MLOps: Continuous Delivery for Machine Learning on AWS

December 21, 2020

Notices

Contents

Abstract

Artificial intelligence (AI) is expanding into standard business processes, resulting in increased revenue and reduced costs. As AI adoption grows, it becomes increasingly important for AI and machine learning (ML) practices to focus on production quality controls. Productionizing ML models introduces challenges that span organizations and processes, involving the integration of new and incumbent technologies.

This whitepaper outlines the challenge of productionizing ML, explains some best practices, and presents solutions. ThoughtWorks, a global software consultancy, introduces the idea of MLOps as continuous delivery for machine learning. The rest of the whitepaper details solutions from AWS, Alteryx, Dataiku, Domino Data Lab, and KNIME.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems on AWS. Using the Framework allows you to learn architectural best practices for designing and operating reliable, secure, efficient, and cost-effective systems in the cloud.

In the Machine Learning Lens, we focus on how to design, deploy, and architect your machine learning workloads in the AWS Cloud. This lens adds to the best practices described in the Well-Architected Framework.

Introduction

by Christoph Windheuser, Global Head of Artificial Intelligence, ThoughtWorks
Danilo Sato, Head of Data & AI Services UK, ThoughtWorks

In modern software development, continuous delivery (CD) principles and practices have significantly improved the throughput of delivering software to production in a safe, continuous, and reliable way and helped to avoid big, disruptive, and error prone deployments.

After machine learning (ML) techniques showed that they can provide significant value, organizations started to get serious about using these new technologies and tried to get them deployed to production. However, people soon realized that training and running a machine learning model on a laptop is completely different than running it in a production IT environment. A common problem is having models that only work in a lab environment and never leave the proof-of-concept phase. Nucleus Research published a 2019 report, where they analyzed 316 AI projects in companies ranging from 20-person startups to Fortune 100 global enterprises. They found that only 38% of AI projects made it to production. Further, projects that made it to production did so in a manual ad hoc way, often then becoming stale and hard to update.

Creating a process to operationalize machine learning systems enables organizations to leverage the new and endless opportunities of machine learning to optimize processes and products. However, it also brings new challenges. Using ML models in software development makes it difficult to achieve versioning, quality control, reliability, reproducibility, explainability, and audibility in that process. This happens because there are a higher number of changing artifacts to be managed in addition to the software code, such as the datasets, the machine learning models, and the parameters and hyperparameters used by such models. And the size and portability of such artifacts can be orders of magnitude higher than the software code.

There are also organizational challenges. Different teams might own different parts of the process and have their own ways of working. Data engineers might be building pipelines to make data accessible, while data scientists can be researching and exploring better models. Machine learning engineers or developers then have to worry about how to integrate that model and release it to production. When these groups work in separate siloes, there is a high risk of creating friction in the process and delivering suboptimal results.

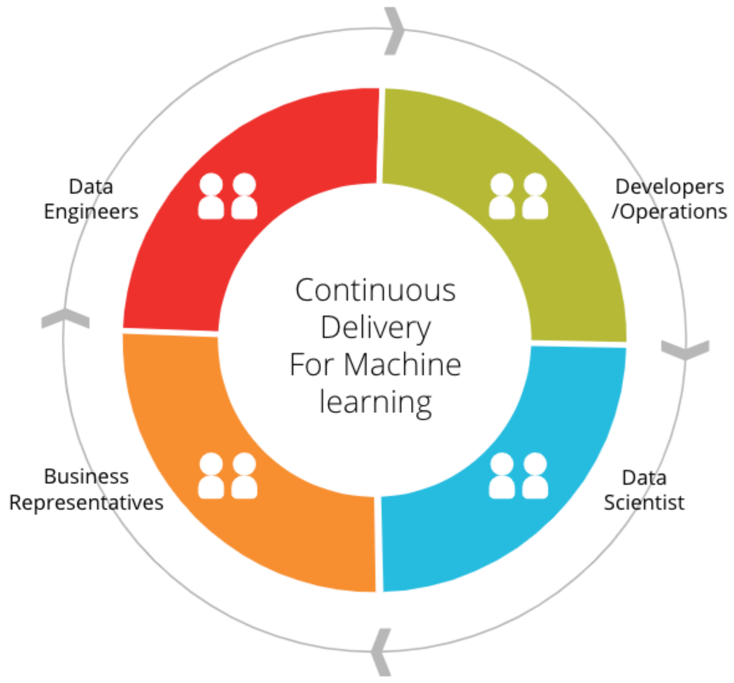


Figure 1 – The different personas usually involved in machine learning projects.

ThoughtWorks, an Advanced APN Consulting Partner, has more than 25 years of experience in professional software development. It is regarded as a strong driver in developing new software engineering processes like agile practices, continuous integration/continuous delivery (CI/CD), and DevOps.

MLOps extends DevOps into the machine learning space. It refers to the culture where people, regardless of their title or background, work together to imagine, develop, deploy, operate, and improve a machine learning system. In order to tackle the described challenges in bringing ML to production, ThoughtWorks has developed continuous delivery for machine learning (CD4ML), an approach to realize MLOps. In one of their first ML projects, ThoughtWorks built a price recommendation engine with CD4ML on AWS for AutoScout24, the largest online car marketplace in Europe. Today, CD4ML is standard at ThoughtWorks for ML projects.

Continuous delivery for machine learning

Continuous delivery has been the approach to bring automation, quality, and discipline to create a reliable and repeatable process of releasing software into production. Jez Humble, one of the authors of the seminal book *Continuous Delivery*[1], states that:

"Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes, and experiments—into production, or into the hands of users, safely and quickly in a sustainable way".[2]

Continuous delivery applies to changes of all types, not just software code. With that in mind, we can extend its definition to incorporate the new elements and challenges that exist in real-world machine learning systems, an approach we are calling *Continuous Delivery for Machine Learning* (CD4ML).

"Continuous Delivery for Machine Learning (CD4ML) is a software engineering approach in which a cross-functional team produces machine learning applications based on code, data, and models in small and safe increments that can be reproduced and reliably released at any time, in short adaptation cycles."[3]

This definition includes the core principles to strive for. It highlights the importance of cross-functional teams with skill sets across different areas of specialization, such as: data engineering, data science, or operations. It incorporates other sources of change beyond code and configuration, such as datasets, models, and parameters. It calls for an incremental and reliable process to make small changes frequently, in a safe way, which reduces the risk of big releases. Finally, it requires a feedback loop: The real-world data is continuously changing and the models in productions are continuously monitored, leading to adaptations and improvements by re-training of the models and the re-iteration of the whole process.

The different process steps of CD4ML

Figure 2 shows the end-to-end process of CD4ML. Let's have a closer look at the different steps.

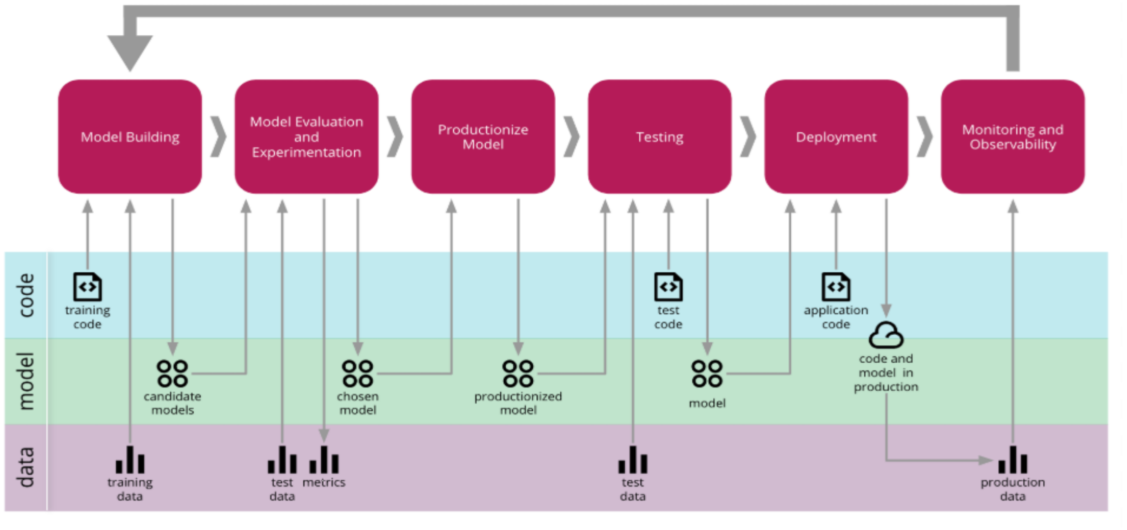


Figure 2 – Continuous delivery for machine learning end-to-end process

Model building

Once the need for a machine learning system is found, data scientists will research and experiment to develop the best model, by trying different combinations of algorithms, and tuning their parameters and hyperparameters. This will produce models that can be evaluated to assess the quality of its predictions. The formal implementation of this model training process becomes the machine learning pipeline.

Having an automated and reproducible machine learning pipeline allows other data scientists to collaborate on the same code base, but also allows it to be executed in different environments, against different datasets. This provides great flexibility to scale out and track the execution of multiple experiments and ensures their reproducibility.

Model evaluation and experimentation

As the data science process is very research-oriented, it is common that there will be multiple experiments running in parallel, and many of them will never make their way to production. This will require an approach that keeps track of all the different experiments, different code versions, and potentially different datasets and parameters/hyperparameters attempted. Your CD4ML architecture will need to support tracking, visualizing, comparing results from different runs, as well as to support the graduation and promotion of models that prove to be useful.

Productionize the model

Once a suitable model is found, you will need to decide how it will be served and used in production. It might be embedded and packaged within the application that consumes it, it might be deployed as a separate service, or it might be treated as data that is loaded at runtime.

Regardless of which pattern you choose, there will always be an implicit contract (API) between the model and how it is consumed. If the contract changes, it will cause an integration bug.

Testing and quality

Machine learning workflows require different types of testing. Some are inherently non-deterministic and hard to automate, while others can be automated to add value and improve the overall quality of the system. Examples include data validation and data quality, component integration and contract testing, evaluating model quality metrics, and validating the model against bias and fairness.

While you cannot write a deterministic test to assert the model score from a given training run, the CD4ML process can automate the collection of such metrics and track their trend over time. This allows you to introduce quality gates that fail when they cross a configurable threshold and to ensure that models don't degrade against known performance baselines.

Deployment

Once a good candidate model is found, it must be deployed to production. There are different approaches to do that with minimal disruption. You can have multiple models performing the same task for different partitions of the problem. You can have a shadow model deployed side by side with the current one to monitor its performance before promoting it. You can have competing models being actively used by different segments of the user base. Or you can have online learning models that are continuously improving with the arrival of new data.

Elastic cloud infrastructure is a key enabler for implementing these different deployment scenarios while minimizing any potential downtime, allowing you to scale the infrastructure up and down on-demand, as they are rolled out.

Monitoring and observability and closing the feedback loop

Once the model is live, you will need the monitoring and observability infrastructure to understand how it is performing in production against real data. By capturing this data, you can close the data feedback loop. A human in the loop can analyze the new data captured from production, curate, and label it to create new training datasets for improving future models. This enables models to adapt and creates a process of continuous improvement.

The technical components of CD4ML

To be able to perform these process steps in an automated and continuous way, CD4ML requires the right infrastructure and tooling. The following image shows the components of the CD4ML infrastructure.

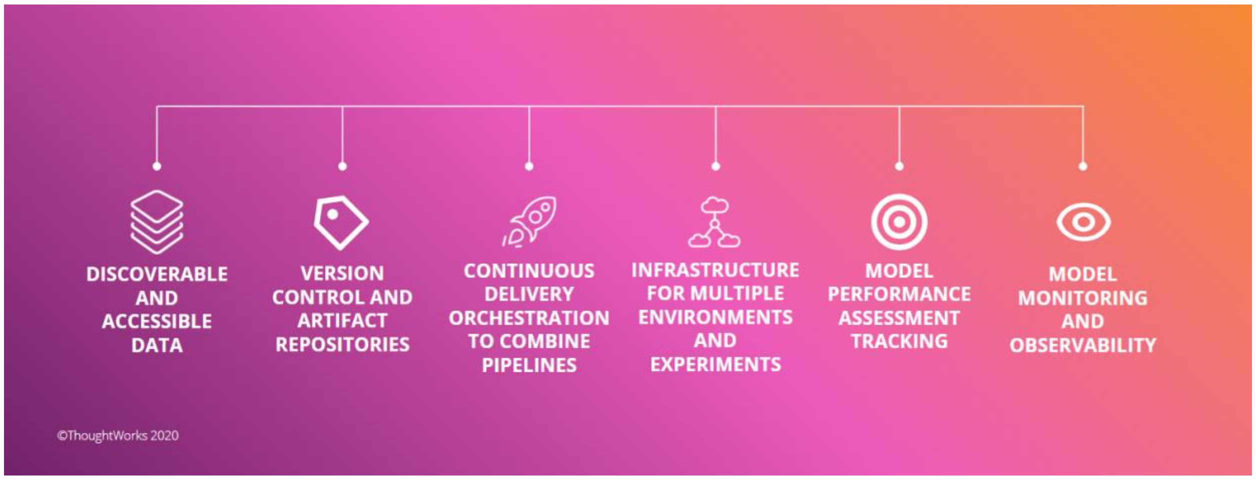

DISCOVERABLE AND ACCESSIBLE DATA
VERSION CONTROL AND ARTIFACT REPOSITORIES
CONTINUOUS DELIVERY ORCHESTRATION TO COMBINE PIPELINES
INFRASTRUCTURE FOR MULTIPLE ENVIRONMENTS AND EXPERIMENTS
MODEL PERFORMANCE ASSESSMENT TRACKING
MODEL MONITORING AND OBSERVABILITY
©ThoughtWorks 2020

Figure 3 – Technical components of CD4ML

Let's look at the different components of the CD4ML infrastructure in more detail.

Discoverable and accessible data

Good machine learning models require good data. And data must be easily discoverable and accessible. Data can be generated and collected from inside or outside your organization. There are several approaches to building a modern data platform, but it must be architected to consider the needs of different stakeholder groups: From application teams building systems that generate or collect data, to teams responsible for data governance, to business stakeholders that use such data for decision-making, and to data scientists who automate it through an AI/ML system.

Version control and artifact repositories

Like in modern software engineering projects, version control tools are mandatory for working efficiently as a team of people and ensuring reproducibility of the results. Machine learning projects bring the additional complexity of data as a new artifact.

Continuous delivery orchestration

To tie everything together, a good continuous delivery orchestration tool will allow you to model the intricate dependencies and workflows to implement the end-to-end CD4ML process. They provide means to automate the execution of deployment pipelines, and to model the data governance process required to promote models from research to production.

Infrastructure for multiple environments

While not a hard requirement, using cloud infrastructure will bring many benefits to CD4ML:

- Elasticity and scalability to use more or less resources as needed
- The ability to use specialized hardware (such as GPUs) for training machine learning models more efficiently
- Access to specialized services to manage your data and machine learning platforms
- The ability to use a “pay-as-you-go” model based on actual usage of the underlying infrastructure

Model performance assessment tracking

Because data science and machine learning development can be highly experimental and repetitive, it is important to have the possibility to track the different models, their parameters, and results in a way that is transparent to the whole development team.

Model monitoring and observability

Machine learning models in production have to be continuously monitored to answer questions like: How does the model perform? What is the actual data fed into the model? Does the data have a different characteristic than the training set? Is the performance deteriorating? Is there bias in the data or model performance?

AWS Solutions

CD4ML is a process for bringing automation, quality, and discipline to the practice of releasing ML software into production frequently and safely. It is not bound to a specific infrastructure or toolset.

CD4ML prescribes an approach to MLOps, which is a practice that spans the management of a production ML lifecycle. We present this approach because it has demonstrated success, and is one that incorporates modern, cloud-friendly software development practices.

This whitepaper showcases MLOps solutions from AWS and the following AWS Partner Network (APN) companies that can deliver on the previously mentioned requirements:

- Alteryx
- Dataiku
- Domino Data Lab
- KNIME

These solutions offer a broad spectrum of experiences that cater to builders and those who desire no-to-low-code experiences. AWS and the APN provide you with choices and the ability to tailor solutions that best fit your organization.

Alteryx

by Alex Sadovsky, Senior Director of Product Management, Alteryx
David Cooperberg, Senior Product Manager, Alteryx

As a global leader in analytics process automation (APA) software, Alteryx unifies analytics, data science, and business process automation in a single, end-to-end platform that accelerates digital transformation. Alteryx powers the analytics, insights, and models at thousands of the world's largest companies. Its platform helps business analysts and data scientists across industries solve problems with data in a code-free environment.

The Alteryx platform consists of four components:

- **Alteryx Connect** – a collaborative data cataloging tool
- **Alteryx Designer** – desktop software that enables the assembly of code-free analytic workflows and apps
- **Alteryx Server** – an analytical hub that allows users to scale their analytic capabilities in the cloud or on premises on enterprise hardware
- **Alteryx Promote** – a deployable, containerized solution that enables the easy deployment of machine learning models as highly available REST APIs

The remainder of this section will discuss how Alteryx can be used in conjunction with AWS to achieve a comprehensive CD4ML solution.

Data governance and curation

Figure 4 highlights how Alteryx Connect can be used to catalog data from disparate sources, including the datasets Alteryx offers as add-ons.

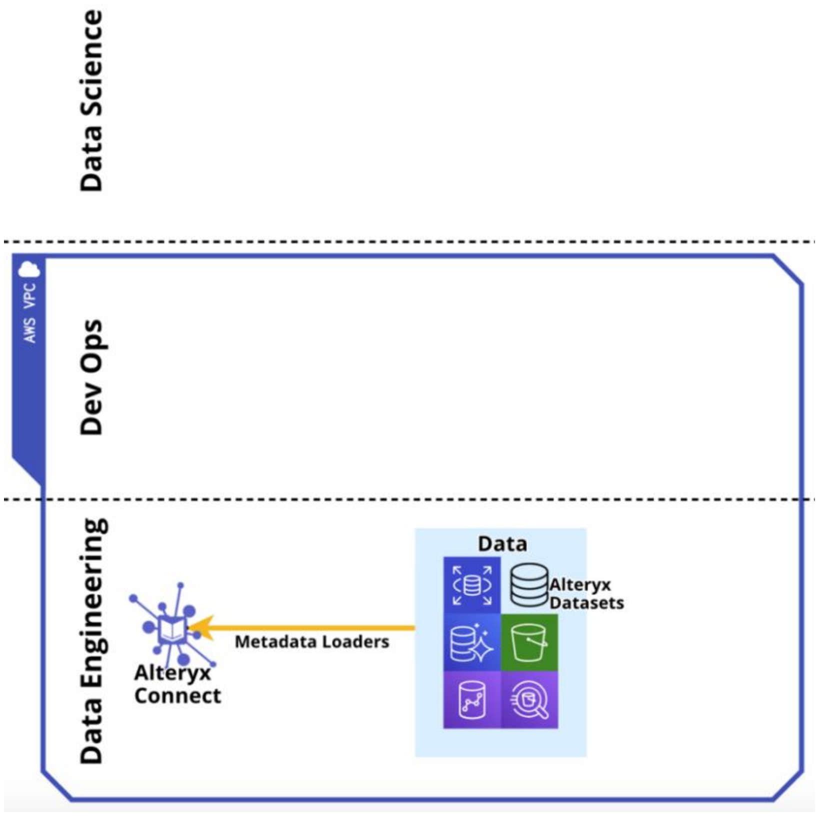


Figure 4 – How to catalog data sources with Alteryx Connect

Connect makes it easy for users to discover and understand relevant data assets. Once a data source is represented in Connect, users collaborate using social validation tools like voting, commenting, and sharing to highlight the usefulness and freshness of the data. Connect installs in a Windows Server environment running in Amazon EC2. Once Connect is installed, one or more of the 25+ existing database metadata loaders are used to add data sources. This includes loaders for Amazon Redshift and Amazon S3, and loaders for Postgres and MySQL that can load metadata from Amazon Aurora. If a data source is missing a metadata loader, Alteryx offers intuitive SDKs that make writing new loaders easy for developers in multiple languages and via REST APIs. Connect offers a cross-platform experience, allowing desktop Designer users and Server users to explore and utilize data assets based upon shared metadata.

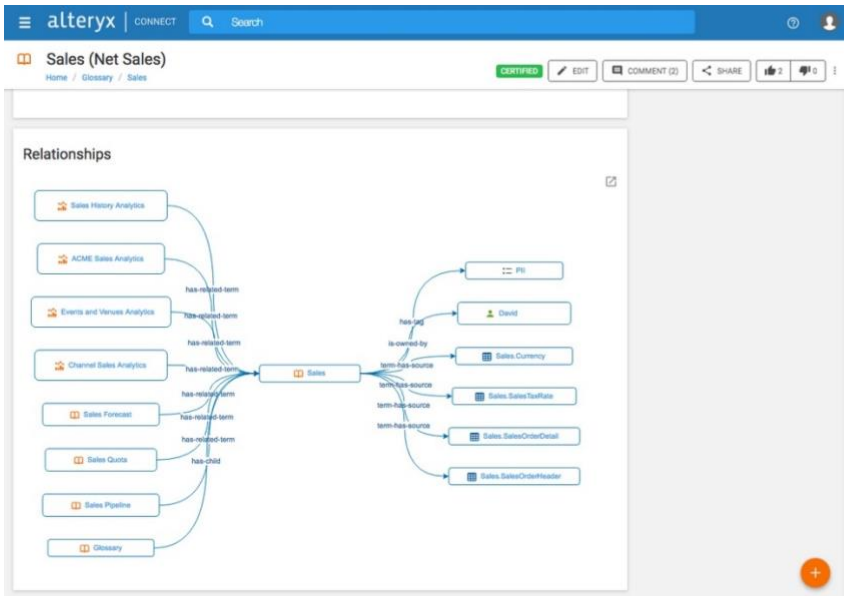


Figure 5 – Data asset lineage in Alteryx Connect

Alteryx also offers the ability to augment user data with datasets from industry data providers. Alteryx Datasets can provide valuable location and business insights when combined with proprietary data. In the modeling realm, this data is most typically paired with proprietary data, to offer demographic and geographic features into models.

