of the original complex model

It is recommended to start the ML journey with a simple model that is both inherently interpretable and provides sufficient model performance. In later iterations, if you need to improve the model performance, AWS recommends increasing the model complexity and leveraging post hoc analysis methods to interpret the results.

Selecting both a local method and a global method gives you the ability to interpret the behavior of the model for a single data point, as well as across all data points in the dataset. It is also essential to validate the stability of model explanations, because methods in post-hoc analysis are susceptible to adversarial attacks, where small perturbations in the input could result in large changes in the output prediction and therefore in the model explanations as well.

Amazon SageMaker Clarify provides tools to detect bias in ML models and understand model predictions. SageMaker Clarify uses a model-agnostic feature attribution approach and provides a scalable and efficient implementation of SHAP.

Model monitoring

After an ML model has been deployed to a production environment, it is important to monitor the model based on:

- **Infrastructure** — To ensure that the model has adequate compute resources to support inference workloads
- **Performance** — To ensure that the model predictions do not degrade over time

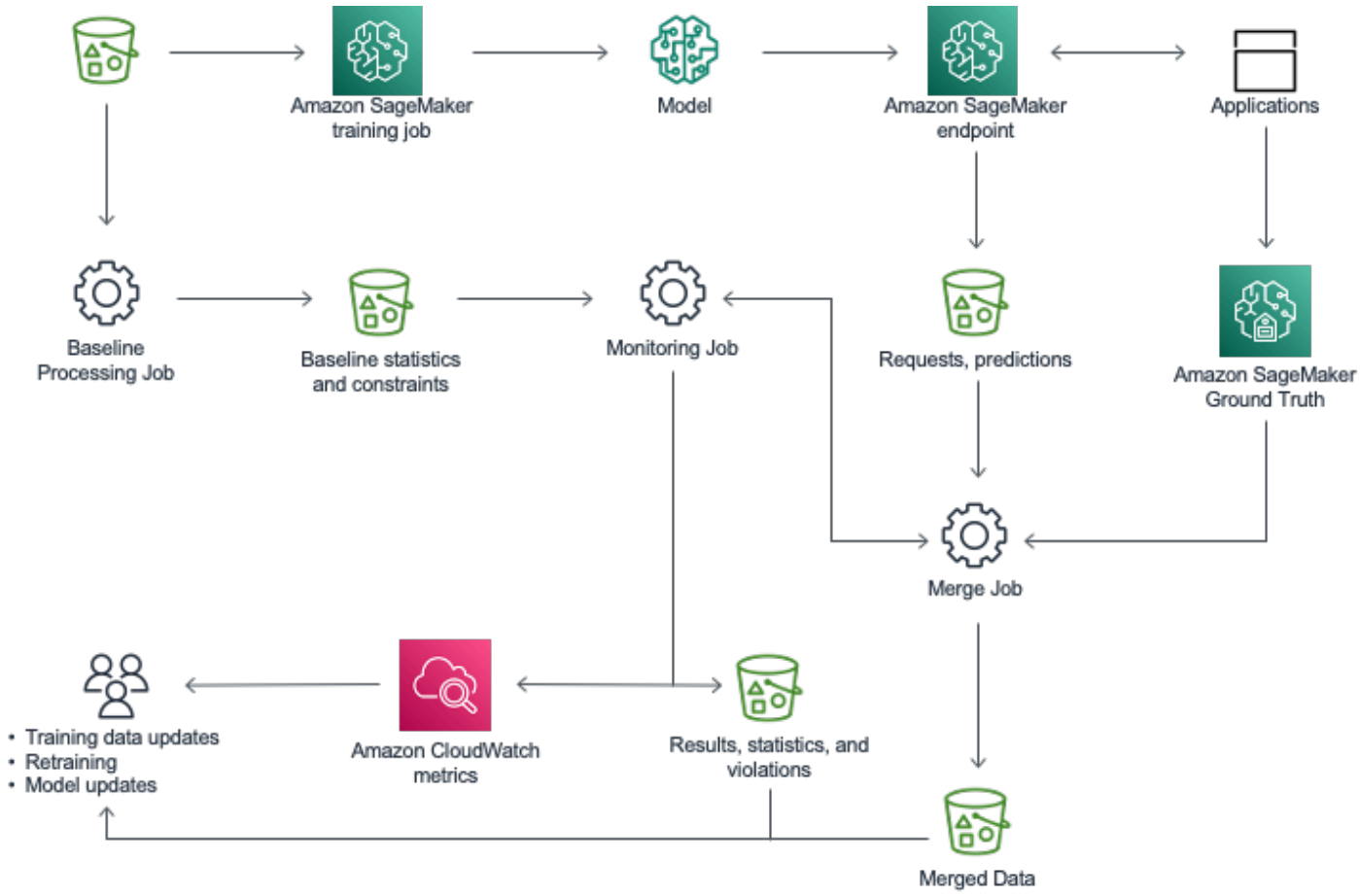
Monitoring model performance is more challenging, because the underlying patterns in the dataset are constantly evolving, which causes a static model to underperform over time.