Machine learning experimentation

Figure 6 shows how Alteryx Designer can be used to import data for use in any of several predictive modeling and machine learning experimentation tool suites. Each tool suite caters to users with a different level of experience with machine learning and aims to upskill users through usage.

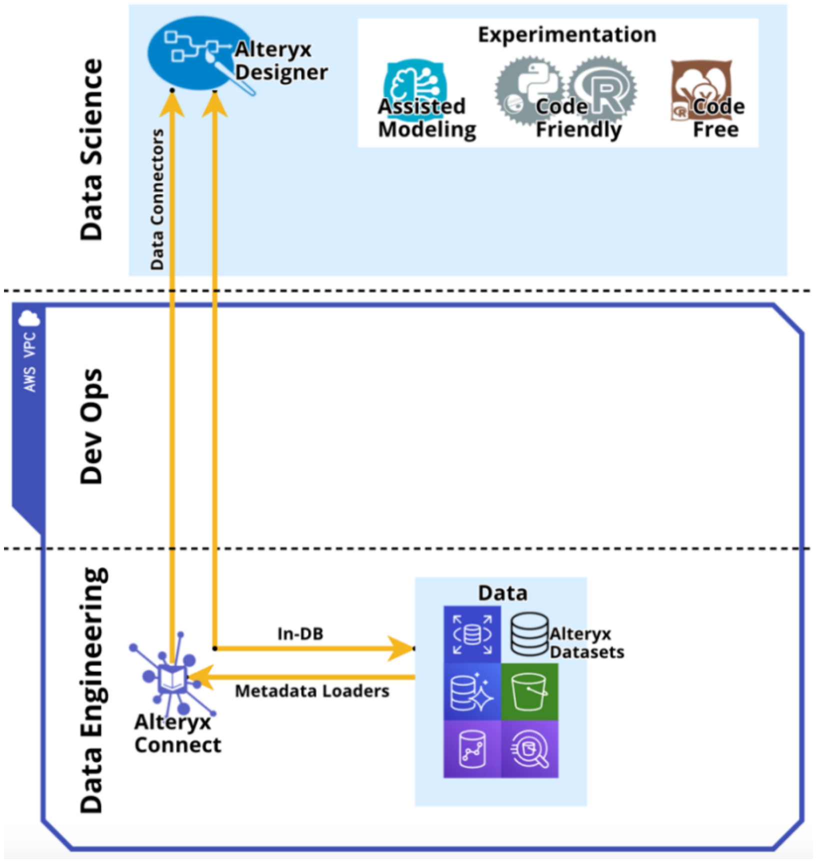


Figure 6 – Alteryx Designer offers several options for modeling and experimentation based on a user's level of experience

Once a data architecture is implemented and the appropriate data assets are identified, analytics can begin. Designer, a code-free and code-friendly development environment, enables analysts of all skill levels to create analytic workflows, including those that require machine learning. Designer can be installed on a local Windows machine.

The first step in any analytic process is to import data. Alteryx is agnostic to where and how data is stored, providing connectors to over 80 different data sources, including an AWS Starter Kit that includes connectors for Amazon Athena, Amazon Aurora, Amazon S3, and Amazon Redshift. Because Alteryx provides a common ground to data processing from multiple sources, for high performant workloads, it is often a best practice to co-localize the data by preprocessing workflows. For example, in order to reduce future processing latency, you could move on-premises data to an AWS source. This can all be done with drag and drop, code free data connector building blocks, avoiding the need to know any CLI/SQL intricacies of the underlying infrastructure, although the latter is possible as well.

Designer includes over 260 automation building blocks that enable the code-free processing of data. This includes building blocks for data prep, cleansing, blending, mapping, visualization, and modeling. Data cleansing, blending, and prep building blocks are often used before machine learning experimentation to prepare training, test, and validation datasets.

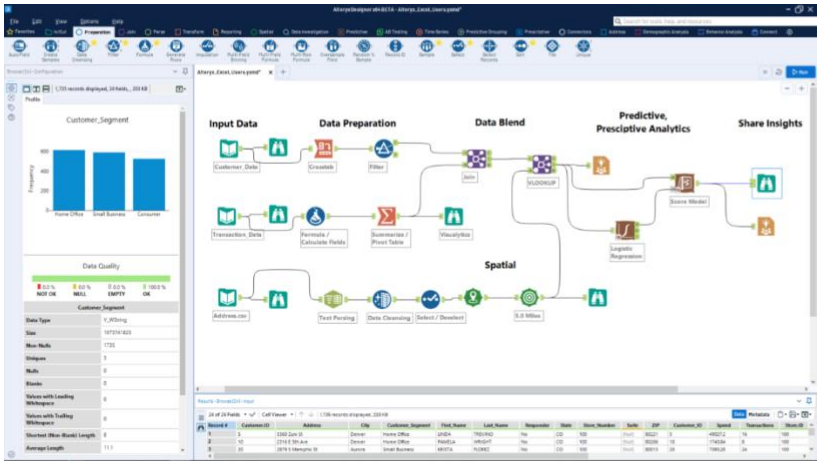


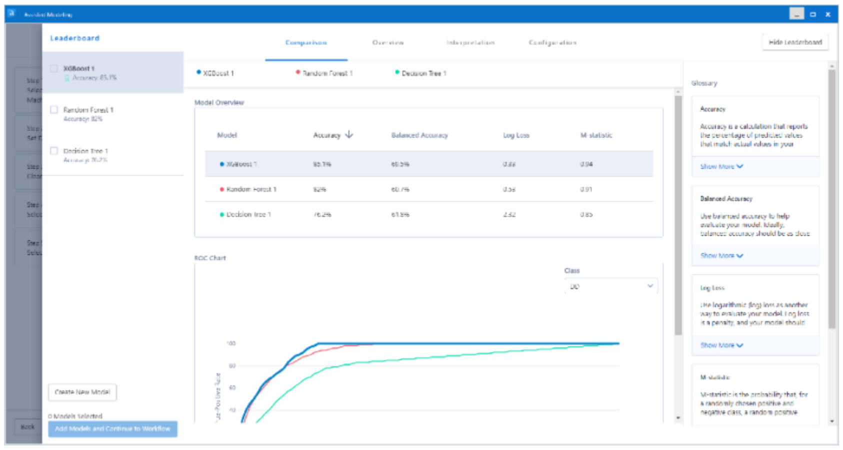
Figure 7 – Build complex analytic workflows in Alteryx Designer

Much of the data preprocessing that occurs before modeling can also be accomplished using Alteryx's [In-Database functionality](). This functionality pushes down data processing tasks to the database and delays the data import until after that processing has been completed and a local machine in-memory action needs to be executed.

Alteryx Designer provides users with several choices for machine learning:

- The **Alteryx Predictive Suite** offers code-free functionality for many descriptive, predictive, and prescriptive analytics tasks. Users can also customize the underlying R code that powers these building blocks to address their specific use cases.
- The **Alteryx Intelligence Suite** offers code-free functionality for building machine learning pipelines and additional functionality for text analytics. The Intelligence Suite also offers [Assisted Modeling](), an automated modeling product designed to help business analysts learn machine learning while building validated models that solve their specific business problems. Assisted Modeling is built on open-source libraries and provides users with the option to export their drag-and-drop or wizard-created models as Python scripts.

- **Code-friendly** building blocks that support R and Python allow users to write machine learning code that is embedded in an otherwise code-free workflow. Users can use these building blocks to work with their preferred frameworks and libraries, and the built-in Jupyter notebook integration enables interactive data experimentation.



Model Overview

Model	Accuracy	Balanced Accuracy	Log Loss	M-statistic
● XGBoost 1	85.1%	65.5%	0.69	0.94
● Random Forest 1	82%	60.7%	0.58	0.91
● Decision Tree 1	76.2%	61.9%	2.32	0.85

Accuracy

Accuracy is a calculation that reports the percentage of predicted values that match actual values in your

Balanced Accuracy

Use balanced accuracy to help evaluate your model. Ideally, balanced accuracy should be as close

Log Loss

Use logarithmic (log) loss as another way to evaluate your model. Log loss is a penalty, and your model should

M-statistic

M-statistic is the probability that, for a randomly chosen positive and negative class, a random positive

Figure 8 – Compare trained models in the Assisted Modeling leaderboard

Productionized ML pipelines

Figure 9 illustrates how Alteryx Server can be leveraged to operationalize workflows, including those that are used for data governance. Server offers a componentized installation experience that works natively in AWS.

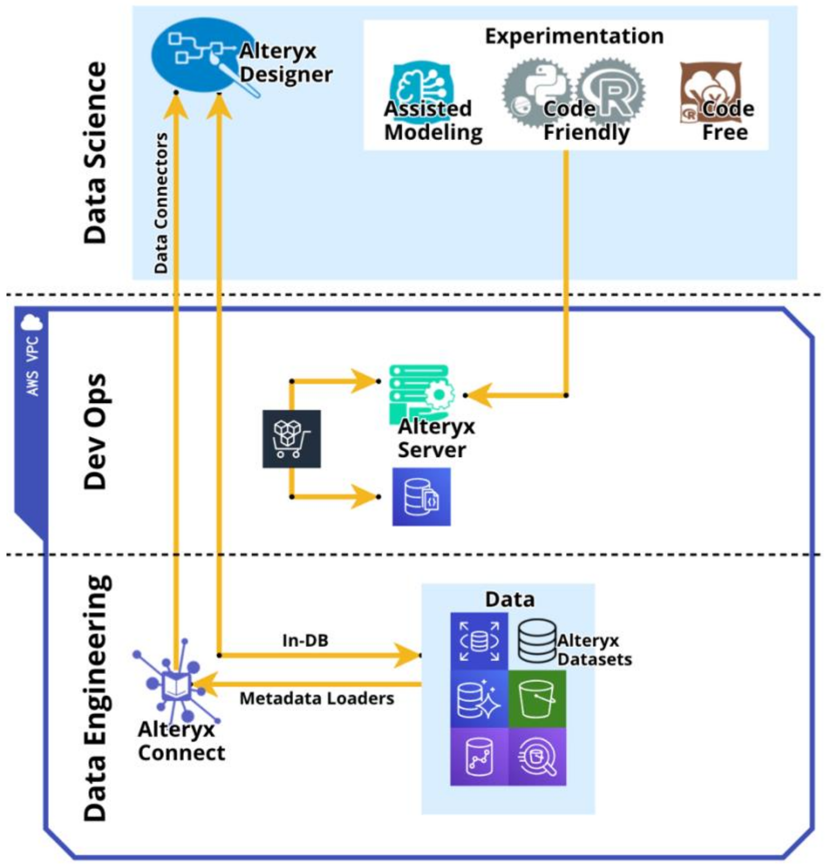


Figure 9 – Alteryx Server can be installed easily in AWS to productionize machine learning and data governance workflows

As experimentation begins to yield promising results, it's often time to scale modeling to support larger training data, hyperparameter tuning, and productionization.

Alteryx Server is a scalable environment that is used to manage and deploy analytic assets. It's easy to append CPU-optimized machines to a Server cluster that can be specified for use by machine learning training pipelines. Executing long-running training jobs in Server offers users the flexibility to continue designing analytic workflows in Designer while the training job executes.

Server enables the scheduling and sequencing of analytic workflows. Each of these features can be used as part of CI/CD pipelines that ensure the quality of models that have been deployed to production. Using REST APIs, workflows can be programmatically triggered and monitored for status to integrate into established DevOps and CI/CD setups.

Alteryx Server can be installed in an on-premises data center or in the AWS Cloud, and supports single and multi-node configurations. It's offered as an Amazon Machine Image (AMI) in the AWS Marketplace for easy one-click deployments. Customized instances can also be deployed in a private subnet using Amazon Virtual Private Cloud. Server offers many options for customization, one of which is the option to store Server metadata in a user-managed MongoDB instance, for which AWS offers a Quick Start. For detailed guidance, see Best Practices for Deploying Alteryx Server on AWS.

Alteryx Server offers built-in governance and version control of analytic assets, which can be used in place of or in addition to other source control solutions.

Model serving and deployment

Figure 10 shows how Alteryx Promote ties the platform together, offering a solution for model management, real-time model serving, and model monitoring.

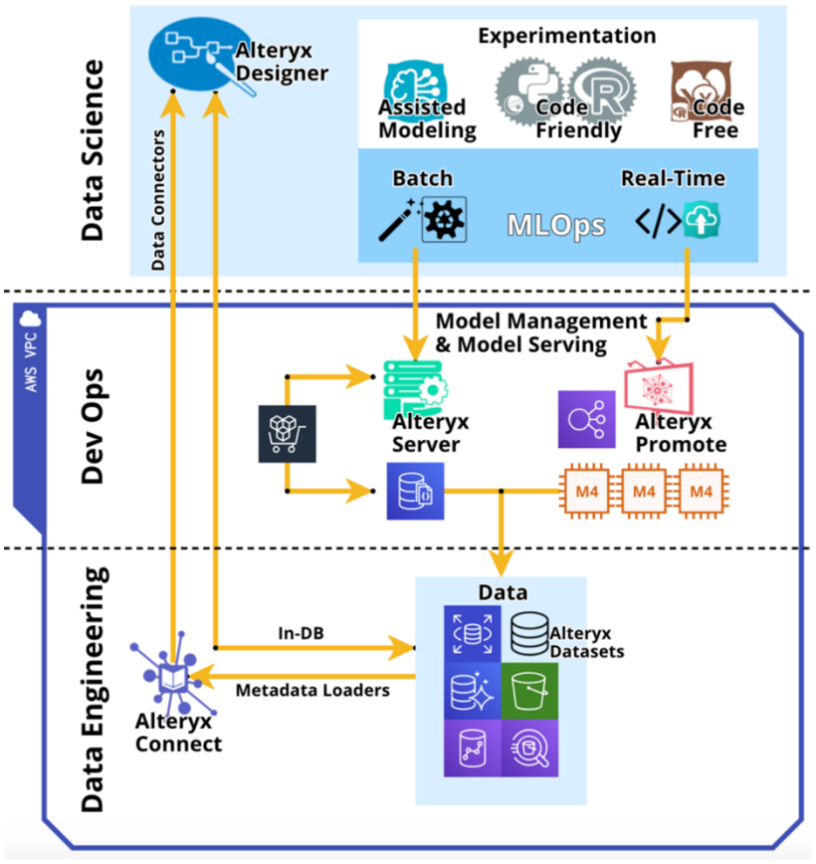
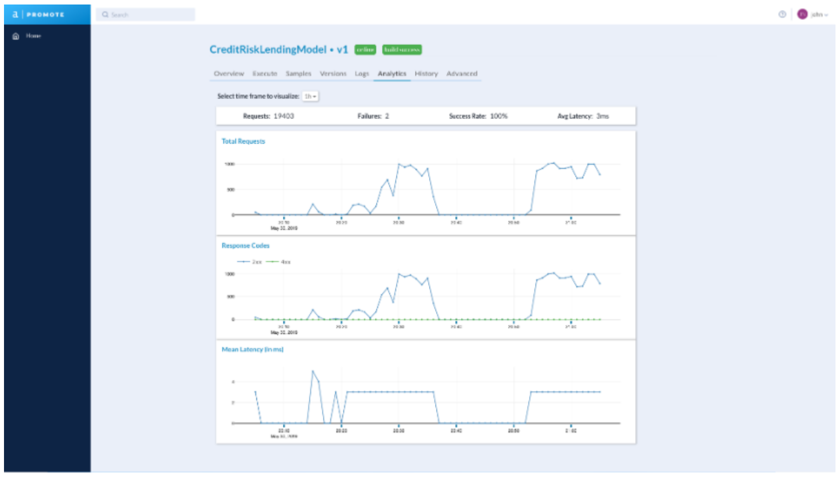


Figure 10 – Alteryx Promote offers a MLOps solution providing model management and highly-available, low-latency model serving

The Alteryx platform offers several options for model deployment. Promote is used primarily for real-time deployments, common for models that interact with web applications. Promote enables the rapid deployment of pre-trained machine learning models through easy-to-use Python and R client libraries or in a code-free manner using Alteryx Designer.

Models that have been deployed to a Promote cluster server environment are packaged as Docker containers, replicated across nodes, and made accessible as highly available REST APIs that host in-memory inference methods. The number of replications of each model is configurable, as is the number of nodes available in the Promote cluster. An internal load balancer spreads requests across the available replications.



Select time frame to visualize: 1h

Requests: 19403 Failures: 2 Success Rate: 100% Avg Latency: 3ms

Figure 11 – Monitor the performance of your models in production with Promote

Like Server and Connect, Promote can be installed in an AWS Cloud environment or in an on-premises data center. The recommended setup also includes an external load balancer, such as Elastic Load Balancing, in order to distribute prediction requests across every Promote node. Promote is ideal for inference cases in which throughput is already known or is acceptable to be changed on demand. While automatic scaling is technically possible, it’s beyond the intended use of the product.

Alteryx Server is the recommended solution for models that require batch inference on known existing hardware. Batch models can be packaged for prediction in workflow or analytic apps and can be scheduled to run in Server on compute-optimized nodes. Server’s workflow management functionality can also be leveraged to ensure that predictions are made only after up-to-date features have been generated through data preprocessing.

Additionally, users often find they need a hybrid of Alteryx and AWS solutions to deploy complex models at scale. One usage pattern we have observed is using our Assisted Modeling tool on the desktop to prototype a model on sample data. Using Designer and Server, clients prep/blend data from local sources and push the resulting data to S3. Then, the model code from Assisted Modeling can be pushed to SageMaker, where the model can be trained on the entire dataset resident in Amazon S3, and deployed as an API in the SageMaker ecosystem to take advantage of containerization, scaling, and serverless capabilities. As Alteryx focuses on friendly model building, this is often the best path for organizations who are light in data science, but who have heavy DevOps or engineering resources.

Model testing and quality

Alteryx enables model testing throughout the modeling and deployment process. During the experimentation phase, Predictive building blocks and Assisted Modeling report performance metrics and visualizations, making it possible to compare the generalizability of each model. Assisted Modeling also offers Explainable AI (XAI) reporting in the form of feature importance scores, calculated using the permutation importance approach.

During model deployment, it’s easy to add test data to a Promote deployment script. The testing step can be used to conditionally allow or disallow the deployment of that model version. New Promote model versions are initially hosted in logical development and staging environments, allowing users to run a new model in parallel with the previously running production model. Testers can set up their systems to make predictions on both the production and staging model versions before deciding to replace the production model, which is accomplishable using an API. Promote also records all request and response data, making it possible for users to develop custom workflows that leverage that data to test for bias, fairness, and concept drift.

Continuous improvement

In addition to recording all incoming requests and their responses, Promote tracks aggregated metrics in Amazon Elasticsearch Service so administrators can observe the performance of the models they have deployed. Metrics for requests, errors, and latency over the previous month inform whether the model needs to be replicated further. Additional system utilization reporting helps administrators determine if additional nodes must be added to the Promote cluster.

Finally, users can export the historical request data to be analyzed for concept or data drift. These analyses can be performed in Alteryx Designer, scheduled to run in Server, and can kick off the CD pipeline if drift is detected.

Alteryx: Your journey ahead

As an end-to-end analytic process automation platform, Alteryx provides the data connectors, building blocks, and functionality to create and deploy modeling solutions, often with little to no coding required. Our open ecosystem in terms of APIs, third-party data connectors, and open-source solutions provides developers the ability to mix and match the Alteryx solution with AWS native components. This gives customers the freedom to deploy machine learning as best fits their business requirements.

Get started with our Intelligence Suite Starter Kit or an interactive demo of Alteryx Designer. Ready to scale? Learn about Best Practices for Deploying Alteryx Server on AWS and deploy Alteryx Server from the AWS Marketplace.

Dataiku DSS

by Greg Willis, Director of Solutions Architecture, Dataiku

Dataiku is one of the world’s leading AI and machine learning platforms, supporting agility in organizations’ data efforts via collaborative, elastic, and responsible AI, all at enterprise scale. At its core, Dataiku believes that in order to stay relevant in today’s changing world, companies need to harness Enterprise AI as a widespread organizational asset instead of siloing it into a specific team or role.

To make this vision of Enterprise AI a reality, Dataiku provides a unified user interface (UI) to orchestrate the entire machine learning lifecycle, from data connectivity, preparation, and exploration to machine learning model building, deployment, monitoring, and everything in between.

Dataiku was built from the ground up to support different user profiles collaborating in every step of the process, from data scientist to cloud architect to analyst. Point-and-click features allow those on the business side and other non-coders to explore data and apply automated machine learning (AutoML) using a visual interface. At the same time, robust coding features (including interactive Python, R, Spark, and SQL notebooks) ensure that data scientists and other coders are first-class citizens as well.

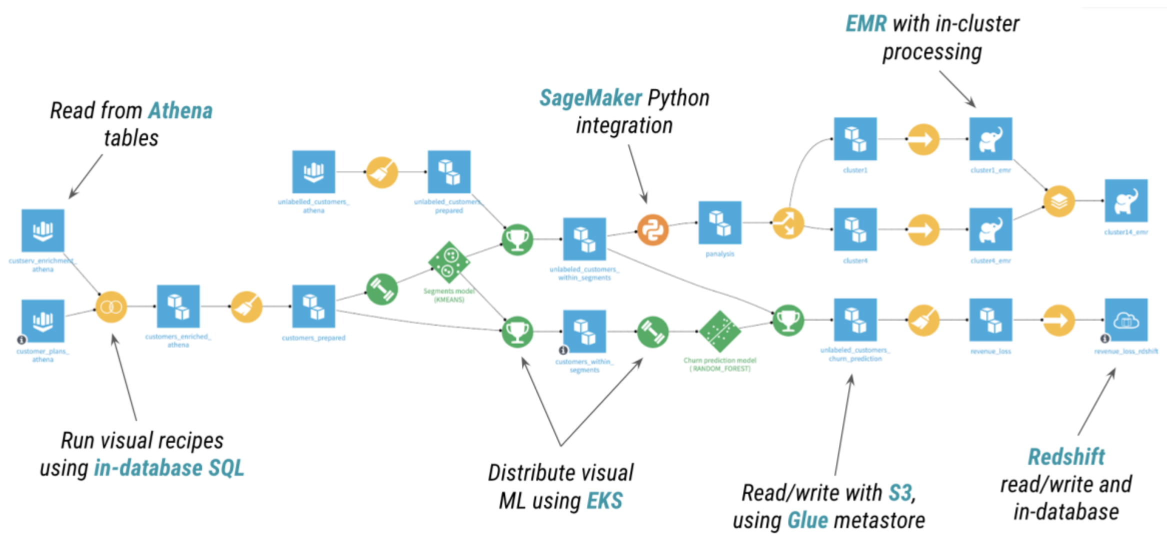

EMR with in-cluster processing
SageMaker Python integration
Read from Athena tables
Run visual recipes using in-database SQL
Distribute visual ML using EKS
Read/write with S3, using Glue metastore
Redshift read/write and in-database

Figure 12 – Machine learning flow within Dataiku DSS, enabling users to visually orchestrate the ML lifecycle in collaboration with multiple AWS technologies

Access, understand, and clean data

Data discovery

After defining your business goal, the first step in the machine learning lifecycle is to discover and access data. Dataiku DSS makes it easy to find and use data through its built-in catalog feature and AWS Glue integration. Metadata for Amazon S3 data sources, which are created as part of a Dataiku DSS project, is automatically registered in AWS Glue and indexed within the internal Dataiku DSS catalog. This makes finding the right data as simple as searching for a few keywords. Metadata tags let you can see which projects and use cases the data was previously associated with. Dataiku DSS supports a wide range of data sources, types, and formats. For a full list, see Connecting to data.

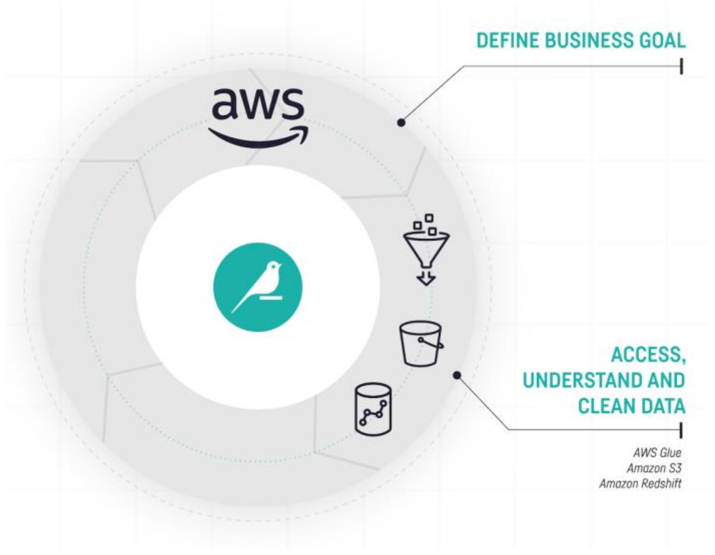
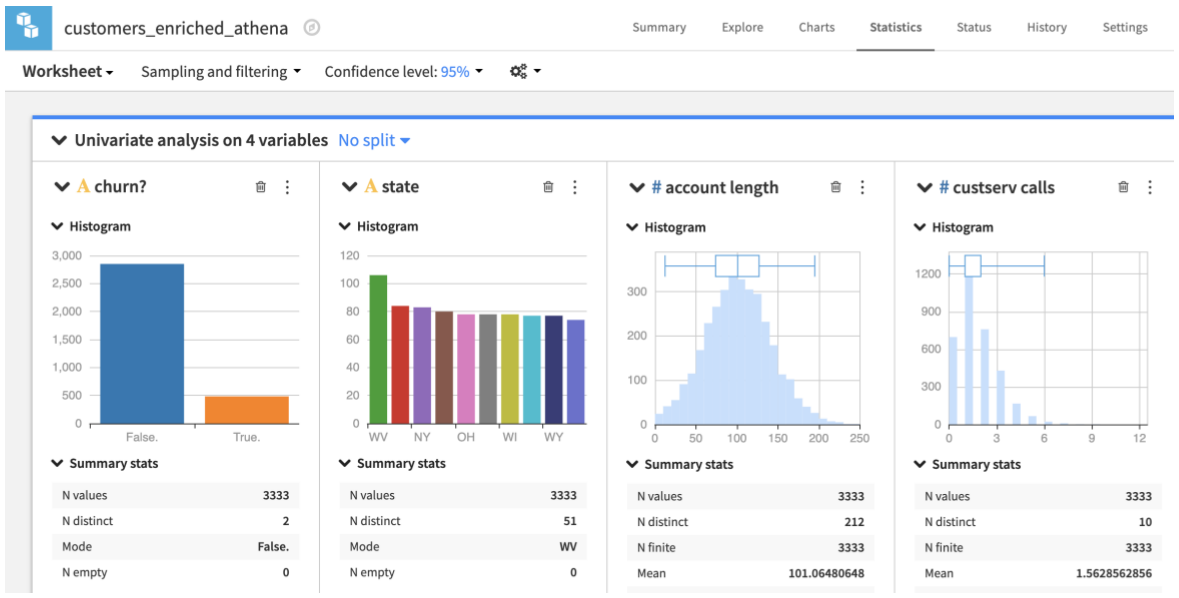


Figure 13 – Access, understand, and clean data, using Dataiku DSS in collaboration with AWS services.

Understanding and preparing data

Most of the effort in a machine learning project is spent preparing the data. So, building a coherent, reliable, and explainable data pipeline is perhaps the most important part of the machine learning lifecycle. Within Dataiku DSS, visual data preparation features allow you to easily create data cleansing, normalization, and feature engineering scripts in an interactive way. You can create these scripts directly in the project flow using *recipes* (either visual or code-based data transformation steps). Or you can experiment using a *visual analysis* that can be deployed back to the project flow as a *visual recipe,* once it is complete. Summary statistics and charts provide additional tools for understanding and explaining the data.



churn?

N values	3333
N distinct	2
Mode	False.
N empty	0

state

N values	3333
N distinct	51
Mode	WV
N empty	0

account length

N values	3333
N distinct	212
N finite	3333
Mean	101.06480648

custserv calls

N values	3333
N distinct	10
N finite	3333
Mean	1.5628562856

Figure 14 – Univariate analysis on Amazon S3 data variables

You can combine visual recipes and code recipes within the same flow to support real-time collaboration between different users working on the same project. For example, you might have a data engineer using an *SQL recipe*, a data scientist using an *R recipe*, and a business analyst using a *visual recipe*, all simultaneously contributing to different part of the data pipeline. Integration with external code repositories and IDEs, such as PyCharm, Sublime and R Studio, as well as built-in Jupyter notebooks, enables coders and developers to use the processes and tools they're familiar with.

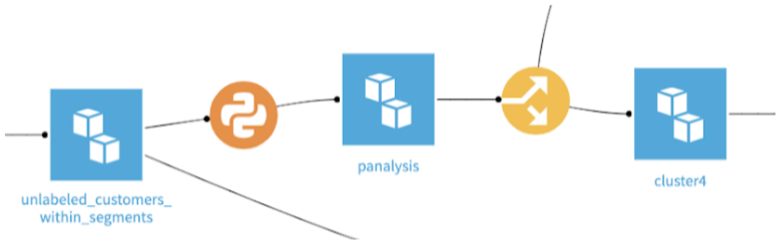


Figure 15 – Python code recipe in conjunction with a visual split recipe using Amazon S3 data

Data preparation for unstructured data, such as images, can be particularly challenging with regards to supervised learning, and often requires manually labeling the data before it can be used. Dataiku DSS includes an ML-assisted labeling tool that can help optimize labeling of images, sound, and tabular data through active learning.

Build machine learning

Model training

Once the data is cleansed, transformed, and enriched, it can then be used as the input to train a model. As part of the machine learning lifecycle, Dataiku DSS provides many built-in features for iteratively creating and training machine learning and deep learning models, including Visual ML, AutoML, code-based custom models, and SageMaker integration.

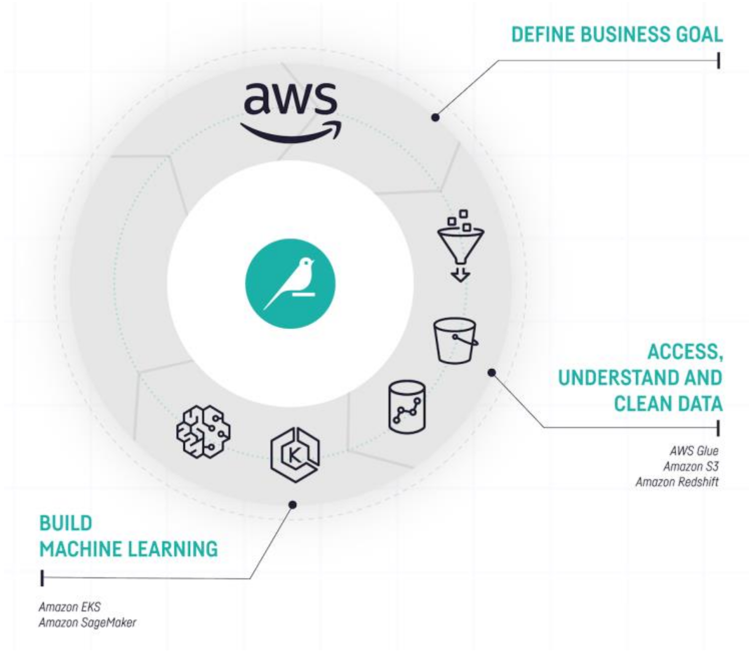
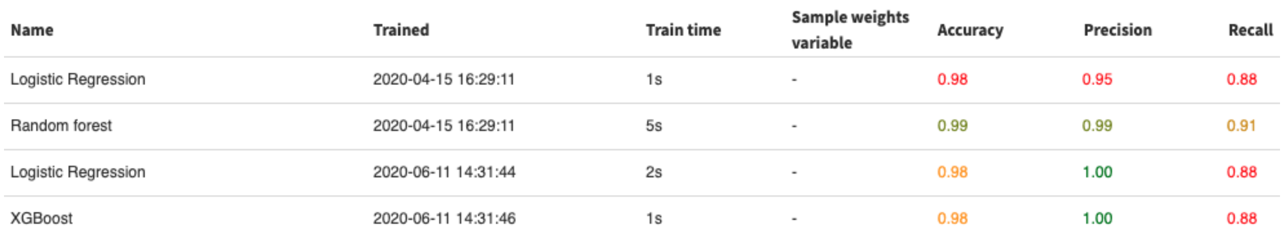


Figure 16 – Build machine learning, using Dataiku DSS in collaboration with AWS services

Business analysts and other non-technical users can use Visual and AutoML features to enhance project collaboration. Data scientists can also use these features to rapidly prototype simple models. Machine learning and deep learning models can be trained using various open source libraries, including Python (scikit-learn and XGBoost), Spark (MLLib), and Keras and TensorFlow. As a clear-box solution based on open-source algorithms, Dataiku DSS allows users to easily tune parameters and hyperparameters in order to optimize models trained using the Visual and AutoML features.

Experiment tracking

Because model training is an iterative process, Dataiku DSS keeps track of all previous experiments, including model versions, feature handling options, the algorithms selected, and the parameters and hyperparameters associated with them. Through the use of either the built-in Git repository or integration with an external repository, Dataiku DSS also manages code and project versions. This means that you can revert to a previous version of an experiment, model, or whole project, at the click of a button.



Name	Trained	Train time	Sample weights variable	Accuracy	Precision	Recall
Logistic Regression	2020-04-15 16:29:11	1s	-	0.98	0.95	0.88
Random forest	2020-04-15 16:29:11	5s	-	0.99	0.99	0.91
Logistic Regression	2020-06-11 14:31:44	2s	-	0.98	1.00	0.88
XGBoost	2020-06-11 14:31:46	1s	-	0.98	1.00	0.88

Figure 17 – Experiment tracking using Dataiku DSS

Validation

Before deploying a new model into production, or updating an existing one, it's important to validate that both the data and model are consistent, performant, and working as expected. Dataiku DSS provides built-in *metrics and checks* to validate data, such as column statistics, row counts, file sizes, and the ability to create custom metrics. An *Evaluate Recipe* produces metrics (such as Precision, Recall, and F1) that can be used to test the performance of a trained model on unseen data.

Scalable infrastructure

By providing access to a self-service, scalable, and performant infrastructure, cloud services can afford many advantages, when used as part of the machine learning lifecycle. Dataiku DSS can create and manage dynamic clusters for both Amazon Elastic Kubernetes Service and Elastic MapReduce. For example, the model training process can be transparently offloaded to automatic scaling Amazon Elastic Kubernetes Service (Amazon EKS) clusters (CPU-based or GPU-based) or SageMaker.

Amazon EKS can be automatically used in conjunction with Python, R and, Spark jobs for model training and data preprocessing tasks. Dataiku DSS will create the required container images based on your specific project requirements and push those to your container repository, to be used by the EKS cluster. For more information, see Reference architecture: managed compute on EKS with Glue and Athena.

Deploy machine learning

Model deployment

Regardless of how we've trained our model—AutoML, Visual ML, custom code, etc.—the final step in the machine learning lifecycle is to deploy it to production. Within Dataiku DSS, this could involve deploying a single model, multiple complementary or competing models, partitioned models, or the model in conjunction with all of the data preprocessing steps.

Dataiku DSS provides infrastructure options to support these different deployment scenarios:

- *Automation Node*, for scheduling batch inference, including data preprocessing
- *API Deployer* and *API Nodes*, for deploying individual models as REST API services

Whichever deployment method is used, Dataiku DSS keeps track of the artifact versions to allow rollback.

The *API Deployer* manages the underlying infrastructure used for the *API Nodes*, whether they are physical servers, virtual machines (for example, EC2 instances), or containers managed by Kubernetes (for example, Amazon EKS). Dataiku DSS also supports exporting models to portable formats, such as PMML, ONNX, and Jar, so they can be deployed using alternative infrastructure options.

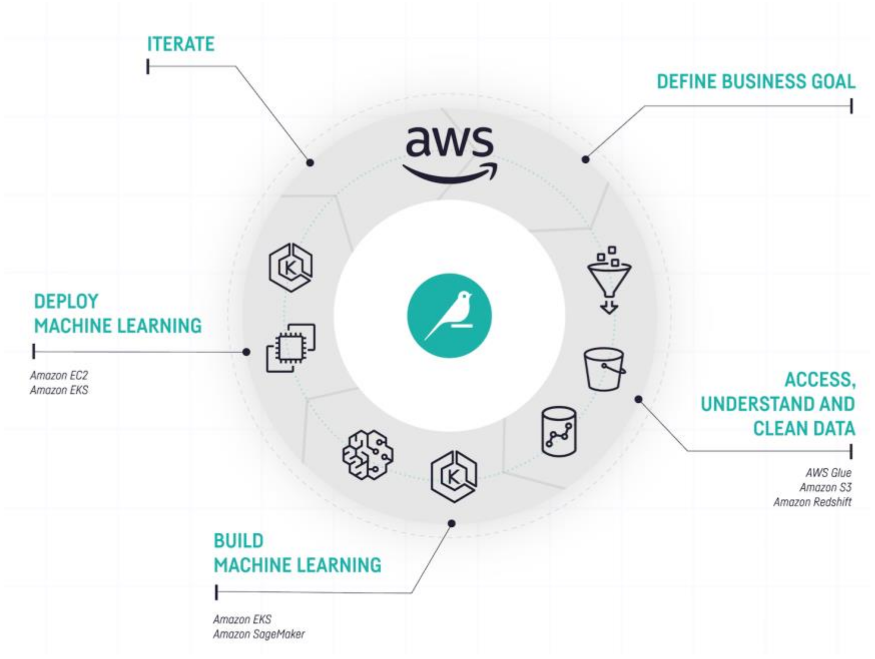


Figure 18 – Deploy machine learning using Dataiku DSS in collaboration with AWS services

Orchestration

Using a combination of the Automation Node, Scenarios, and the Public API, alongside native Git integration (GitHub, Bitbucket, etc.), and common CI/CD & configuration management techniques and tools, Dataiku DSS makes it simple to orchestrate the deployment of machine learning applications and the associated infrastructure and data preprocessing pipelines. For example, Dataiku DSS can dynamically create and manage an Amazon EKS cluster onto which it will deploy your real-time machine learning REST API services, to automatically retrain and redeploy a model based on key performance metrics, or to rebuild an entire machine learning application.

Model monitoring

Advanced model monitoring capabilities such as *model drift detection*, *metrics and checks,* and the *Evaluate Recipe* can be used to assess your model's key performance indicators against real data. Using visual recipes in Dataiku DSS, you can easily create validation feedback loops to compute the true performance of a saved model against a new validation dataset with the option to automatically retrain and redeploy models.

The feedback mechanism in Dataiku DSS delivers the ability to determine when existing models can merely need to be retrained, or whether a challenger model, with better performance, should be used to replace the existing production-deployed model instead. Alternatively, you can run A/B tests on your machine learning models to arrive at the best performing model.

Further, dedicated dashboards for monitoring global data pipelines and model performance in Dataiku allow for greater transparency and easier collaboration on MLOps.

Dataiku DSS: Your journey ahead

The Dataiku DSS image on the AWS Marketplace provides an easy way to start testing out these capabilities today. You can use a 14-day trial of the Enterprise Edition, including access to Amazon S3, Amazon Redshift, Amazon EMR, Spark on Kubernetes with Amazon EKS, and much more.

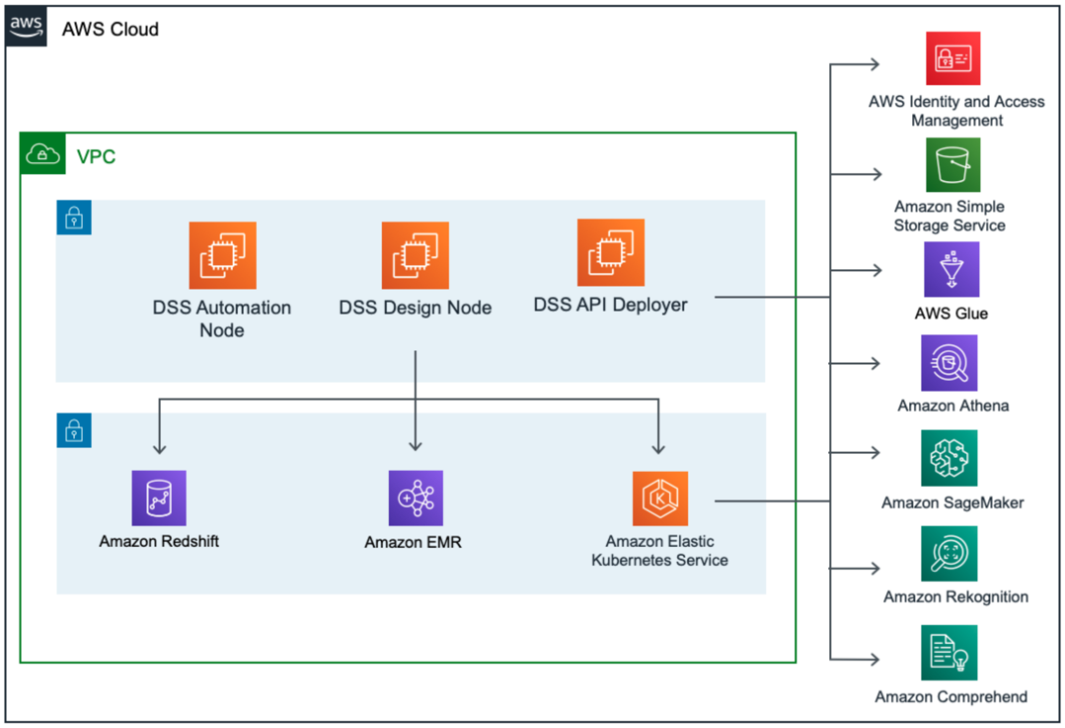


Figure 19 – Example architecture of Dataiku DSS components deployed in collaboration with AWS services to support MLOps.