In addition, obtaining ground truth labels for data in a production environment is expensive and time consuming. An alternative approach is to monitor the change in data and model entities with respect to a baseline. Amazon SageMaker Model Monitor can help to nearly continuously monitor the quality of ML models in production, which may play a role in postmarket vigilance by manufacturers of Software as a Medical Device (SaMD).

SageMaker Model Monitor provides the ability to monitor drift in data quality, model quality, model bias, and feature attribution. A drift in data quality arises when the statistical distribution of

data in production drifts away from the distribution of data during model training. This primarily occurs when there is a bias in selecting the training dataset; for example, where the sample of data that the model is trained on has a different distribution than that during model inference, or in non-stationary environments when the data distribution varies over time.

A drift in model quality arises when there is a significant deviation between the predictions that the model makes and the actual ground truth labels. SageMaker Model Monitor provides the ability to create a baseline to analyze the input entities, define metrics to track drift, and nearly continuously monitor both the data and model in production based on these metrics. Additionally, Model Monitor is integrated with SageMaker Clarify to identify bias in ML models.



Model deployment and monitoring for drift

For model monitoring, perform the following steps:

1. After the model has been deployed to a SageMaker endpoint, enable the endpoint to capture data from incoming requests to a trained ML model and the resulting model predictions.

2. Create a baseline from the dataset that was used to train the model. The baseline computes metrics and suggests constraints for these metrics. Real-time predictions from your model are compared to the constraints, and are reported as violations if they are outside the constrained values.
3. Create a monitoring schedule specifying what data to collect, how often to collect it, how to analyze it, and which reports to produce.
4. Inspect the reports, which compare the latest data with the baseline, and watch for any violations reported and for metrics and notifications from Amazon CloudWatch.

The drift in data or model performance can occur due to a variety of reasons, and it is essential for the technical, product, and business stakeholders to diagnose the root cause that led to the drift. Early and proactive detection of drift enables you to take corrective actions such as model retraining, auditing upstream data preparation workflows, and resolving any data quality issues.

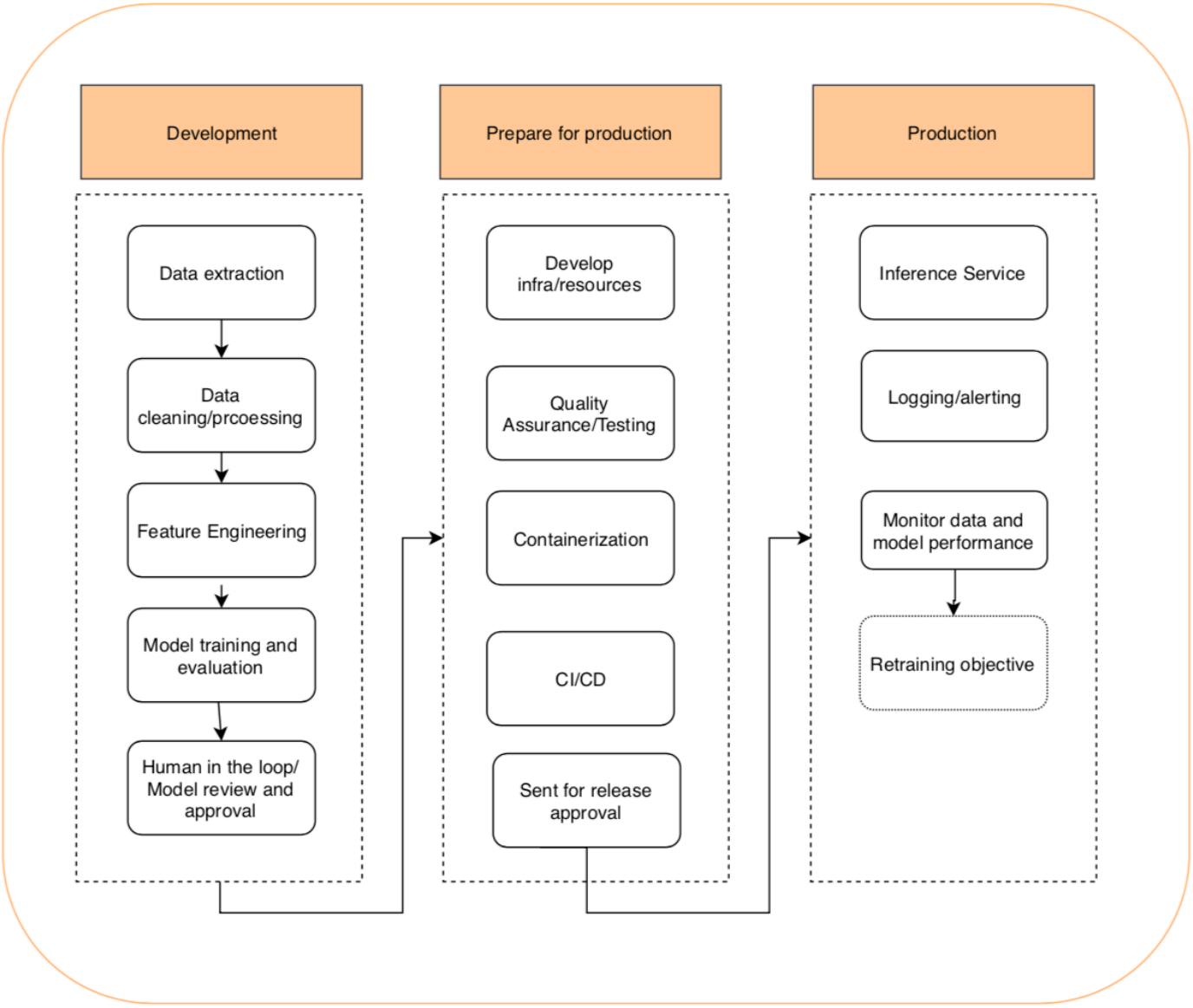
If all else remains the same, then the decision to retrain the model is based on considerations such as:

- Reevaluate target performance metrics based on the use-case
- A tradeoff between the improvement in model performance vs. the time and cost to retrain the model
- The availability of ground truth labeled data to support the desired retraining frequency

After the model is retrained, you can evaluate the candidate model performance based on a champion/challenger setup, or with A/B testing, prior to redeployment.

Operationalize AI/ML workloads

This section discusses some best practices for operationalizing MLOps workflow. MLOps (Machine Learning Operations) is a relatively new discipline that seeks to systematize the entire ML lifecycle, from science to production.



MLOps workflow

MLOps workflow

As shown in the preceding figure, the next-generation ML lifecycle breaks down the silos among all the different stakeholders that need to be involved for ML projects to capture business value. This

starts with the data acquisition and modeling activities of the data science team being informed by a clear understanding of the business objectives for the ML application, and of the governance and compliance issues that should be taken into account.