Domino Data Lab

by Josh Poduska, Chief Data Scientist, Domino Data Lab

Introduction

Domino Data Lab was founded in 2013 to accelerate the work of code-first data scientists and help organizations run core components of their business on the models they develop. It pioneered the Data Science Platforms category, and today powers data science research at over 20% of Fortune 100 companies.

Domino brings order to the chaos of enterprise data science through:

- **Instant access to compute** – IT-approved, self-service, elastic
- **Data, code, environment, and model management** – automatically versioned, searchable, shareable, and always accessible
- **Openness** – open source and proprietary IDEs and analytical software, all containerized under one platform and deployed on premises or in the cloud
- **Full reproducibility of research –** central knowledge management framework, with all units of work tied together
- **Collaboration** – for teams and for enterprises, with access to comprehensive search capabilities to share all key assets and collaborate on projects
- **Easy deployment** – models, web apps, other data products
- **Platform and project management** – best-in-class governance and security for IT and data science leaders

Domino does not provide any proprietary machine learning algorithms or IDEs, and does not push any single type of distributed compute framework. Instead, Domino is focused on providing an open platform for data science, where large data science teams have access to the tools, languages, and packages that they want to use, plus scalable access to compute, in one central place. Domino's code-driven and notebook-centric environment, combined with AWS's expertise in large-scale enterprise model operations makes for a natural technical pairing when optimizing the productionization of ML.

Enterprise data science workflows

1. Domino architecture

The Domino platform begins with containerization of ML workloads. Domino was the first data science platform to run all its workloads on Docker, and can now run natively on Kubernetes to future-proof its customers' ability to use the next generation of tools and distributed compute frameworks.

Domino can be deployed in one of three ways:

- Customer hosted and managed, in which you bring your own Kubernetes cluster
- Customer hosted and Domino managed, in which you give Domino a dedicated VPC in your AWS account for a Kubernetes deployment
- Domino hosted and managed on a certified reference Kubernetes architecture

Domino is available on the AWS Marketplace.

When running Domino on Amazon Elastic Kubernetes Service (EKS), the architecture uses AWS resources to fulfill the Domino cluster requirements as follows:

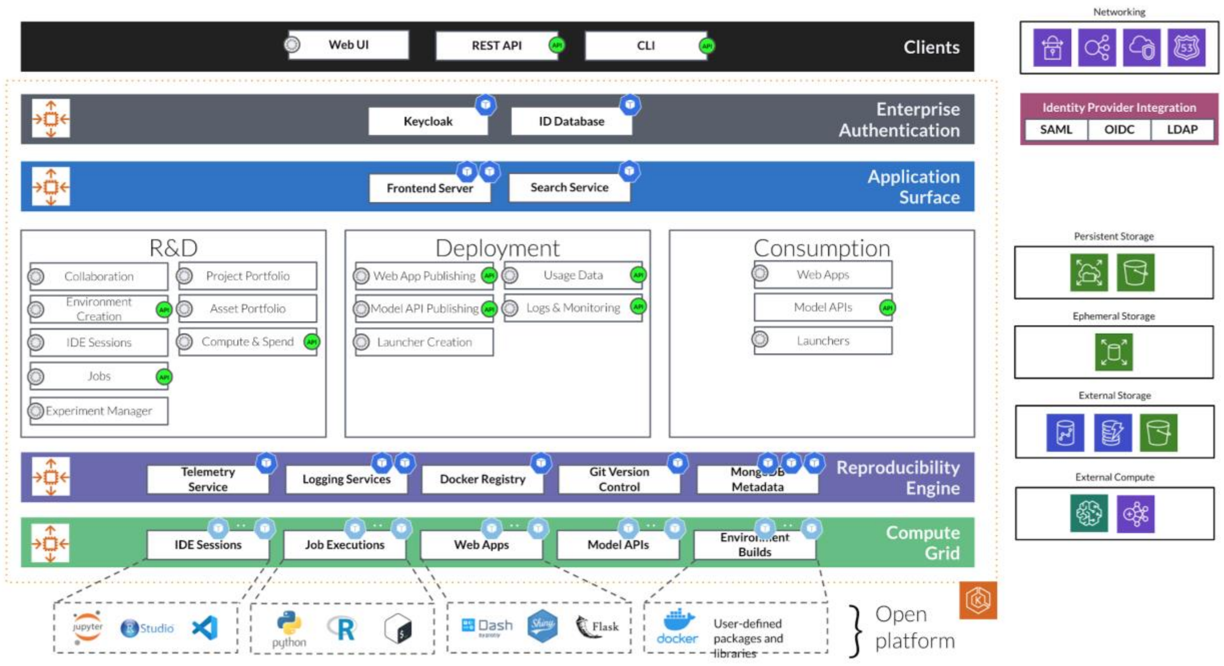
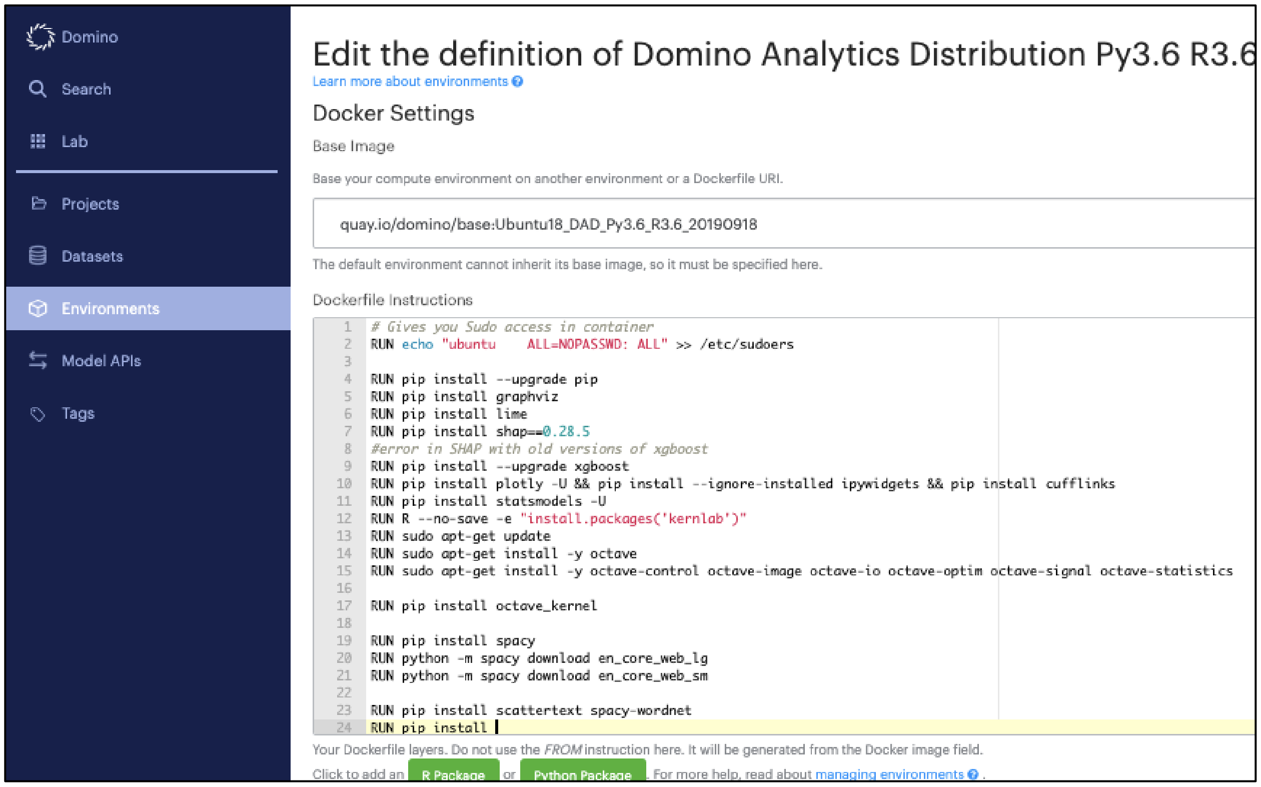


Figure 20 – Domino on Amazon EKS

Core services for reproducibility, application services, and enterprise authentication run in persistent pods. User execution and production workloads are brokered to the compute grid in ephemeral pods. Execution resources scale automatically using the Kubernetes Cluster Autoscaler and Amazon EC2 Auto Scaling groups.

2. Environment management

Domino allows admins to manage Docker images and control their visibility across the organization, but image modification is not limited to platform administrators. All Domino users can self-serve environment management to create and modify Docker environments for their own use. Once created, these pre-configured environments are available to data scientists throughout the enterprise, based on configurable permission levels.



Edit the definition of Domino Analytics Distribution Py3.6 R3.6

Learn more about environments

Docker Settings

Base Image

Base your compute environment on another environment or a Dockerfile URI.

quay.io/domino/base:Ubuntu18_DAD_Py3.6_R3.6_20190918

The default environment cannot inherit its base image, so it must be specified here.

Dockerfile Instructions

```
# Gives you Sudo access in container
RUN echo "ubuntu    ALL=NOPASSWD: ALL" >> /etc/sudoers

RUN pip install --upgrade pip
RUN pip install graphviz
RUN pip install lime
RUN pip install shap==0.28.5
#error in SHAP with old versions of xgboost
RUN pip install --upgrade xgboost
RUN pip install plotly -U && pip install --ignore-installed ipywidgets && pip install cufflinks
RUN pip install statsmodels -U
RUN R --no-save -e "install.packages('kernlab')"
RUN sudo apt-get update
RUN sudo apt-get install -y octave
RUN sudo apt-get install -y octave-control octave-image octave-io octave-optim octave-signal octave-statistics

RUN pip install octave_kernel

RUN pip install spacy
RUN python -m spacy download en_core_web_lg
RUN python -m spacy download en_core_web_sm

RUN pip install scattertext spacy-wordnet
RUN pip install
```

Your Dockerfile layers. Do not use the *FROM* instruction here. It will be generated from the Docker image field.

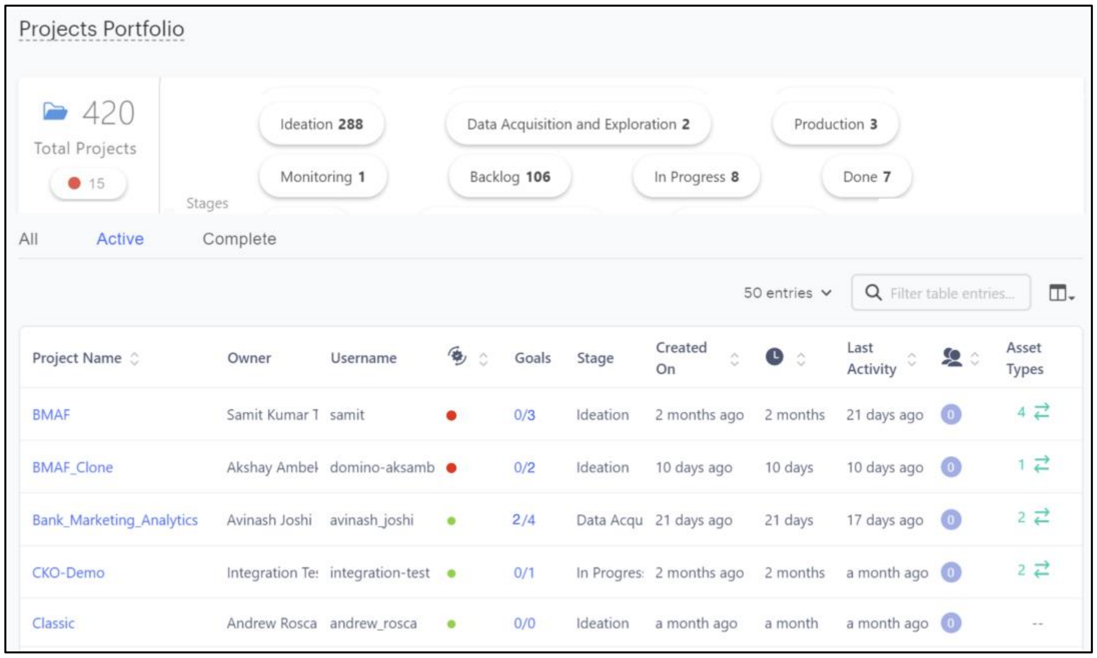
Click to add an R Package or Python Package. For more help, read about managing environments.

Figure 21 – Users can easily duplicate and modify an environment for their own use

3. Project management

Key, and often overlooked, aspects of ML productionization occur during the initial stages of a project. Leveraging prior work on similar projects is one such aspect. Because Domino automatically tracks all practitioner work, it offers a repository of past projects, code, experiments, and models from research scattered across the organization. Data scientists search this repository for prior art as they kick off collaborative project work.

Additionally, project parameters need to be defined for successful work in collaborative organizations. Teams and stakeholder roles must be set. Goals and milestones must be defined and then documented with reproducible artifacts once completed. Lifecycle stages should be chosen and tracked as a project progresses. Domino helps enforce these best practices and allows data science leaders to track project details in the [Projects Portfolio.] Everyone involved in the productionization of ML has visibility into the full set of research going on inside their organization.


Projects Portfolio

420 Total Projects ● 15

Stages: Ideation 288 · Data Acquisition and Exploration 2 · Production 3 · Monitoring 1 · Backlog 106 · In Progress 8 · Done 7

All · Active · Complete

50 entries · Filter table entries...

Project Name	Owner	Username		Goals	Stage	Created On		Last Activity		Asset Types
BMAF	Samit Kumar T	samit	🔴	0/3	Ideation	2 months ago	2 months	21 days ago	0	4 ⇄
BMAF_Clone	Akshay Ambel	domino-aksamb	🔴	0/2	Ideation	10 days ago	10 days	10 days ago	0	1 ⇄
Bank_Marketing_Analytics	Avinash Joshi	avinash_joshi	🟢	2/4	Data Acqu	21 days ago	21 days	17 days ago	0	2 ⇄
CKO-Demo	Integration Te:	integration-test	🟢	0/1	In Progres:	2 months ago	2 months	a month ago	0	2 ⇄
Classic	Andrew Rosca	andrew_rosca	🟢	0/0	Ideation	a month ago	a month	a month ago	0	--

Figure 22 – Projects Portfolio details all research and the status toward established goals

4. Data connections

With environment management, reproducibility, and project best practices set, collaborative data science research and productionization can proceed at a rapid pace. This begins with connecting to, iterating on, and managing data.

Data sources

Domino users can bring the data and files stored outside of Domino to their work inside of Domino. Domino is easily configured to connect to a wide variety of data sources, including Amazon S3, Amazon Redshift, and an Amazon EMR cluster. This involves loading the required client software and drivers for the external service into a Domino environment (many come pre-loaded), and loading any credentials or connection details into Domino environment variables.

For example, connecting to Amazon Redshift is usually done by creating a connection and retrieving the results of a query using Python or R. Because the Amazon Redshift database is likely secured with a user name and password, credentials can be stored as environment variables in the Domino project so they can be accessed at runtime without needing to include them explicitly in code. Popular libraries like psycopg2 and RPostgreSQL are used to manage the connection.

You can optionally configure AWS credential propagation, which allows for Domino to automatically assume temporary credentials for AWS roles that are based on roles assigned to users in the upstream identity provider. Once a user has logged into the system, their short-lived access token is associated with any workloads executed during their session.

Domino-managed files and data

Saved data science work in Domino is automatically versioned, accessible to collaborators, and fully reproducible. In a centralized repository of sorts, the storage of these project files and data is backed by a durable object storage system, such as Amazon S3 for AWS deployments. With interactive sessions involving notebooks or batch jobs, these files and data are automatically synchronized to the runtime containers when users want to do active work. By default, any changes to files are automatically versioned back to the centralized repository. Domino’s versioning system provides rich file difference information between revisions made inside and outside of experiments.

Sometimes a project requires or accumulates very large datasets, either in terms of hundreds of data files or very large-sized files. Typical examples include images for computer vision applications and event-based data that are accumulated regularly, such as financial time series data. For these use cases, Domino recommends storing the data in its purpose-built Domino Datasets, which is effectively filesystem storage designed for quick access to large file-based datasets. In AWS, this is backed by the scalable Amazon Elastic File System. At runtime, this Domino-managed data is mounted to execution containers exposing it as just another directory to data science practitioners.

5. Computing frameworks

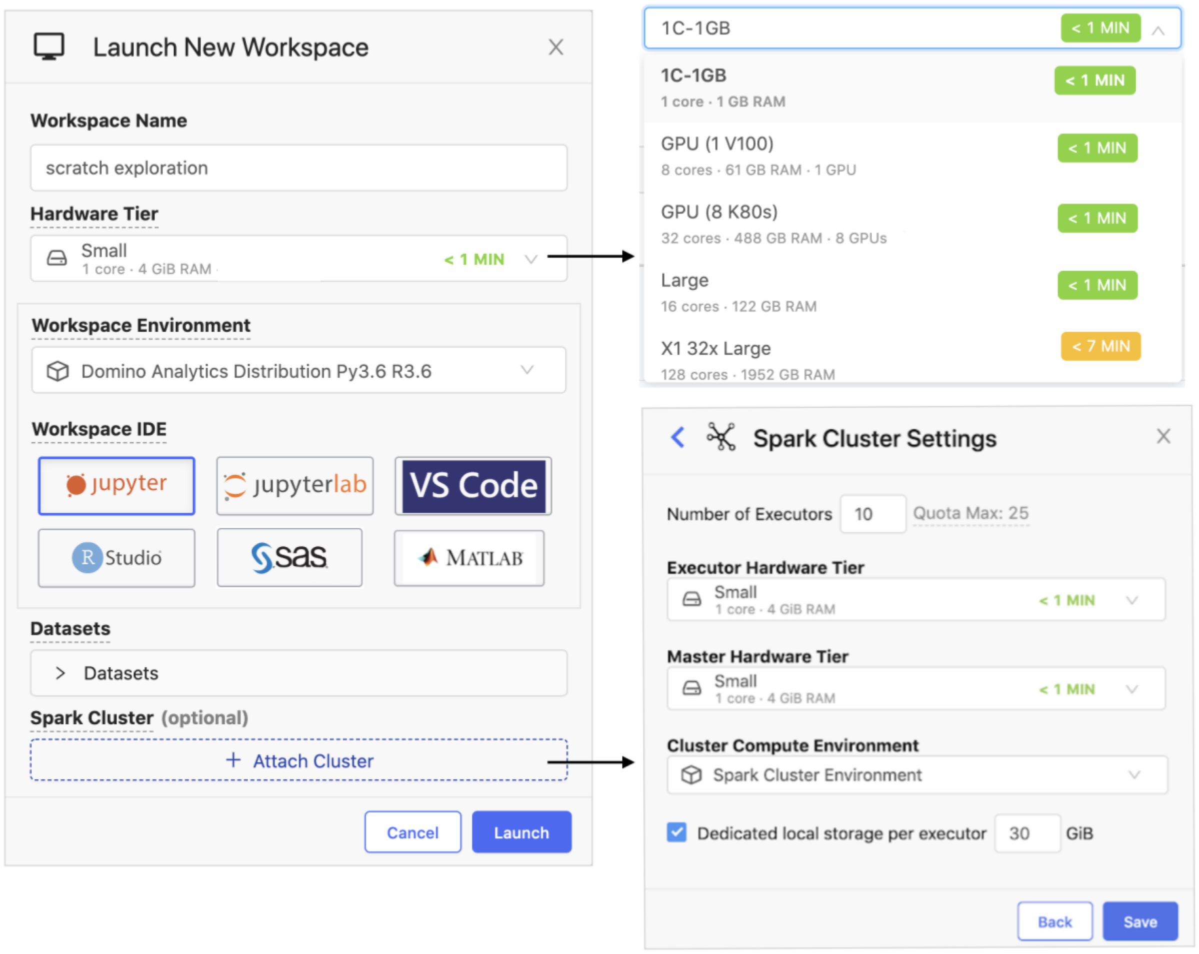
When it's time to process data, practitioners have the freedom to select the compute and environment that best meets their needs. Users easily choose from Domino-managed Docker images preconfigured to work on AWS GPU and CPU tiers. Included in this are the latest in NVIDIA configurations specific to data science workloads.

These environments can be configured to run a variety of computing frameworks, including Spark, Ray, Dask, SAS Viya, and the next big unknown breakthrough, effectively future-proofing the Domino platform as the single location to manage all data science work.

Domino's Spark integration includes both Domino-managed ephemeral Spark clusters, called On-Demand Spark, and connections to persistent clusters that could be managed by Amazon EMR. On-Demand Spark supports fully containerized execution of Spark workloads on the Domino Kubernetes cluster. Users can interact with Spark interactively through a Domino workspace or in batch mode through a Domino job, as well as directly with spark-submit. Domino will ensure that the packages and dependencies are properly distributed across the cluster, removing a common source of DevOps headaches and wasted time. Code and the results associated with Spark jobs are fully tracked and reproducible in Domino, just as any other type of work is.

With compute and environments defined, data scientists have the freedom to do their coding in open or proprietary IDEs, such as Jupyter Lab, RStudio, SAS, MATLAB, Superset, VS Code, and more.