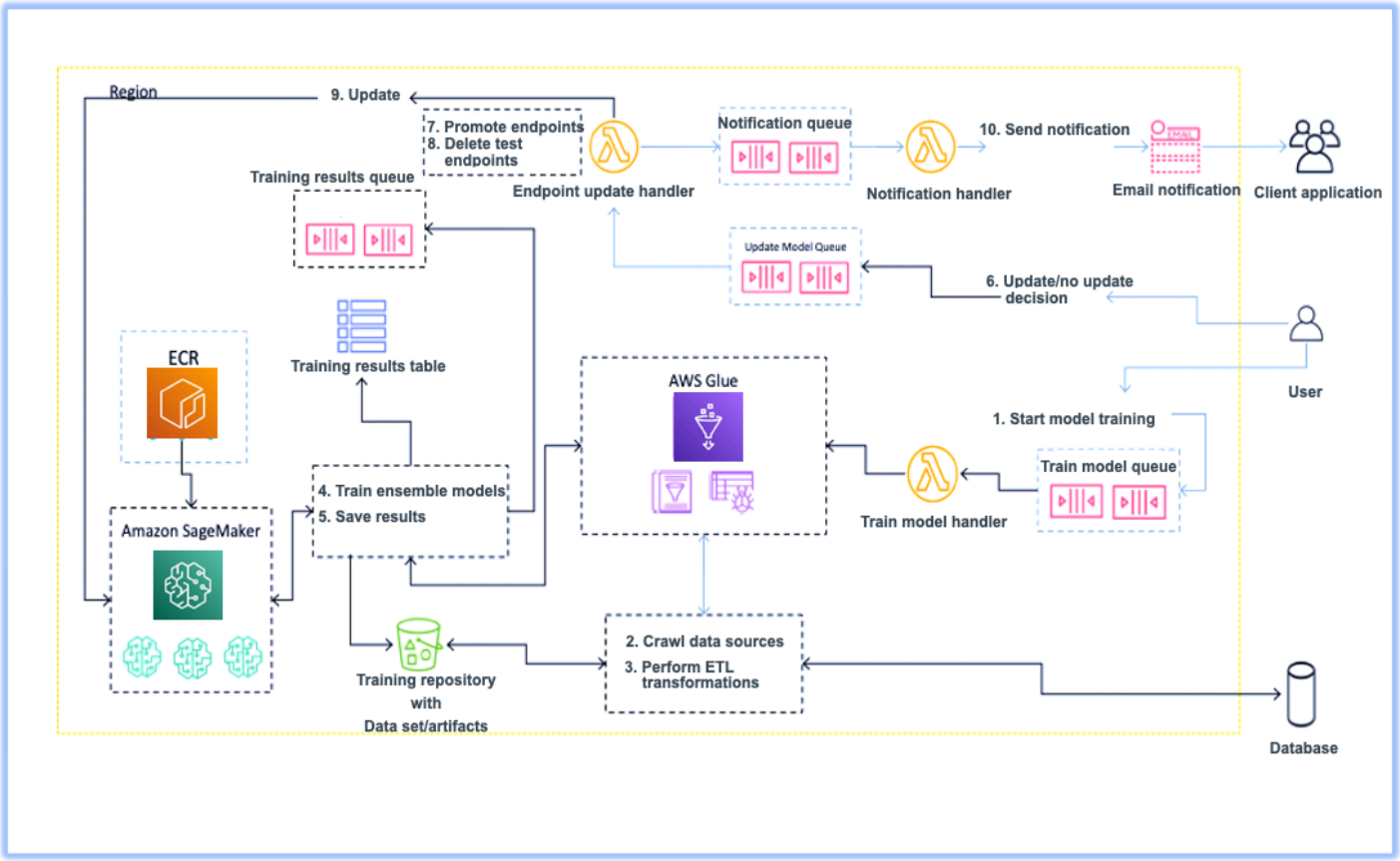
The MLOps model then ensures that the data science, production, and operations teams work seamlessly together across ML workflows that are as automated as possible. Human intervention can be incorporated as needed, ensuring smooth deployments and effective ongoing data monitoring and model performance tracking.

Performance and data drift issues are reflected back to the data science team (with “human in the loop”) so that they can tune and improve the model which is then thoroughly tested before being put into production.

Reference architectures

Training pipeline

The following shows a specific model training and tracking architecture that leverages AWS native tools and services.



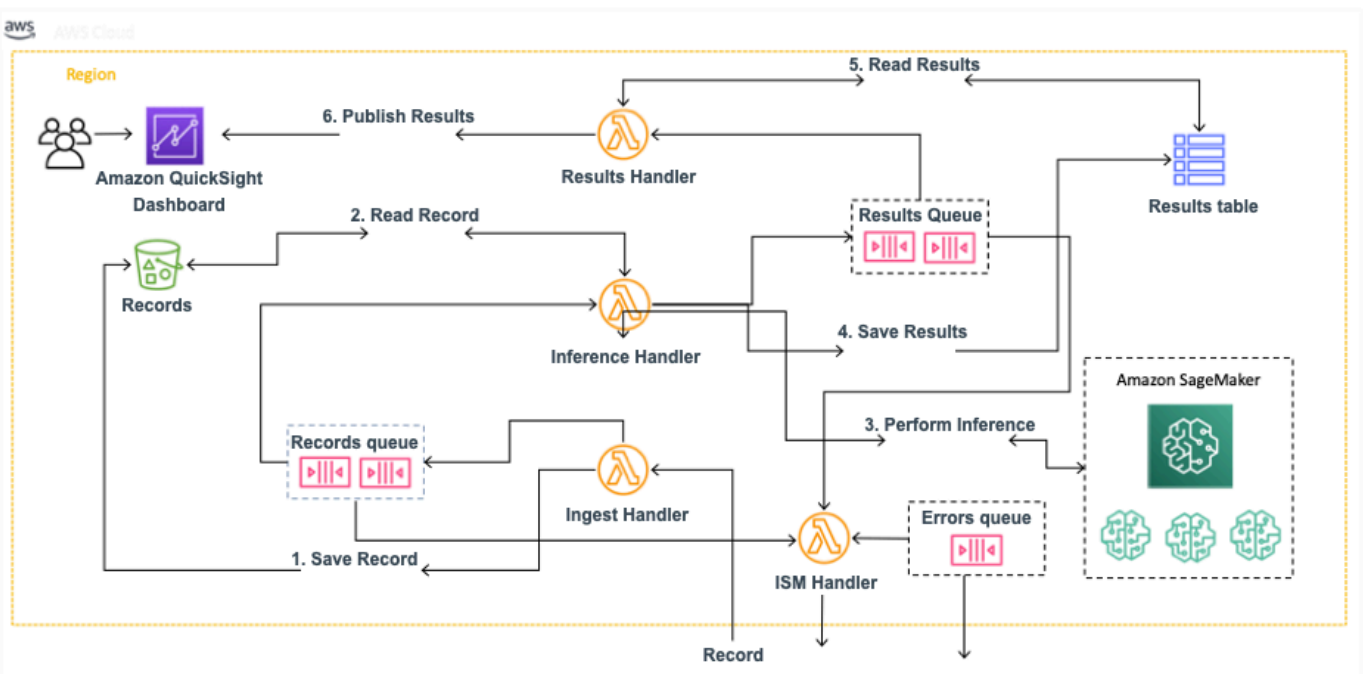
Model training and tracking architecture

- **Step 1:** Train Model Handler (Lambda) | Train Model Queue (SQS) — Model training is initiated; training could be initiated in multiple ways:
  - **Manually** (follows human in the loop design)
  - **Automated** (by scheduling relevant jobs)
- **Steps 2, 3:** AWS Glue — Reads and processes data from multiple sources. Intermediate files like train/test/validation datasets are stored in a training repository (S3)

- **Steps 4, 5:** Amazon SageMaker — Trains the ensemble models and saves the model file and train and test results into S3 and DynamoDB training results tables
- **Step 6:** Training Results Queue (SQS) — Upon completion of model training, a notification is sent to an analyst/data scientist to review the model performance results, based on the decision made by the reviewer that an action would be taken
- **Steps 7, 8, 9:** Endpoint Update Handler (Lambda) — Depending on the decision of the reviewer, either
  - The endpoint is updated with the new model, or
  - The existing endpoint is not deleted or reused
- **Step 10:** After the endpoint is updated, an email notification will be sent to the client.

Inference pipeline

The following figure shows a specific model inference architecture that leverages AWS native tools and services.



Specific model inference architecture that leverages AWS native tools and services

- **Step 1:** When the input data is available, ingest handler is invoked and necessary preprocessing steps are performed. Data is stored in Amazon S3 and a message is passed to Records Queue

- **Step 2:** Inference Handler is involved by the Records Queue, and data is read from Amazon S3
- **Step 3:** Inference Handler performs inference using Amazon SageMaker resources
- **Step 4:** After performing inference, the results are saved in the DynamoDB results table and the message is passed to the Results Queue
- **Step 5:** Results Handler then publishes the results to the Client application or a Amazon QuickSight dashboard