These IDEs are containerized on AWS to create a powerful workflow for organizations with heterogeneous coding teams.



Launch New Workspace

Workspace Name: scratch exploration

Hardware Tier: Small, 1 core · 4 GiB RAM, < 1 MIN

Workspace Environment: Domino Analytics Distribution Py3.6 R3.6

Workspace IDE: jupyter, jupyterlab, VS Code, RStudio, sas, MATLAB

Datasets: Datasets

Spark Cluster (optional): + Attach Cluster

Cancel | Launch

1C-1GB 1 core · 1 GB RAM	< 1 MIN
GPU (1 V100) 8 cores · 61 GB RAM · 1 GPU	< 1 MIN
GPU (8 K80s) 32 cores · 488 GB RAM · 8 GPUs	< 1 MIN
Large 16 cores · 122 GB RAM	< 1 MIN
X1 32x Large 128 cores · 1952 GB RAM	< 7 MIN

Spark Cluster Settings

Number of Executors: 10 Quota Max: 25

Executor Hardware Tier: Small, 1 core · 4 GiB RAM, < 1 MIN

Master Hardware Tier: Small, 1 core · 4 GiB RAM, < 1 MIN

Cluster Compute Environment: Spark Cluster Environment

✓ Dedicated local storage per executor: 30 GiB

Back | Save

Figure 23 – Domino offers a self-service selection of environments, compute, and IDEs

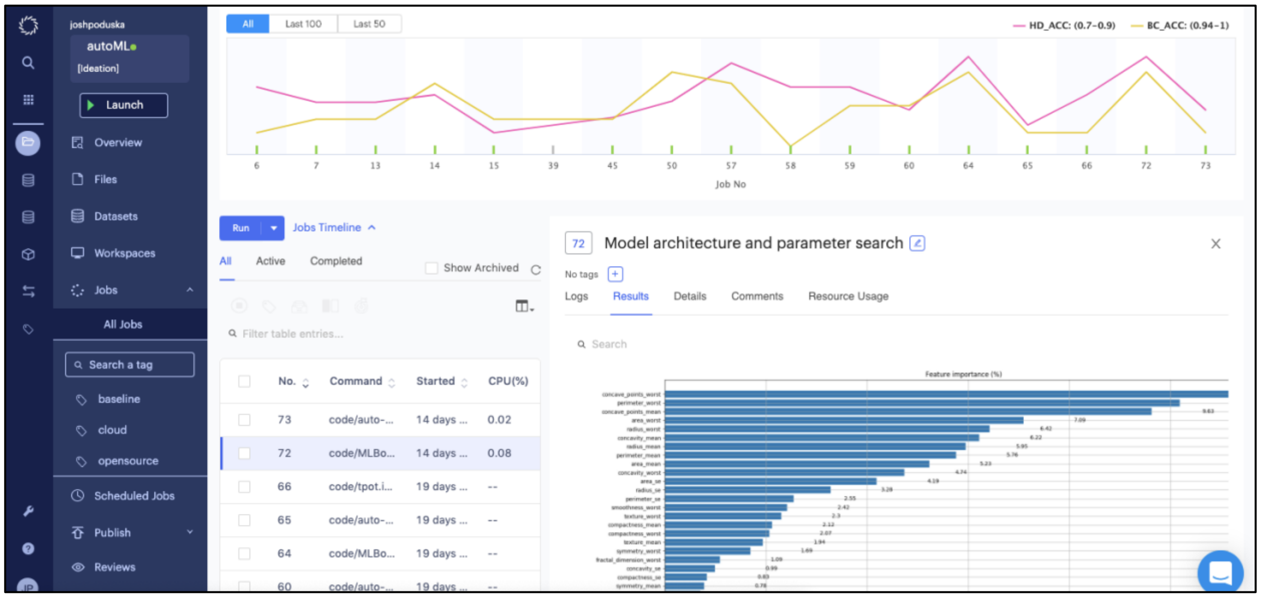
6. Model building

Data scientists write code to build models in Domino using their favorite statistical computing software and IDE as outlined in the previous section. Code is written in an interactive workspace and then executed in that same workspace, in a batch job, or in a scheduled job. Typically, model training (and all the preparation that leads up to it) is done via open-source packages in R and Python. SAS and MATLAB are also popular choices for model building in Domino. Domino's API allows for the integration of batch jobs into production pipelines.

Both traditional ML and deep learning workflows are supported via Domino's preconfigured environments for GPUs and deep learning frameworks, such as TensorFlow and PyTorch. Given the ease of building your own environment in Domino, new frameworks can be quickly integrated.

As would be expected with any Docker-based interactive workspace, Domino users can also leverage Amazon SageMaker tools to build models. Users simply make calls to tools such as Amazon SageMaker Autopilot or training jobs from a Domino workspace. This can be done from any IDE that supports a language with extensions for SageMaker. A common workflow is to kick off SageMaker calls via cells in a Jupyter notebook in a Domino interactive workspace. Training then happens in SageMaker exactly as it would when working directly with SageMaker.

Domino's central and open platform allows data scientists to seamlessly switch between tools while Domino keeps a constant track of the flow of research, down to the smallest details.



No.	Command	Started	CPU(%)
73	code/auto-...	14 days ...	0.02
72	code/MLBo...	14 days ...	0.08
66	code/tpot.i...	19 days ...	--
65	code/auto-...	19 days ...	--
64	code/MLBo...	19 days ...	--
60	code/auto-...	19 days ...	--

Figure 24 – The Experiment Tracker is one of Domino's tools for central tracking, collaboration, and reproducibility of research

7. Productionizing models

Deploying in Domino

Data scientists can use Domino's simplified deployment process to deploy and host models in Domino on AWS infrastructure, which provides an easy, self-service method for API endpoint deployment. When deploying in Domino, Domino manages the containers hosted on AWS. Endpoints are not automatically registered with AWS. For more details, see Video introduction to model publishing.

Deploying models with Domino's deployment process has several advantages. First, the publishing process is simple and straightforward, and designed for data scientists to easily manage the process themselves. Second, Domino provides full model lineage down to the exact version of all software used to create the function that calls the model. Third, Domino can provide an overview of all assets and link those to individuals, teams, and projects.

Deploying outside Domino

Sometimes you may want to build models in Domino but host them in an environment outside Domino. This could be because you have already made investments in a production environment that supports very high scale or low latency, the production data may not be cleared for export outside of a particular environment, or you may want to control deployment using a custom CI/CD pipeline.

Domino allows you to export model images built in Domino to an external container registry. These images include all the information needed to run the model, such as model code, artifacts, environment, and project files. Domino exposes REST APIs to programmatically build and export the model image, which can be called by your CI/CD pipelines or integrated with workflow schedulers, such as Apache Airflow. By default, the images are built in Model API format. These images can be easily deployed in an environment that can run Docker containers.

You can also leverage Domino's SageMaker Export feature. This API-based feature builds a Docker image for a given version of a Model API in an AWS SageMaker-compliant format and then exports it to Amazon ECR or any third-party container registry outside Domino. As part of the API request, users need to provide credentials for their registry to push an image to it. These credentials are not saved inside Domino and can have a Time to live (TTL) attached.

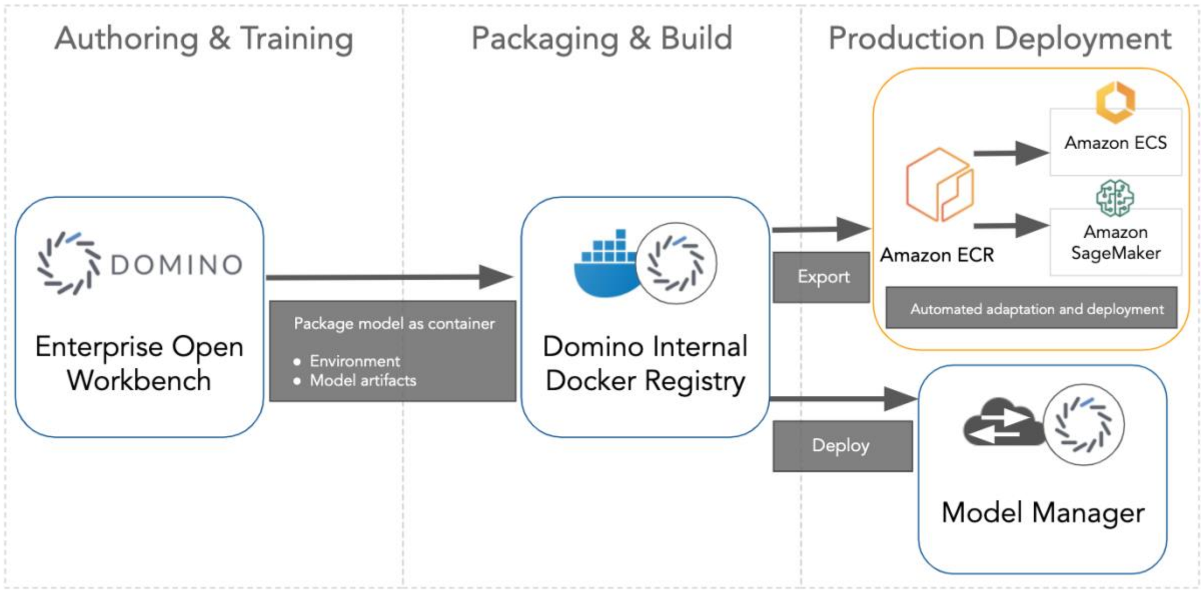
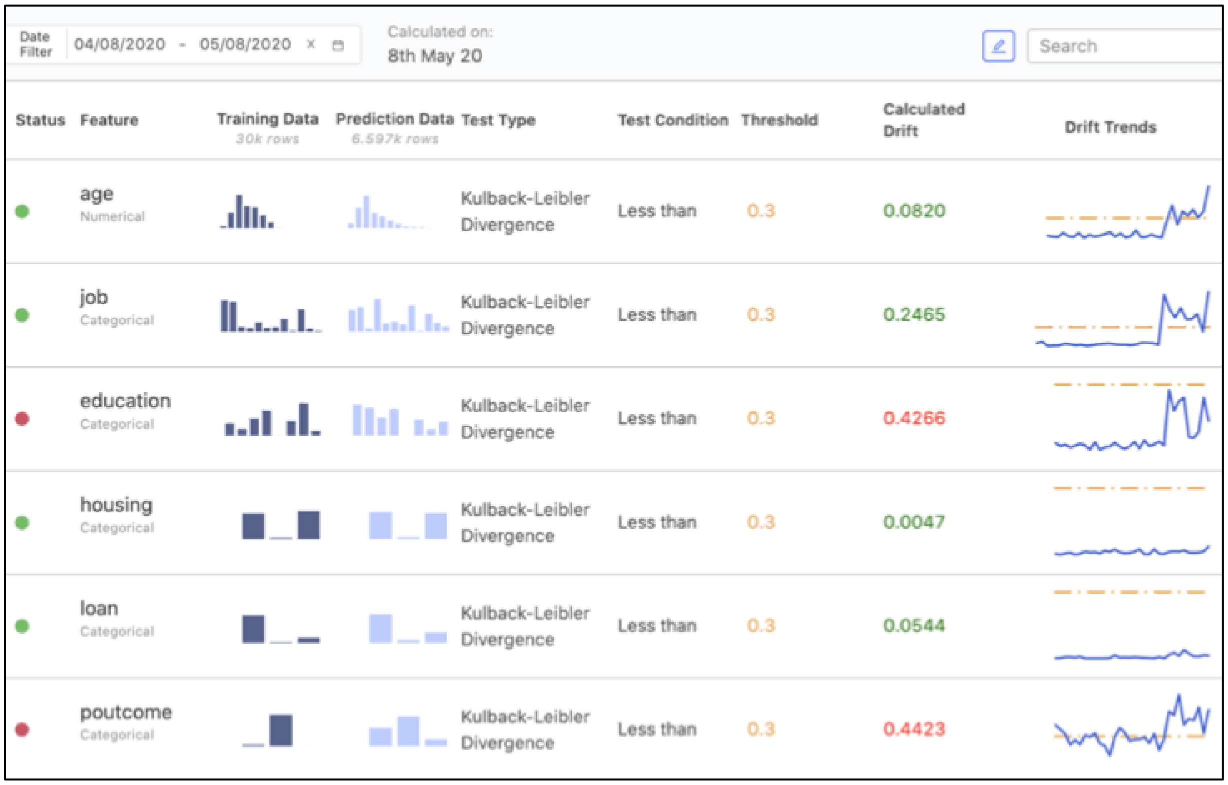


Figure 25 – Models can be deployed in Domino or exported to external environments

8. Model monitoring

After models are deployed, it's critical that they are monitored to ensure that the data that's presented for scoring is aligned with the data that was used to train the model. Proactive model monitoring allows you to detect economic changes, shifts in customer preferences, broken data pipelines, and any other factor that could cause your model to degrade.

Domino Model Monitor (DMM) automatically tracks the health of all deployed models, no matter if they were deployed in Domino, SageMaker, or another platform. DMM provides a single pane of glass for anomaly detection against input data drift, prediction drift, and ground truth-based accuracy drift. If thresholds are exceeded, retraining and redeployment of models can be initiated via integration with the Domino API.



Date Filter 04/08/2020 - 05/08/2020 Calculated on: 8th May 20 Search

Status	Feature	Training Data 30k rows	Prediction Data 6.597k rows	Test Type	Test Condition	Threshold	Calculated Drift	Drift Trends
●	age Numerical			Kulback-Leibler Divergence	Less than	0.3	0.0820	
●	job Categorical			Kulback-Leibler Divergence	Less than	0.3	0.2465	
●	education Categorical			Kulback-Leibler Divergence	Less than	0.3	0.4266	
●	housing Categorical			Kulback-Leibler Divergence	Less than	0.3	0.0047	
●	loan Categorical			Kulback-Leibler Divergence	Less than	0.3	0.0544	
●	poutcome Categorical			Kulback-Leibler Divergence	Less than	0.3	0.4423	

Figure 26 – Domino Model Monitor tracks the health of all models in production

Domino: Your journey ahead

Domino manages the complicated data science research and development (R&D) process in a way that amplifies the effectiveness of large and scattered teams. When combined with the scalable infrastructure and deployment expertise of AWS, the two technologies allow data science organizations to better manage the productionization of ML models with great success.

Download a free trial today so you can see how Domino works with AWS to provide an open data science platform in the cloud. Full details on how you can purchase Domino, including pricing information and user reviews, can be found on the Domino page in the AWS Marketplace.

KNIME

by Jim Falgout, VP of Operations, KNIME

KNIME Software: creating and productionizing data science

KNIME is software used to create and productionize data science in one easy and intuitive environment. This enables every stakeholder in the data science process to focus on what they do best. KNIME supports the data science lifecycle from data discovery to production model deployment, all with a single workflow paradigm.

This section explains how one software environment enables individuals and organizations to tackle their data challenges.

Two tools to cover the entire data science lifecycle

KNIME software consists of two complementary, integrated products that interact for a complete data science solution.

KNIME Analytics Platform is where data engineers and data scientists get started with KNIME. Usually run on a desktop or laptop, KNIME provides a great way to get started with a no-code environment (although if users do want to code, that is possible). It's a visual development environment used to build workflows that perform various functions such as data discovery, data engineering, data preparation, feature engineering and parameter optimization, model learning, as well as model testing and validation. As an open platform, KNIME supports other open source technologies such as H2O, Keras, TensorFlow, R and Python. KNIME's open approach ensures that the latest, trending technologies are quickly usable on the platform.

KNIME Server complements KNIME Analytics Platform by offering a collaborative environment for data science teams to work productively. It provides automation through job scheduling and a REST API for executing workflows. KNIME Server extends the reach of data science to all members of an organization with all levels of technical expertise via the KNIME WebPortal. Workflows built in KNIME Analytics Platform are deployed as end user applications and made available as web-based applications, exposing users to the right amount of detail and complexity. KNIME Server contains workers, or what we call Executors, that do the work of executing workflows. Executors can be run in Amazon EC2 Auto Scaling groups on AWS, elastically scaling to support your computation needs.

With these two software platforms, KNIME Software covers the entire data science cycle as seen in Fig. 27. No additional add-ons or plugins needed.

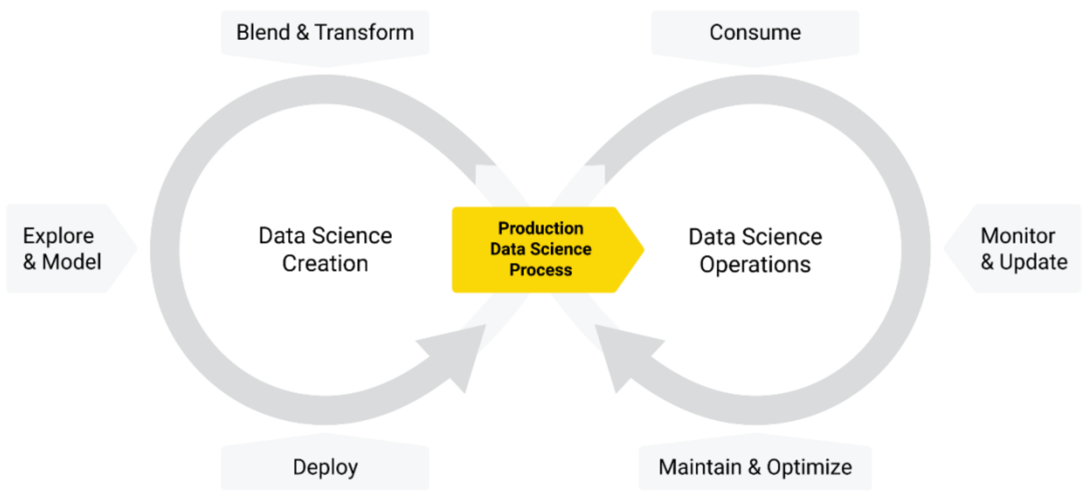


Figure 27 – End-to-End data science integrating the creation and operations processes

All the features and functionality needed in one environment.

KNIME is continuously developing new features and functionality. Whatever gets developed is, and will continue to be, designed to work seamlessly within the existing workflow paradigm.

Share and find solutions

The KNIME Hub is the place where the global community of KNIME users shares and finds solutions for their data science challenges and collaborates on KNIME workflows, nodes, and components.

Use KNIME on AWS

Both KNIME Analytics Platform and KNIME Server are offered in various forms in the AWS Marketplace. For more information about deploying KNIME Server on AWS, see KNIME Server on AWS Marketplace.

KNIME supports sourcing data from Amazon S3, Amazon Redshift, Amazon Relational Database Service, and Amazon Athena. These services are all integrated into KNIME as native nodes. With KNIME it's easy to mix and match data from AWS services with other data sources as needed. KNIME supports a broad range of services within AWS such as Amazon Translate, Amazon Comprehend, and Amazon Personalize. Additional integrations can be built within KNIME using Python and the Boto3 library. Wrapping that into a shareable component with KNIME makes it available for others to use. For example workflows, components, and nodes, see the KNIME Hub.

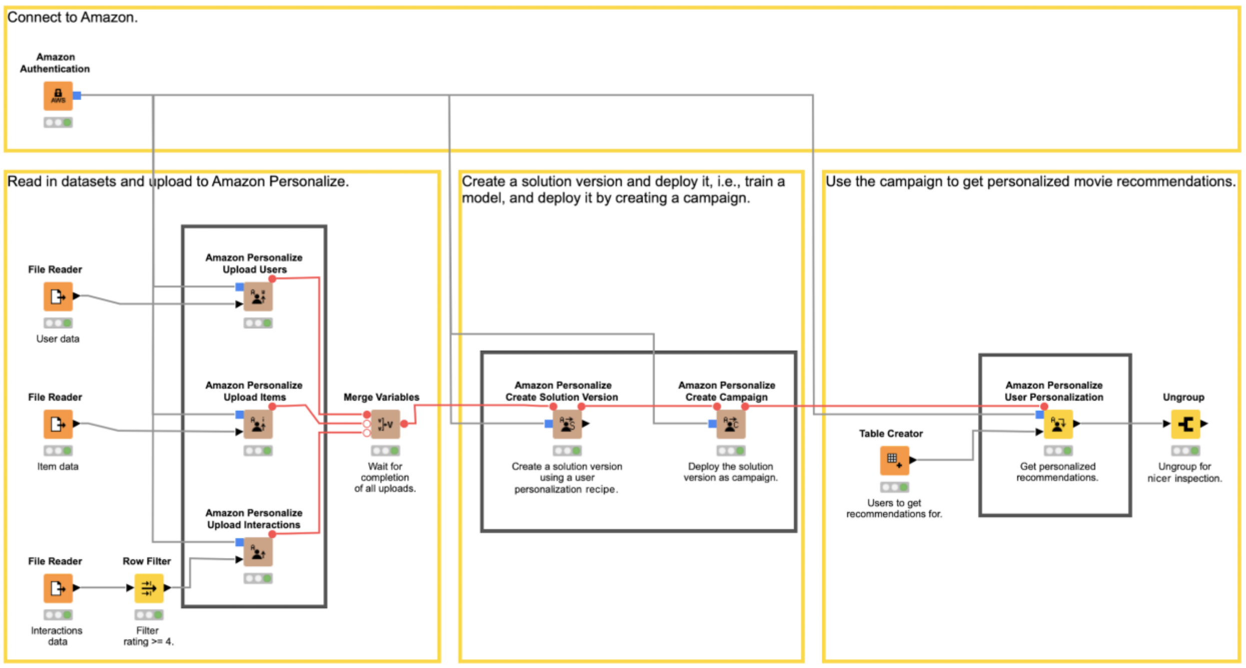

Connect to Amazon.
Amazon Authentication
Read in datasets and upload to Amazon Personalize.
File Reader
User data
File Reader
Item data
File Reader
Interactions data
Row Filter
Filter rating >= 4.
Amazon Personalize Upload Users
Amazon Personalize Upload Items
Amazon Personalize Upload Interactions
Merge Variables
Wait for completion of all uploads.
Create a solution version and deploy it, i.e., train a model, and deploy it by creating a campaign.
Amazon Personalize Create Solution Version
Create a solution version using a user personalization recipe.
Amazon Personalize Create Campaign
Deploy the solution version as campaign.
Use the campaign to get personalized movie recommendations.
Table Creator
Users to get recommendations for.
Amazon Personalize User Personalization
Get personalized recommendations.
Ungroup
Ungroup for nicer inspection.

Figure 28 – Creating and deploying an Amazon Personalize recommendation model using KNIME

The workflow in Figure 28 demonstrates how easy it is to use Amazon Personalize with KNIME. It shows the entire cycle of importing data into Amazon Personalize, creating a model, deploying a campaign, and getting real-time recommendations. It also takes advantage of KNIME's ability to prepare data for AWS Machine Learning services and post-process service results. Adding in KNIME Server to productionize integrated workflows completes the lifecycle.

Getting started with data science creation

KNIME Analytics Platform is the starting place for data scientists working in teams or on their own. KNIME provides nodes to connect to a broad set of data sources. Once connected, there are nodes to develop a deep understanding of your data, including sophisticated visualizations. The next step is data preparation (also called *data wrangling*). 80-90% of data science work is getting the data prepared and KNIME provides a wide variety of nodes to do this.

Experimenting during the early phases of a project is a strength of KNIME. The visual development interface enables quick prototyping of different approaches and supports collaboration with others. Combining ML algorithms from KNIME, Python, R, H2O, and deep learning technology such as Keras and TensorFlow is easy. Users can mix and match the technologies that make the most sense for the problem at hand. Or they can use the technologies that they have experience working with.

Workflows created using a visual development environment are well suited for collaboration. Understanding the logic behind a visual workflow is more easily accomplished than browsing a large set of code, which makes KNIME workflows well suited for a data science task that requires reproducible results.

KNIME Server supports working within a team. The workflow repository within KNIME Server supports deploying workflows from the KNIME Analytics Platform. Deployed workflows are versioned, and changes can be tracked over time. Changes to a workflow across versions can be visually inspected. Visual inspection of changes allows users to quickly see what changes were made to a workflow by team members over time.

Building and training models

There are many ways to build models in KNIME Analytics Platform. The workflow in Figure 29 is simplified but it shows a common pattern of sourcing data, preparing the data for modeling, training a model, and applying a test dataset to determine the viability of the model.

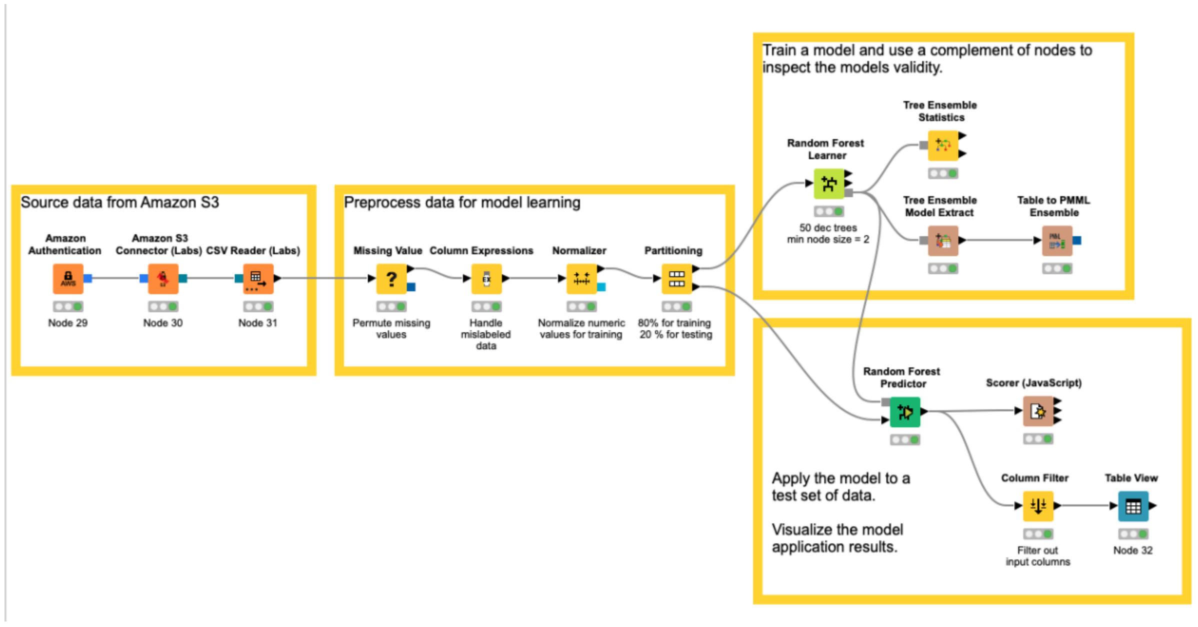

Source data from Amazon S3
Amazon Authentication
Amazon S3 Connector (Labs)
CSV Reader (Labs)
Node 29
Node 30
Node 31
Preprocess data for model learning
Missing Value
Column Expressions
Normalizer
Partitioning
Permute missing values
Handle mislabeled data
Normalize numeric values for training
80% for training 20 % for testing
Train a model and use a complement of nodes to inspect the models validity.
Random Forest Learner
50 dec trees min node size = 2
Tree Ensemble Statistics
Tree Ensemble Model Extract
Table to PMML Ensemble
Random Forest Predictor
Scorer (JavaScript)
Column Filter
Table View
Filter out input columns
Node 32
Apply the model to a test set of data.
Visualize the model application results.

Figure 29 – Using the KNIME Analytics Platform to train a random forest model

KNIME also supports more complex modeling techniques such as feature engineering, automated parameter optimization, model validation, and model interpretability.

Deploying models

Models are deployed within KNIME Server as workflows. The concept is that a model is not standalone but requires additional functionality, such as data pre-processing and post-processing. Wrapping a model in a KNIME workflow supports all those needs. Models wrapped in a KNIME workflow can be deployed to the KNIME Server as an API endpoint, a scheduled job, or available for batch processing.

The process starts with creating a workflow to generate an optimal model. The Integrated Deployment nodes allow data scientists to capture the parts of the workflow needed for running in a production environment, plus data creation and preparation and the model itself. These captured subsets are saved automatically as workflows with all the relevant settings and transformations. There is no limitation in this identification process. It can be as simple or as complex as required.

The workflow in Figure 30 shows the model training workflow used in Figure 29 with the Integrated Deployment nodes added. The data preparation and model application parts of the workflow are captured, stitched together, and persisted as a new workflow. In this case, the generated workflow with the embedded model is automatically deployed to a KNIME Server instance.

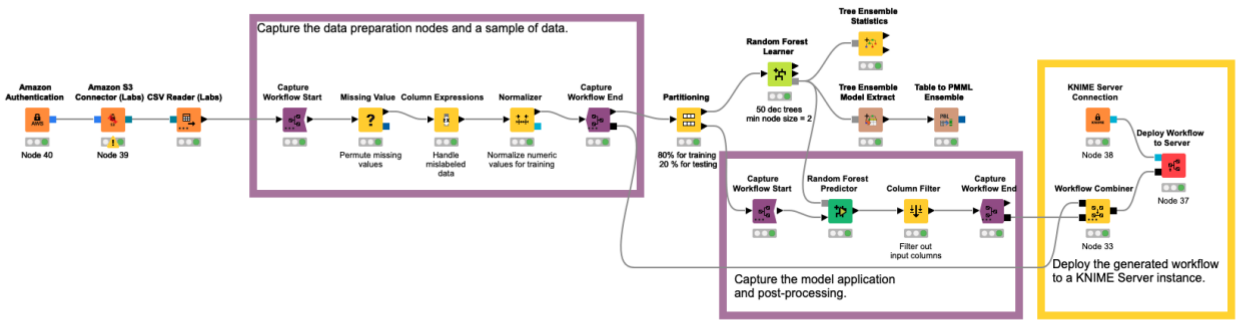


Figure 30 – Integrated Deployment of a model as a generated workflow

Integrated Deployment allows for combining model creation, including pre- and post-processing, into a single workflow that is deployment ready. As design of the model preparation and training changes, those changes are automatically captured in the workflow and regenerated and redeployed to your chosen target. Closing the gap between model readiness and production readiness as described here can increase the productivity of data science teams drastically.

Generated workflows can be persisted to the KNIME Server, an Amazon S3 bucket, the local KNIME environment, or anywhere KNIME can write data. In the preceding example, we show deploying the generated workflow directly to a KNIME Server.

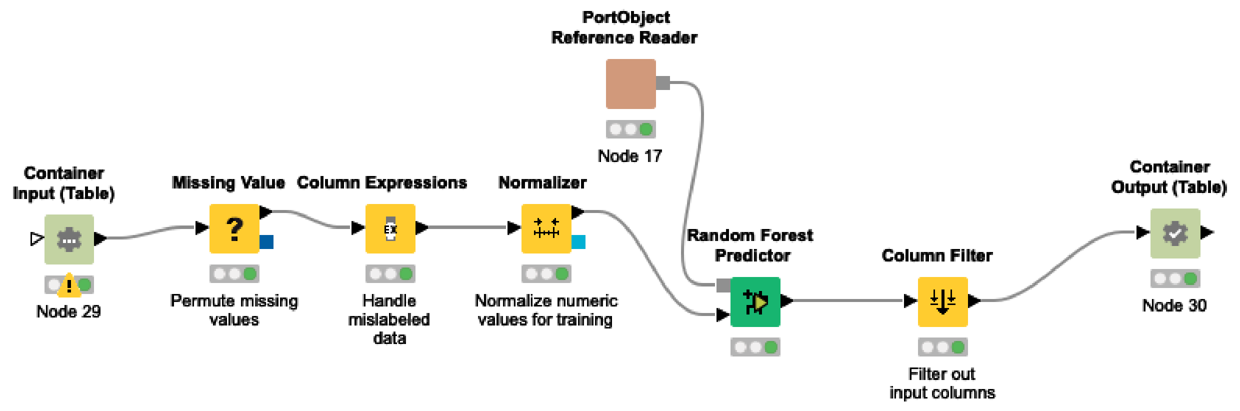


Figure 31 – The generated workflow deploying the trained model

The workflow in Figure 31 is the generated, production-ready workflow. The workflow uses KNIME container nodes, which are placeholders in the workflow for data. Workflows generated by Integrated Deployment are ready for deployment into KNIME Server as API Endpoints. The Container Input node captures the payload from a REST API call. Similarly, the data in the Container Output node is used as the payload for the response.

KNIME provides integration with external software that can be used to integrate with CI/CD pipeline components, such as Jenkins and GitLab. For example, Figure 32 outlines the process of writing a generated workflow to Amazon S3 and using a Lambda trigger to automatically deploy the workflow to KNIME Server. The Lambda function can also be used to kick off a set of workflow tests within KNIME Server to validate the new workflow as part of a CI/CD flow. Likewise, workflows can be used to integrate with external systems.

All of these nodes are available as native nodes, ready to use in KNIME Analytics Platform.

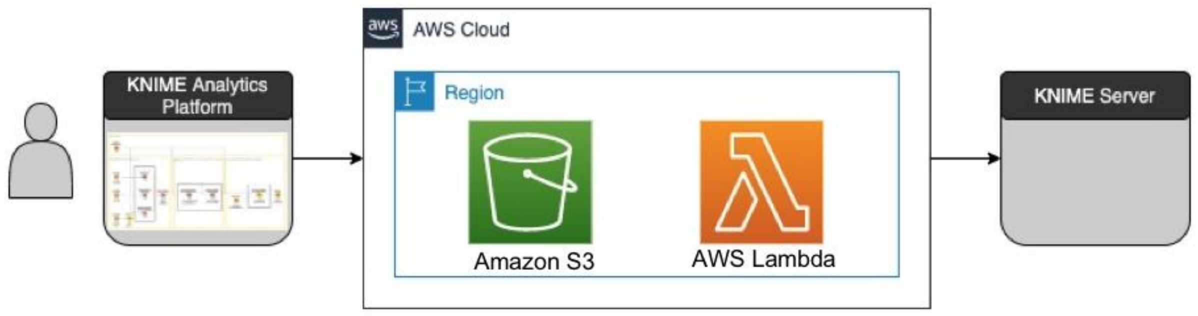


Figure 32 – Automating deployment and testing using Amazon S3 and Lambda

Continuous delivery and monitoring

As mentioned, KNIME Server supports deploying models encased in workflows providing an inference service as an API endpoint. Workflow API endpoints are a ubiquitous way to integrate model inferencing into any application or IT process. Model inferencing can also be accomplished in batch mode.

KNIME also supports continuous delivery of models and model monitoring using a concept we call the Model Process Factory. This provides a flexible and extensible means to monitor model performance and automatically build new, challenger models as needed. We've done all the work in putting it together, ready for you to use directly from the KNIME Hub.

Some highlights are from this section include:

- The orchestration workflow, which is the director of the whole process and provides flexibility, allowing you to define the process to fit your specific needs.
- A number of workflows for initializing, loading, transforming, modeling, scoring, evaluating, deploying, monitoring, and retraining data analytics models.
- Best practices for packaging sub-workflows for quick, controlled, and safe reuse by other workflows.
- Workflows dedicated to checking whether model performance has fallen below a specified accuracy threshold and to retrigger its retraining as needed.

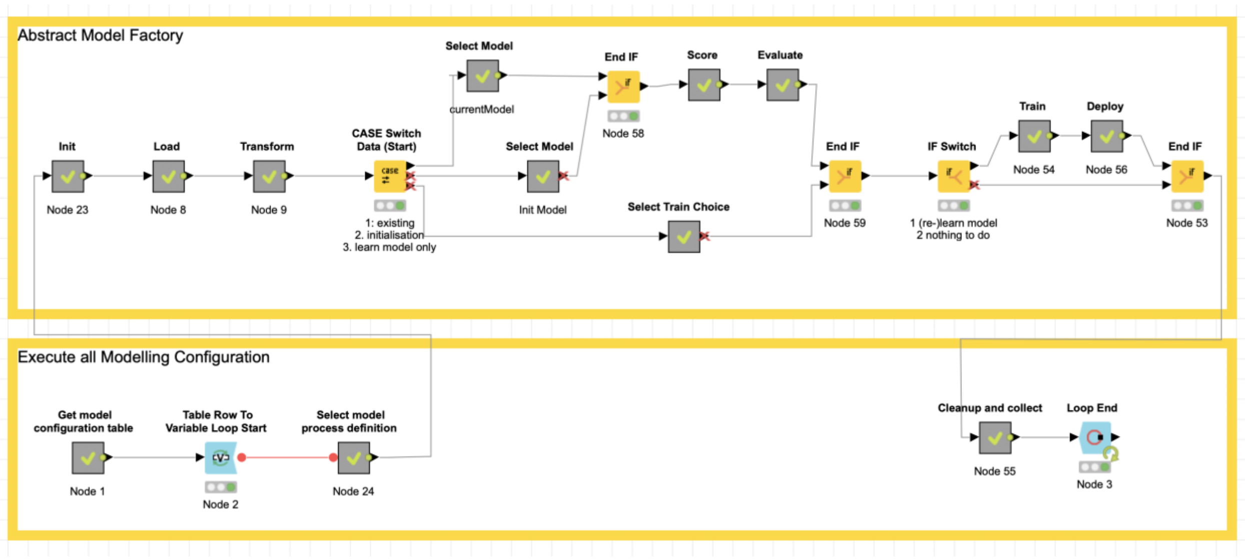


Figure 33 – The orchestration workflow of the Model Process Factory

Interpreting predictions of a model

Many times, end users of ML technology want to understand why a certain prediction or classification was made, because the result from a model inference may seem unintuitive to them. This is where model interpretability is key. KNIME has a comprehensive set of nodes that provide model interpretability. Figure 34 shows a workflow that uses the SHAP nodes to interpret the results of a decision tree model. Combining the interpretability nodes with visualization node(s) can lead to a very useful web application for KNIME Server that can be shared with ML consumers.

All of these nodes are available as native nodes, ready to use in KNIME Analytics Platform.

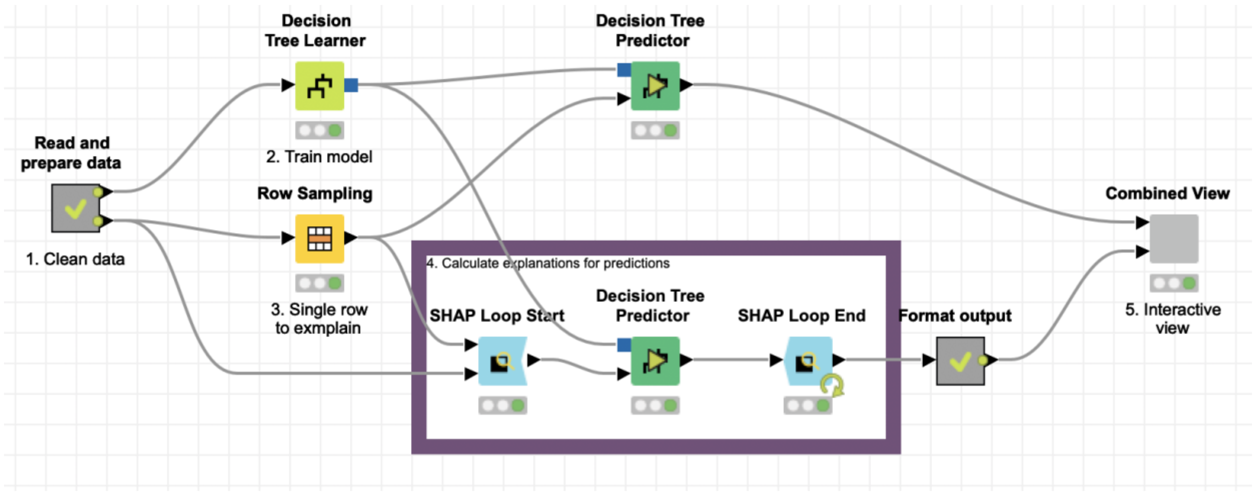


Figure 34 – A workflow using the SHAP nodes to help interpret predictions of a model

KNIME: Your journey ahead

To learn more about KNIME, visit our website. To get started with KNIME on AWS, several product offerings are available in the AWS Marketplace, including a free trial offer for KNIME Server. To learn more about the offerings and how to get started, see KNIME Software on Amazon Web Services.

AWS reference architecture

by Dylan Tong, Machine Learning Partner Solutions Architect, AWS

In this section, we present builders with the option to tailor your own AWS solution. To serve as guidance, we prescribe an architecture designed from production successes, such as this implementation by WordStream and Slalom. This solution leans on AWS Managed Services so that your team can minimize operational overhead and focus on developing differentiated experiences.

We'll explain how the solution comes together by walking through the CD4ML process from ML experimentation to production. We'll present architecture diagrams organized across three disciplines: data science, data engineering, and DevOps, so that you get a sense for organizational ownership and collaboration.

Model building

As prescribed by CD4ML, the process starts with model building and the need for data curation.

Discoverable and accessible assets

Success in ML is rooted in a solid data architecture to manage datasets. This solution prescribes a data lake design. A data lake will enable your data engineers to curate datasets at scale, enabling ML development across your organizations.

Many AWS services and partner technologies are designed to operate on this architecture. Thus, it provides a future-proof design for an evolving ML toolchain.