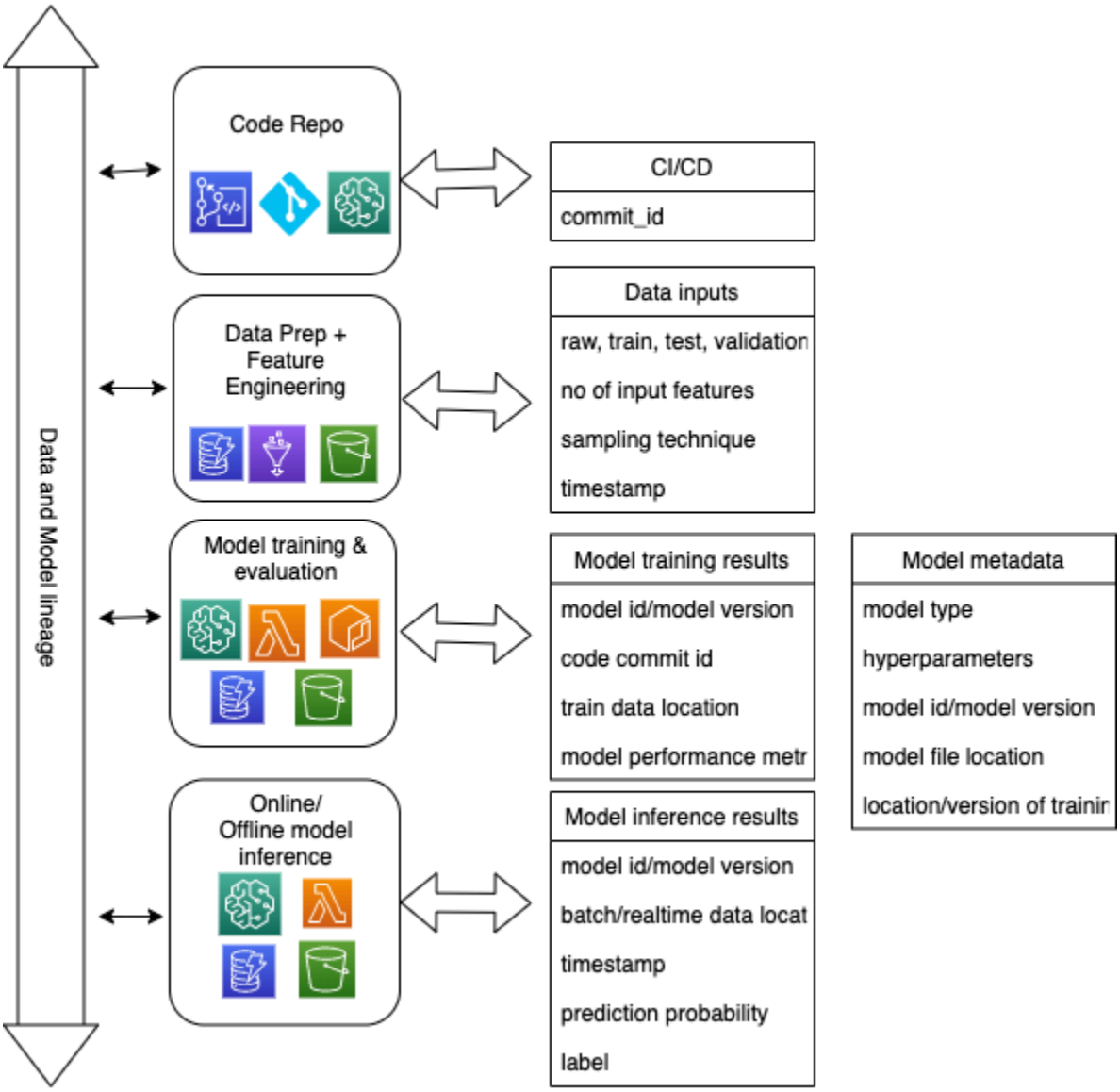
Orchestration

General guidelines for creating pipelines include using an orchestration layer, such as AWS CodePipeline.