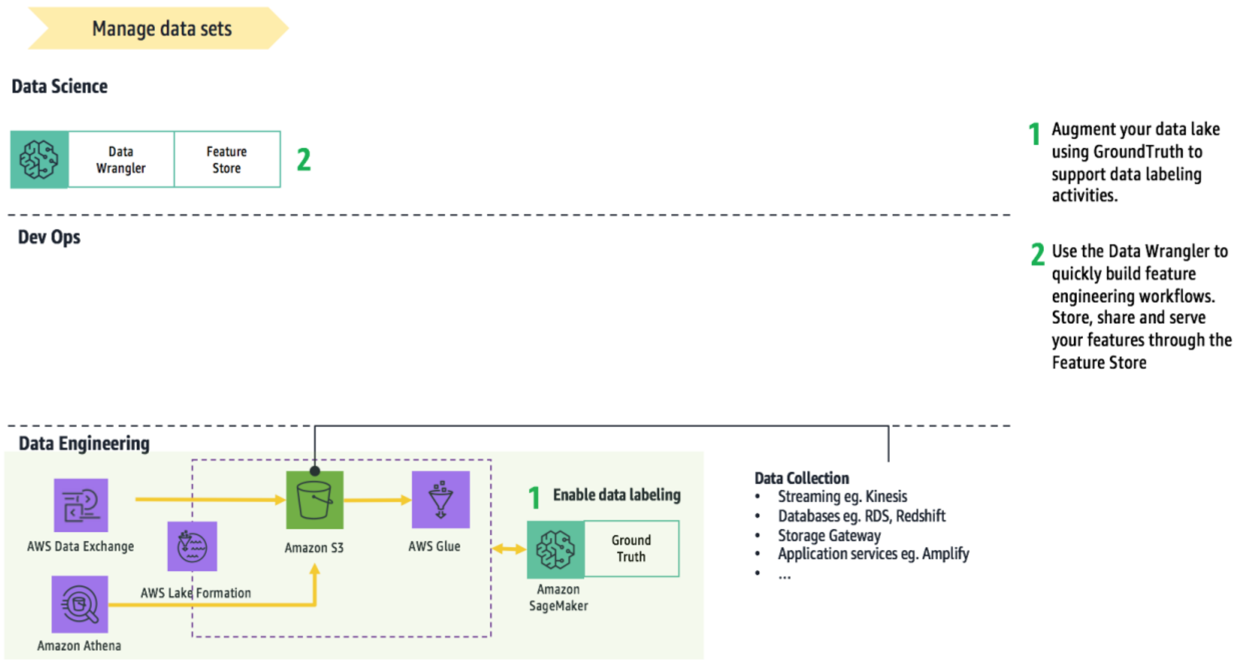

Manage data sets
Data Science
Data Wrangler
Feature Store
2
Dev Ops
Data Engineering
AWS Data Exchange
AWS Lake Formation
Amazon Athena
Amazon S3
AWS Glue
1 Enable data labeling
Ground Truth
Amazon SageMaker
Data Collection
• Streaming eg. Kinesis
• Databases eg. RDS, Redshift
• Storage Gateway
• Application services eg. Amplify
• ...
1 Augment your data lake using GroundTruth to support data labeling activities.
2 Use the Data Wrangler to quickly build feature engineering workflows. Store, share and serve your features through the Feature Store

Figure 35 – An AWS data lake for facilitating discoverable and accessible data

Your organization will be able to get a data lake up and running in quickly by using AWS Lake Formation. Lake Formation brings together core data services and centralizes data access controls to simplify data management.

Amazon S3 provides storage, AWS Glue provides data processing and a catalog, and Amazon Athena enables analysis on your data in Amazon S3. These fully-managed components provide core data governance and curation capabilities without the need to manage infrastructure.

Once in place, the data lake supports a wide range of data ingestion methods to centralize your data sources. Your team can also augment your datasets through the AWS Data Exchange. The exchange provides a diverse catalog of industry data sources—some of which can be used for ML—ranging from financial services, economics, retail, climate and beyond.

At some point, your data science teams will need support for data labeling activities. Amazon SageMaker Ground Truth will enable your team to scale and integrate labeling activities into your data lake. It provides integrated workflows with labeling services, like Amazon Mechanical Turk, to augment your private labeling workforce, automated labeling using active learning to reduce cost, and annotation consolidation to maintain label quality.

As your ML workloads increase and become more sophisticated, your organization will need a better way to share and serve features—the data used for model training and inference. You can augment your data lake with the Amazon SageMaker Feature Store, which allows your organization to ingest, store, and share standardized features from batch and streaming data sources.

Your team can use built-in data transformations from Amazon Data Wrangler to quickly build feature engineering workflows that feed into the Feature Store. These features can be automatically cataloged and stored for offline use cases like training and batch inference. Online use cases that require low latency and high throughput also exist. Imagine your team has to deliver a service to predict credit fraud on real-time transactions. Your model might use transaction, location, and user history to make predictions. Your client app doesn't have access to user history, so you serve these features from the online Feature Store.

Model experimentation

With your data lake in place, you've established a foundation for enabling ML initiatives. Your data science teams can work with their business partners to identify opportunities for ML.

You'll need to provide your data scientists with a platform for experimentation. Figure 36 presents the components in this solution that facilitate experimentation to prove the art-of-the-possible. SageMaker provides ML practitioners with many capabilities in this area.

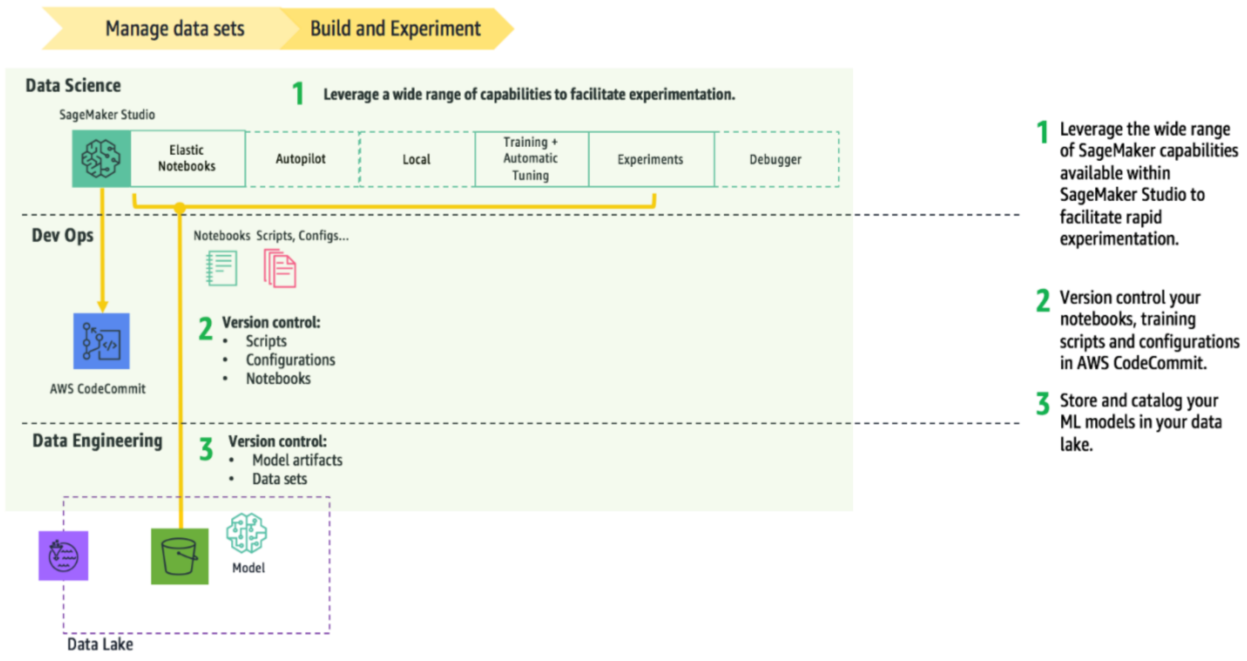
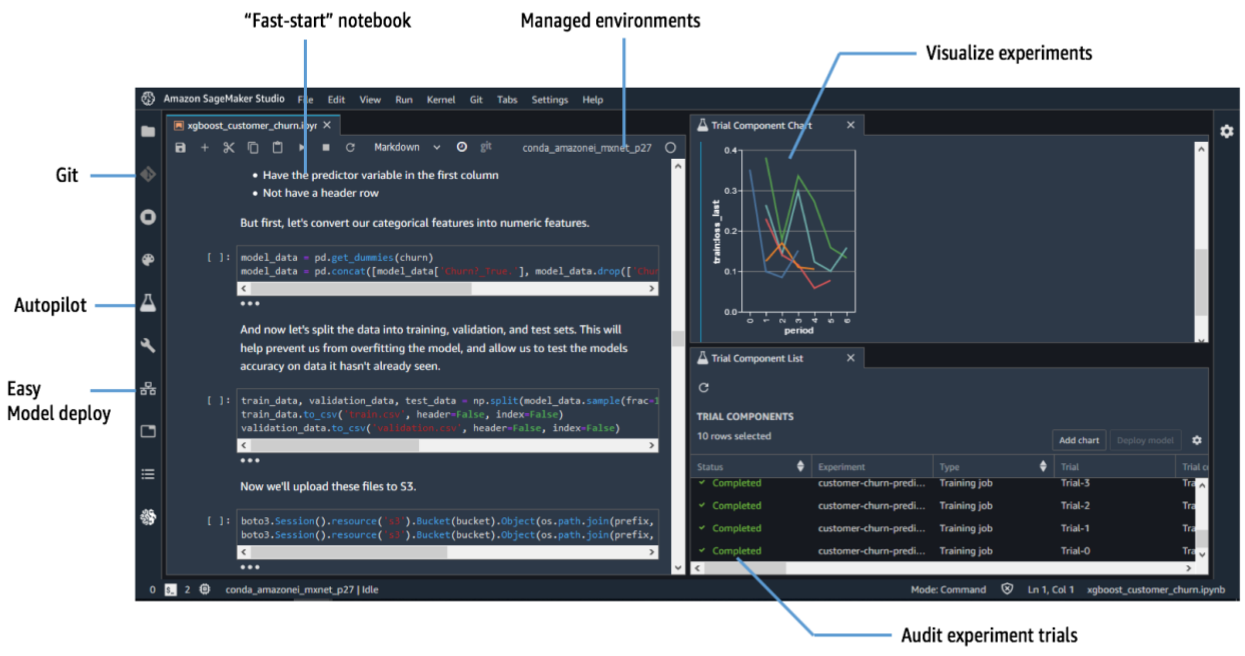

Manage data sets
Build and Experiment
Data Science
SageMaker Studio
1 Leverage a wide range of capabilities to facilitate experimentation.
Elastic Notebooks
Autopilot
Local
Training + Automatic Tuning
Experiments
Debugger
Dev Ops
Notebooks Scripts, Configs...
AWS CodeCommit
2 Version control:
• Scripts
• Configurations
• Notebooks
Data Engineering
3 Version control:
• Model artifacts
• Data sets
Model
Data Lake
1 Leverage the wide range of SageMaker capabilities available within SageMaker Studio to facilitate rapid experimentation.
2 Version control your notebooks, training scripts and configurations in AWS CodeCommit.
3 Store and catalog your ML models in your data lake.

Figure 36 – Facilitating machine learning experimentation

Amazon SageMaker Studio provides you with a fully-managed IDE and a unified user experience across SageMaker. Your Studio environment is secured through AWS Identity and Access Management (IAM) or AWS SSO. Thus, you can authenticate as an AWS user or a corporate identity from a system like Active Directory.

Remember to apply security best practices at all times and across all environments. Your team should become familiar with the security in SageMaker, as outlined in the documentation. You should use IAM controls to govern minimal permissions and apply data protection best practices. You should deploy notebooks in an Amazon Virtual Private Cloud (Amazon VPC), and use Amazon VPC interface endpoints and a hybrid network architecture to enable private network security controls. Internet-free model training and serving options can also be used to support security sensitive use cases.



- Have the predictor variable in the first column
- Not have a header row

But first, let's convert our categorical features into numeric features.

```
[ ]: model_data = pd.get_dummies(churn)
     model_data = pd.concat([model_data['Churn?_True.'], model_data.drop(['Chur
```

And now let's split the data into training, validation, and test sets. This will help prevent us from overfitting the model, and allow us to test the models accuracy on data it hasn't already seen.

```
[ ]: train_data, validation_data, test_data = np.split(model_data.sample(frac=1
     train_data.to_csv('train.csv', header=False, index=False)
     validation_data.to_csv('validation.csv', header=False, index=False)
```

Now we'll upload these files to S3.

```
[ ]: boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix,
     boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix,
```

TRIAL COMPONENTS

10 rows selected

Status	Experiment	Type	Trial	Trial c
✓ Completed	customer-churn-predi...	Training job	Trial-3	Tra
✓ Completed	customer-churn-predi...	Training job	Trial-2	Tra
✓ Completed	customer-churn-predi...	Training job	Trial-1	Tra
✓ Completed	customer-churn-predi...	Training job	Trial-0	Tra

Figure 37 – SageMaker Studio, a fully managed IDE with a one-console experience

Within Studio, your ML practitioners will launch elastic Studio notebooks. These notebooks are fully-managed and include pre-built environments that you can customize. You can select from a variety of popular ML libraries and frameworks, such as TensorFlow, PyTorch, MXNet, Spark ML, and Python libraries like scikit-learn, among others. ML problems come in many forms, as do the skill sets of your engineering resources. Having an arsenal of tools provides your ML practice with flexibility.

Also, experimentation workloads tend to evolve and they come in many forms. An environment that scales efficiently and seamlessly is important for rapid prototyping and cost management. The notebook's compute resources can be adjusted without interrupting your work. For instance, for lightweight experimentation, you can run on burstable CPU and use SageMaker remote training services to minimize cost. Later, you might have a need to experiment with Deep Learning (DL) algorithms, and you could seamlessly add GPU for rapid prototyping.

Throughout ML experimentation, you can share and version your notebooks and scripts through SageMaker's Git integration and collaboration features. This solution prescribes AWS CodeCommit, which provides a fully-managed, secured Git-based repository.

For ML problems on tabular datasets, you can opt to use Amazon SageMaker Autopilot to automate experimentation. Autopilot applies AutoML on data tables and delivers an optimized ML model for regression or classification problems. It does this by automating data analysis and generating candidate ML pipelines that explore a variety of featurization steps and algorithms. It parallelizes a large number of training jobs, performs hyperparameter tuning, model evaluation, and publishes a leaderboard to rank experiments. Autopilot doesn't require your team to manage any infrastructure.

All training jobs are tracked by Amazon SageMaker Experiments. This includes tracking datasets, scripts, configurations, model artifacts and metrics associated with your training runs.

Now that you have an audit trail of all your organizations' experiments, your data scientists can reproduce results.

Productionize the model

As your team shifts from experimentation to productionizing models, you need to increase your focus on scalability, operational efficiency and quality controls.

Scale model training

The need to scale model training is often the first hurdle towards production. For instance, DL typically involves training complex models on large datasets. Without a distributed training cluster and GPUs, it could take days or even weeks rather than hours or minutes to train some models.

To scale and automate training jobs, your ML practitioners will use SageMaker's zero-setup training services to automatically build training clusters, perform distributed training, and parallelize data ingestion on large datasets.

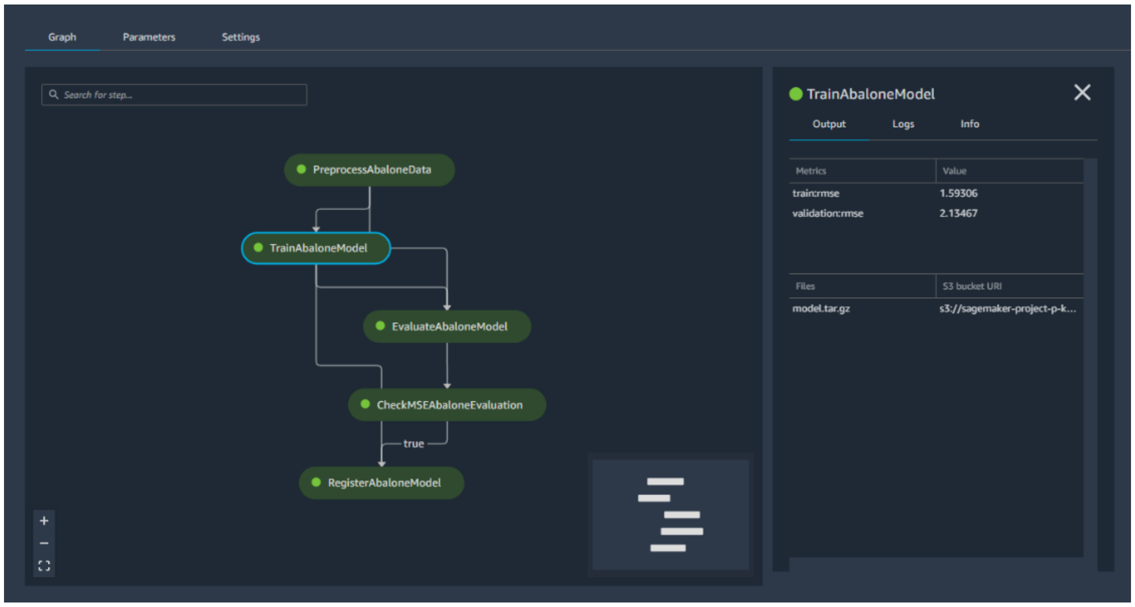
These training services provide options that range from code-free to full-control. Typically, a training process is launched by providing a script or configuring a built-in algorithm. If your data scientists desire full-control, there's the option to "bring-your-own-model" or algorithm by managing your own compatible container.

Your teams will also need the ability to scale the volume of training jobs in addition to scaling individual jobs. This is needed for hyperparameter optimization, a process that involves exploring algorithm configurations (hyperparameters) to improve a model's effectiveness. The process requires training many models with varying hyperparameter values. Your data scientists can choose to use Automatic Model Tuning to automate and accelerate this process.

Automate auditable pipelines

The other critical step towards production is to package your ML projects into ML pipelines. These pipelines address the need for orchestration, operational scalability and auditability.

From within Studio, your data scientists can craft and experiment with different ML workflows using Amazon SageMaker Pipelines as shown in Figure 38.



Metrics	Value
train:rmse	1.59306
validation:rmse	2.13467

Files	S3 bucket URI
model.tar.gz	s3://sagemaker-project-p-k...

Figure 38 – SageMaker Pipelines manages your ML workflows

Figure 39 illustrates the top-level pattern that your ML workflows will typically resemble. In each of these high-level stages, your data scientists—possibly in collaboration with data and DevOps engineers—will craft workflow steps as part of a SageMaker Pipeline.

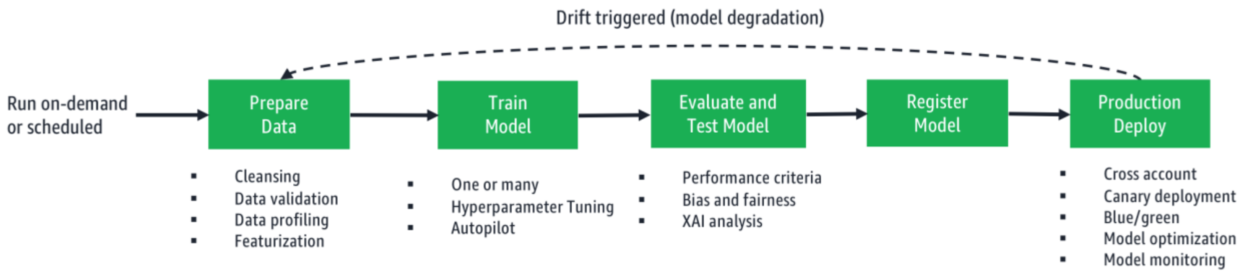


Figure 39 – Common ML workflow pattern

During the *Prepare Data* stage, you'll typically create steps to validate and apply feature engineering transforms to your data. The data prep workflows that your team creates with the Data Wrangler can be exported for use in your production pipelines. Data prep steps could make use of services like Athena and capabilities like SageMaker Processing. Data processing steps can run on a remote Spark cluster and is a natural point for data scientists and data engineers to collaborate.

After data prep, your workflow will introduce one or more training steps to train models using the aforementioned training services. Consider adding hooks into your training job by using Amazon SageMaker Debugger. For instance, you can use built-in rules to stop a training job if it detects a vanishing gradient problem to avoid resource waste. To learn how Autodesk used the Debugger to optimize their training resources, see the blog post, Autodesk optimizes visual similarity search model in Fusion 360 with Amazon SageMaker Debugger.

Additionally, use Debugger's profiling functionality to obtain real-time, granular insights into your training resources. Figure 40 is a real-time heatmap generated by the profiler. It provides insights into resource utilization across CPU and GPU cores during training. These insights help you identify performance issues such as bottlenecks and cost reduction opportunities due to poor resource utilization. The Debugger will prescribe recommendations automatically to guide you through this process.

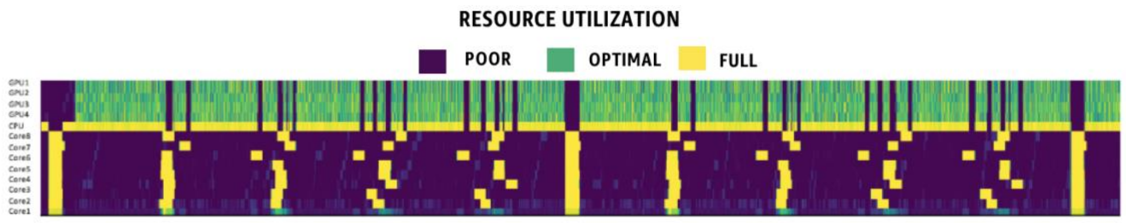


Figure 40 – Real-time Profiling of Training Resource Utilization using the SageMaker Profiler

Consider adding branches in your workflow to recompile your model for different target platforms using Amazon SageMaker Neo. Neo can reduce the size of your models and inference latency.

These steps are followed by model evaluation, testing and deployment stages that we'll discuss later.

Figure 41 summarizes how the key systems at this stage come together. Your DevOps team will likely have a Continuous Integration/Continuous Delivery (CI/CD) pipeline in place to support existing software development practices. If not, you have the option to roll out the solution highlighted in Figure 41. SageMaker provides pipeline templates that can be deployed through AWS Service Catalog. This includes an integrated CI/CD pipeline built on AWS DevOps services.