To incorporate traceability, use the following services listed in the following architecture:

- Amazon SageMaker, a fully-managed ML service that lets data scientists and developers build, train, and deploy ML models to production
- Amazon S3, a highly scalable, reliable service to store and retrieve any amount of data, at any time, from anywhere on the web
- Amazon DynamoDB, a fully managed key-value and document database that delivers single-digit millisecond performance at any scale
- AWS Lambda, a serverless compute service to run code without provisioning or managing servers
- AWS Glue, a fully-managed data preparation service for ETL operation
- AWS CodePipeline, a fully managed nearly continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates

The following figure highlights only the key components for tracking data and model lineage:



Key components for tracking data and model lineage

- Latest code changes; `commit_id` can be extracted from AWS CodeCommit or any GitHub toolkit.
- AWS Glue Crawlers, AWS Glue jobs, or any ETL services can be used to extract raw data, run preprocessing, split data into train/test/validation datasets, and export data to Amazon S3 and metadata to DynamoDB tables.
- Amazon SageMaker can be used for model development and evaluation steps. Necessary model training results and model artifacts can be sent to Amazon S3. Necessary metadata can be sent to DynamoDB tables.

- Model performance results can be reviewed by data scientists or other human reviewers to decide if the model can be promoted to the production environment, in which case the management table model version status can be set to `ACTIVE`.
- After data is scored, input data, predicted probability, predicted label, and the model version used for performing inference can be saved in a DynamoDB inference results table, and necessary results can be sent to Amazon S3.

Orchestration for SageMaker jobs

Amazon SageMaker Pipelines can automate various tasks across the ML lifecycle. All infrastructure is fully managed. SageMaker Pipelines have the ability to:

- Orchestrate SageMaker jobs and author reproducible ML pipelines
- Deploy custom-build models for inference in real-time with low latency or offline inferences with Batch Transform
- Monitor lineage of objects

Through a simple interface, you can implement sound operating practices for deploying and monitoring production workflows, model artifact deployment, and artifact lineage tracking while adhering to safety and best practice paradigms for ML application creation.

Certain use cases may necessitate a single, larger, end-to-end pipeline that can handle anything. Other use cases, such as the following, can necessitate multiple pipelines:

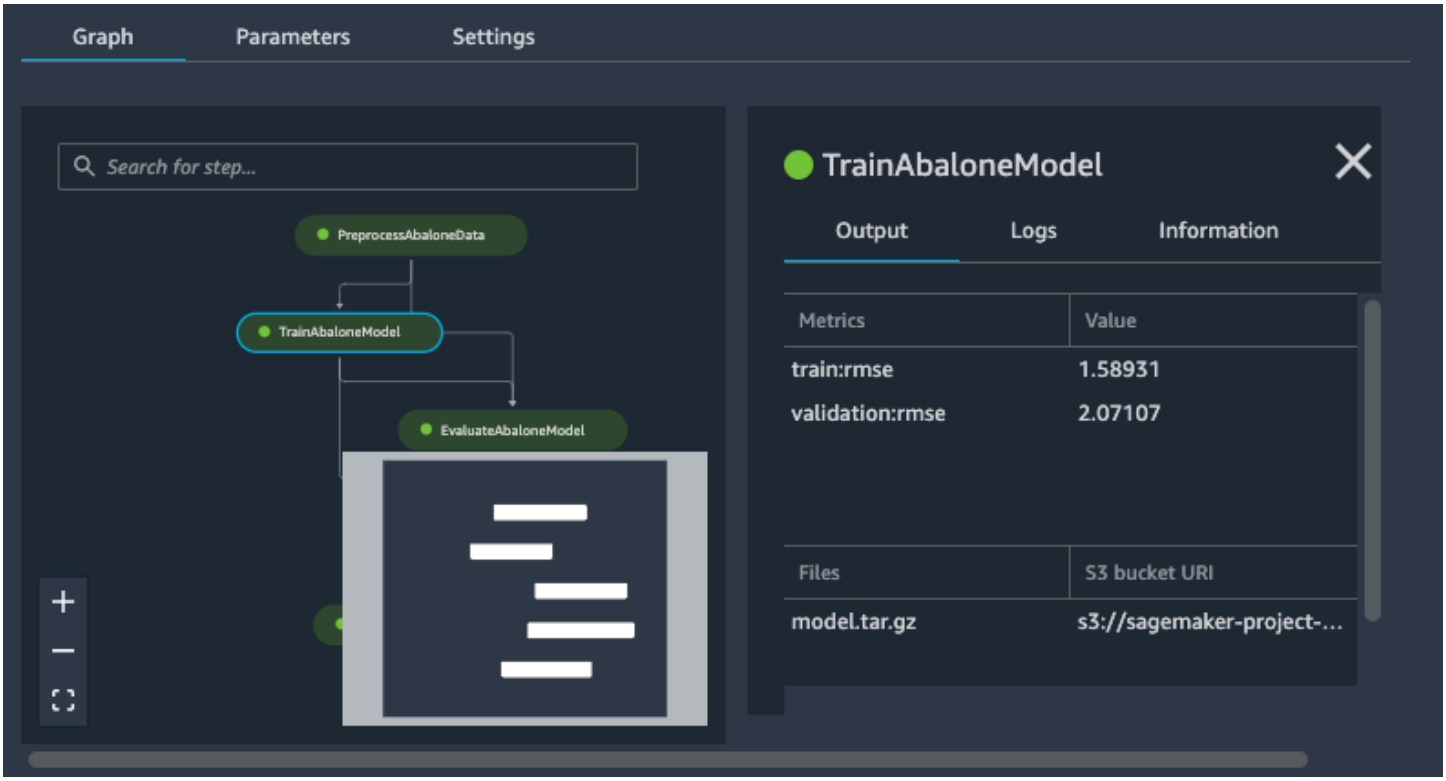
- One pipeline for ETL and data transformation steps
- One pipeline for model training, tuning, lineage, and depositing into the Model Registry
- Possibly another pipeline for specific inference scenarios (such as real-time vs. batch)
- One pipeline for triggering retraining by using SageMaker Model Monitor to detect model drift or data drift and trigger retraining.

Some of the main components in Amazon SageMaker Pipelines include pipelines, Model Registry, and MLOps templates.

- **Pipelines** – Model building pipelines are defined with a simple Python SDK. They can include any operation available in Amazon SageMaker, such as data preparation with Amazon SageMaker

Processing or Amazon SageMaker Data Wrangler, model training, model deployment to a real-time endpoint, or batch transform. You can also add Amazon SageMaker Clarify to your pipelines to detect bias prior to training, or after the model has been deployed. Likewise, you can add Amazon SageMaker Model Monitor to detect data and prediction quality issues.

- After they are launched, model building pipelines are run as CI/CD pipelines. Every step is recorded, and detailed logging information is available for traceability and debugging purposes. You can also visualize pipelines in Amazon SageMaker Studio, and track their different runs in real-time.
- **Model Registry** – The Model Registry lets you track and catalog your models. In SageMaker Studio, you can easily view model history, list and compare versions, and track metadata such as model evaluation metrics. You can also define which versions may or may not be deployed in production. You can even build pipelines that automatically trigger model deployment after approval has been given. You'll find that the Model Registry is very useful in tracing model lineage, improving model governance, and strengthening your compliance posture.
- **MLOps templates** – SageMaker Pipelines includes a collection of built-in CI/CD templates published via Service Catalog for popular pipelines (build/train/deploy, deploy only, and so on). You can also add and publish your own templates, so that your teams can efficiently discover them and deploy them. Not only do templates save lots of time, they also make it easier for ML teams to collaborate from experimentation to deployment, using standard processes, and without having to manage any infrastructure. Templates also enable Ops teams to customize steps as needed, and give them full visibility for troubleshooting.



Directed acyclic graph of steps that orchestrate SageMaker jobs

The preceding figure is a Directed Acyclic Graph (DAG) of steps and conditions that orchestrate SageMaker jobs and resource creation. Sample notebooks are available to get you started.

Conclusion

This whitepaper discusses security and GxP compliance considerations for operationalizing AI/ML workloads in the life sciences industry. It references Good Machine Learning Practices (GMLP), as referenced by the FDA, and details best practices for implementing ML workflows. However, you should work with your company's regulatory and compliance team and understand your company's policies and regulatory responsibilities before implementing these solutions, which may impact your use of ML.

Contributors

Contributors to this document include:

- Sai Sharanya Nalla, Senior Data Scientist, Amazon Web Services
- Wajahat Aziz, Principal Global Architect, Amazon Web Services
- Kartik Kannapur, Data Scientist, Amazon Web Services
- Ujjwal Ratan, Principal Architect, Amazon Web Services
- Garin Kessler, Senior Data Science Manager, Amazon Web Services
- Ian Sutcliffe, Worldwide Tech Lead, HCLS Compliance, Amazon Web Services

Document history

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Initial publication	Whitepaper first published.	November 22, 2021

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser that you are using.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.