Your team will commit your ML scripts, which include a SageMaker Pipeline. These scripts will run through your standard CI process. Once your scripts pass your automated tests and approval process, your new SageMaker Pipeline will be activated. The workflow will then execute either on demand, on a scheduled basis, or on event-based triggers.

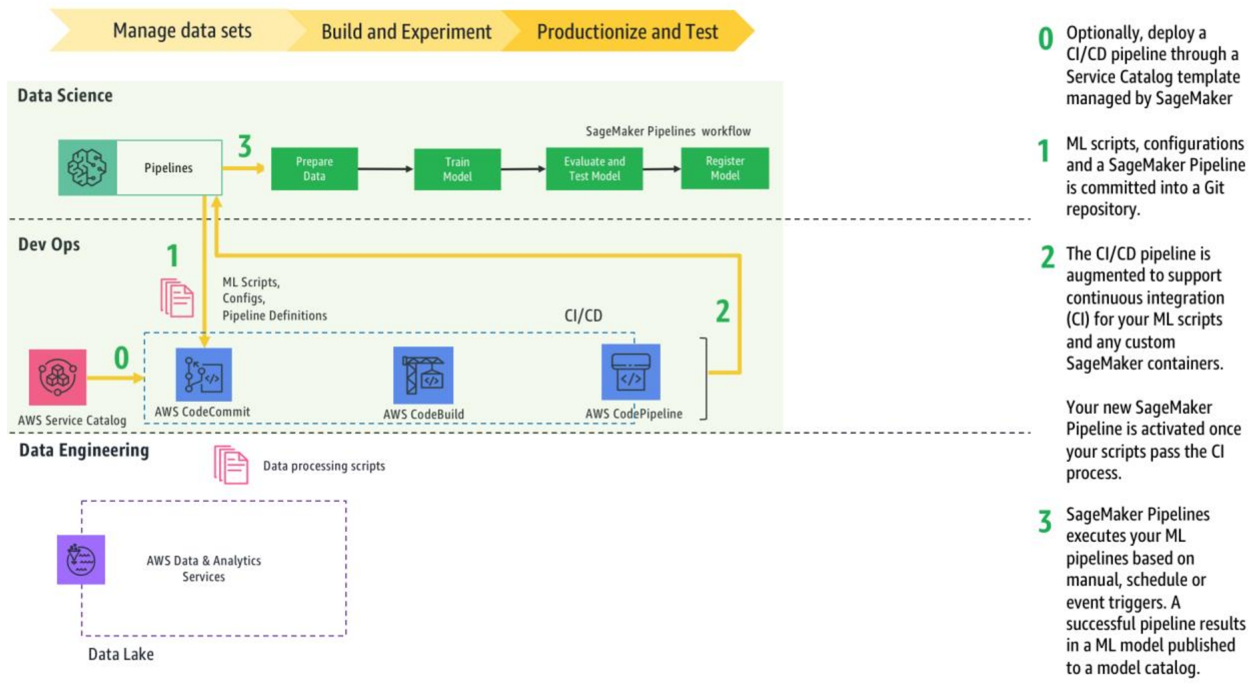

Manage data sets
Build and Experiment
Productionize and Test
Data Science
SageMaker Pipelines workflow
Pipelines
3
Prepare Data
Train Model
Evaluate and Test Model
Register Model
Dev Ops
1
ML Scripts, Configs, Pipeline Definitions
CI/CD
2
0
AWS Service Catalog
AWS CodeCommit
AWS CodeBuild
AWS CodePipeline
Data Engineering
Data processing scripts
AWS Data & Analytics Services
Data Lake
0 Optionally, deploy a CI/CD pipeline through a Service Catalog template managed by SageMaker
1 ML scripts, configurations and a SageMaker Pipeline is committed into a Git repository.
2 The CI/CD pipeline is augmented to support continuous integration (CI) for your ML scripts and any custom SageMaker containers.
Your new SageMaker Pipeline is activated once your scripts pass the CI process.
3 SageMaker Pipelines executes your ML pipelines based on manual, schedule or event triggers. A successful pipeline results in a ML model published to a model catalog.

Figure 41 – Deploying reproducible and auditable ML pipelines

What if your pipeline requirements extend beyond the scope of SageMaker Pipelines? For instance, you may need to manage pipelines for other AWS AI services. To minimize time-to-value, evaluate these other options: AWS Step Functions and the Data Science SDK, Apache Airflow, and Kubeflow Pipelines. These workflow managers have integrations with SageMaker.

Testing and quality

One of the goals of a MLOps practice is to improve quality controls and deliver trusted models. Like any piece of software, ML models are prone to defects and there are consequences when defects leak into production.

Following training in Figure 39 are steps for model evaluation and testing. During this stage, you should create rules to enforce quality standards. For example, you could create a rule that ensures candidate models exceed performance baselines on metrics like accuracy. The probabilistic nature of ML models can introduce edge cases. Your team should brainstorm risky scenarios and create tests accordingly. For example, for some computer vision use cases, you should ensure your models act appropriately when exposed to explicit images.

Your team will also need to update your CI tests. Load tests should be in place to measure the performance of your model server endpoints. Changes to your ML model can affect latency and throughput. SageMaker provides Amazon CloudWatch metrics to assist you in this area, and AWS CodePipeline provides a variety of QA automation tools through partner integrations.

AI fairness and explainability

We've seen PR nightmares play out as a consequence of biased models. In turn, regulatory and business requirements are driving the need for fairness-aware ML and the ability to explain decisions driven by opaque models. Through Amazon SageMaker Clarify, your team has access to utilities for bias detection and Explainable AI (XAI).

Figure 42 illustrates where you can add bias detection into a SageMaker Pipeline. These bias evaluation steps can be incorporated into pre-training, post-training and online stages of your workflow.

You can create workflow steps to measure and mitigate bias in your data. For instance, the Class Imbalance metric can help you discover an underrepresentation of a disadvantaged demographic in your dataset that will cause bias in your model. In turn, you can attempt to mitigate the issue with techniques like resampling.

You can also create steps to identify prediction bias on your trained models. For instance, Accuracy Difference (AD) measures prediction accuracy differences between different classes in your models. A high AD is problematic in use cases such as facial recognition because it indicates that a group experiences mistaken identity disproportionately and this could have an adverse impact.

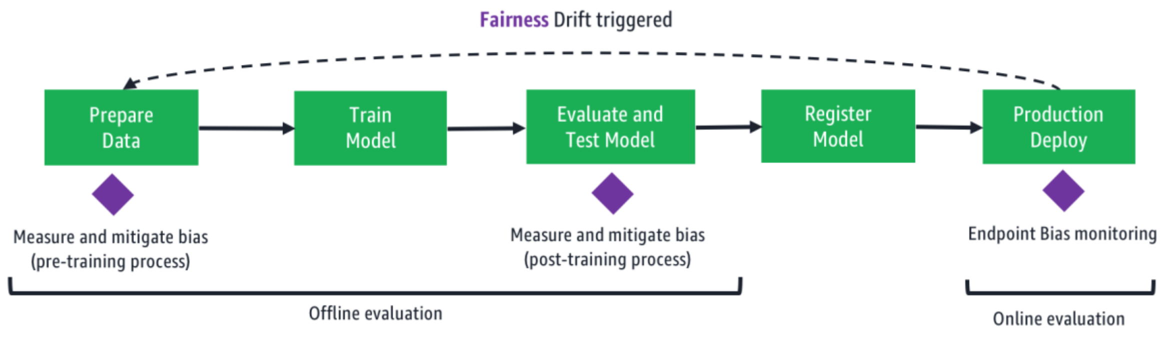


Figure 42 – Bias detection in SageMaker workflows

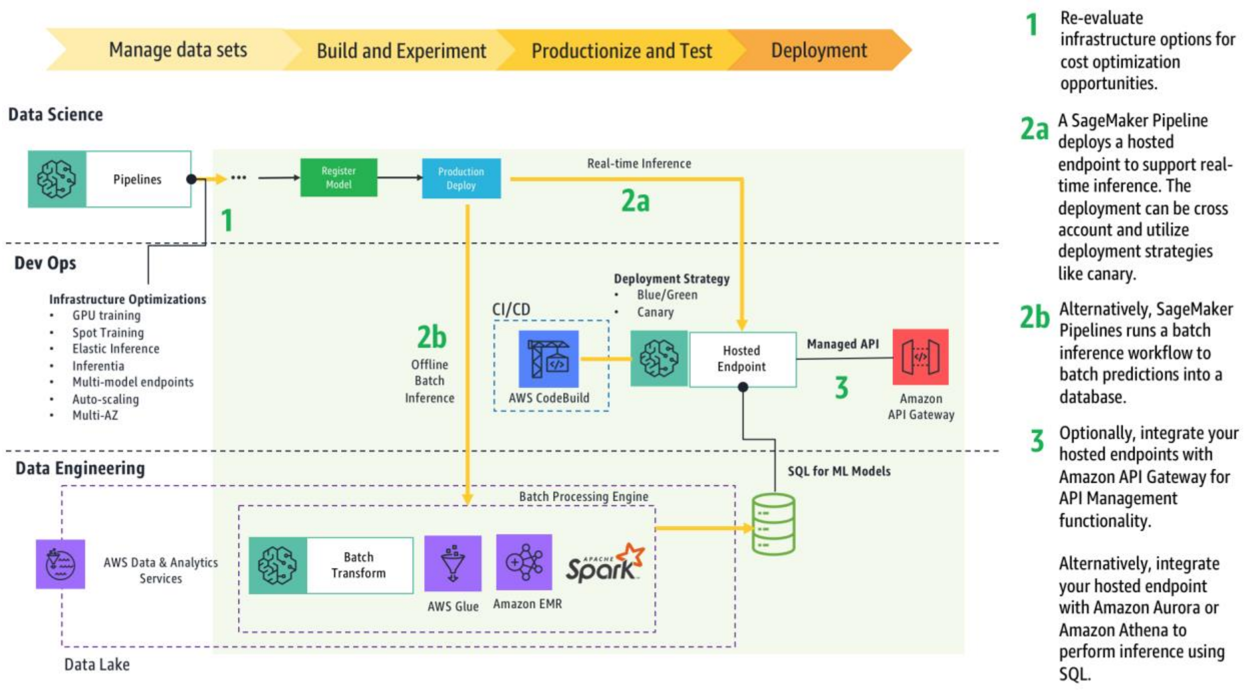
SageMaker provides an implementation of SHAP, which can attribute the impact of input features on a prediction. These attributions can help explain opaque models holistically as well as each prediction made by the model. You can generate SHAP values as part of your workflow to complement your bias analysis. These values can help identify features that might be related to bias issues, or help defend the decisions proposed by your model.

More generally, SHAP values can be included in granular error analysis reports to attribute the impact of features on specific results. This helps your team understand the cause of incorrect predictions and helps validate that correct predictions aren't due to flaws like leakage.

Deployment

Models that pass your quality controls should be added to your SageMaker catalog of trusted models. Next, your team should create workflows to deploy these models into production to support real-time and batch workloads.

Figure 43 illustrates how your system extends to support these scenarios. This phase in the lifecycle will require collaboration across teams.



1 Re-evaluate infrastructure options for cost optimization opportunities.

2a A SageMaker Pipeline deploys a hosted endpoint to support real-time inference. The deployment can be cross account and utilize deployment strategies like canary.

2b Alternatively, SageMaker Pipelines runs a batch inference workflow to batch predictions into a database.

3 Optionally, integrate your hosted endpoints with Amazon API Gateway for API Management functionality.

Alternatively, integrate your hosted endpoint with Amazon Aurora or Amazon Athena to perform inference using SQL.

Figure 43 – Deploy models using best practice deployment strategies

First, this is a good time to re-evaluate your SageMaker Pipelines and ensure your setup is optimized for both performance and cost. Evaluate Amazon Elastic Inference and AWS Inferentia to cost-optimize GPU inference workloads. Leverage multi-model endpoints and automatic scaling to improve resource utilization, and use the provided CloudWatch metrics to monitor the situation. You're likely to have to re-train your model throughout the lifespan of your application, so use managed spot training when it's suitable.

To support real-time predictions, you will create a workflow step to deploy an endpoint on SageMaker Hosting Services. Consider the following best practices:

- For high availability, configure your endpoint to run on two or more instances.
- Set up an automatic scaling policy to adapt to evolving workloads.
- Use A/B testing to mitigate the risk of underperforming model variants.
- Use VPC interface endpoints for SageMaker Runtime to enhance network security.

SageMaker Pipelines support cross-account deployments and strategies like one-box, canary, and blue-green so that you can deploy your models from your development environment into production in accordance with industry best practices.

Next, you can choose to attach your model endpoints to Amazon API Gateway for API management. Among the added benefits are: edge caching, resource throttling, WebSocket support, and additional layers of security.

Alternatively, you can integrate your model endpoints with Amazon Aurora and Amazon Athena. Doing so will allow a data-centric application to blend in predictions using SQL, and reduce data pipeline complexity.

Your team can also build a workflow for batch inference. For instance, you might create a scoring pipeline that pre-calculates prediction probabilities for loan-default risk or customer churn likelihood. Following that, these predictions are loaded into a database to support predictive analytics.

You can create a batch inference step that uses batch transform. Batch transform is designed to scale out and support batch inference on large datasets. It's optimized for throughput and you're only billed on transient resource usage.

Monitoring and observability and closing the feedback loop

Finally, we need to ensure our system can detect problems in production and create a feedback loop that enables continuous improvement.

Model monitoring

As depicted in Figure 44, your SageMaker Pipelines should deploy a Model Monitor alongside your model endpoints to provide drift detection. Your team will choose between the default monitor or a custom monitoring job. When drift is detected, your monitors should be set up to trigger workflows to enable an automated training loop or alert your data scientists about the issue.

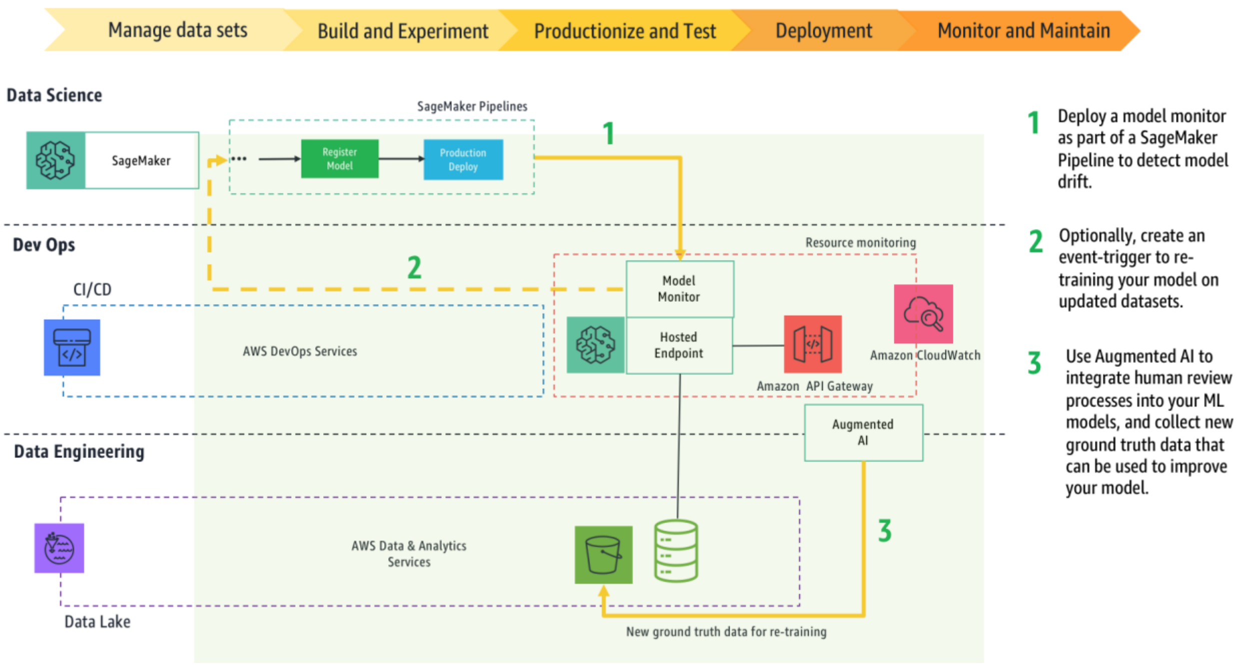

Manage data sets
Build and Experiment
Productionize and Test
Deployment
Monitor and Maintain
Data Science
SageMaker
SageMaker Pipelines
Register Model
Production Deploy
1
Dev Ops
CI/CD
2
AWS DevOps Services
Resource monitoring
Model Monitor
Hosted Endpoint
Amazon API Gateway
Amazon CloudWatch
Augmented AI
Data Engineering
AWS Data & Analytics Services
Data Lake
3
New ground truth data for re-training
1 Deploy a model monitor as part of a SageMaker Pipeline to detect model drift.
2 Optionally, create an event-trigger to re-training your model on updated datasets.
3 Use Augmented AI to integrate human review processes into your ML models, and collect new ground truth data that can be used to improve your model.

Figure 44 – Production monitoring and continuous improvement

Traceability and auditing

In the event a defective model is detected, you can perform root cause analysis with the help of the model lineage tracking system. Your SageMaker Pipelines, whether they are training, real-time inference, or batch workflow, maintain an audit trail of the associated artifacts.

Critical artifacts like datasets, container environments, configurations, models, and endpoints are automatically linked together. Your team is now empowered to run a trace from an artifact version to all the associated assets.

As a best practice, your team should keep a snapshot of the data used to train each of your production model variants and create an Amazon S3 prefix that encodes version tracking metadata. The lineage tracking system currently uses the Amazon S3 location to track data artifacts. Thus, by following this best practice, your team will have traceability between model and dataset versions.

Human review and continuous improvement

To enable the continuous improvement of your models, your team should implement Amazon Augmented AI (A2I) workflows to support human review. Human review in ML is important for supporting sensitive use cases, such as loan approval automation. For these use cases, low confidence predictions should be delegated for human review and decision-making to mitigate risk. You can integrate A2I to enable human intervention and collect feedback for re-training your models.

High-level AI services

In addition to SageMaker, AWS provides other AI services designed for building AI-enabled apps without requiring ML expertise.

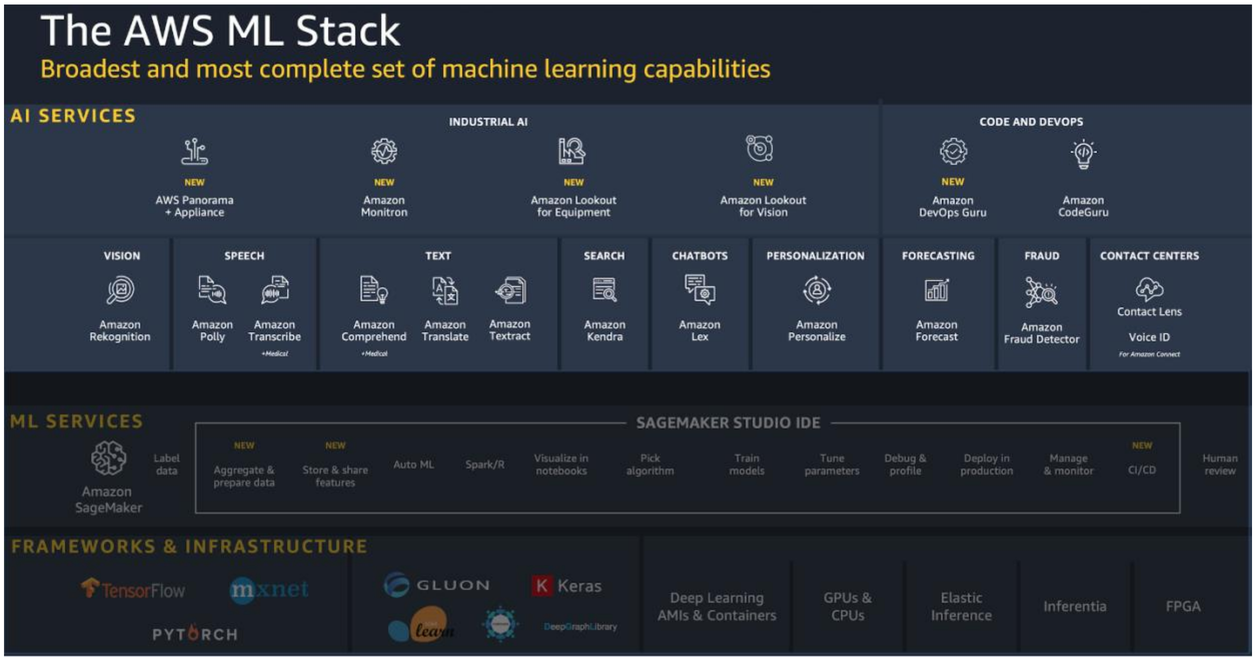


Figure 45 – The Amazon ML stack and high-level AI services

These services provide domain-focused capabilities in areas like computer vision, forecasting, and chatbots. Capabilities typically come in two types:

- **Turnkey** capabilities that are production-ready from the get-go – These capabilities use ML models that are fully-managed by AWS.
- **AutoML** capabilities that deliver customized models – Users provide prepped data and configure AutoML jobs to create custom models designed to support specific use cases.

Productionizing AWS AI services

Services like Amazon Comprehend and Amazon Polly include examples of turnkey capabilities. No changes are needed to existing software engineering practices. These features are integrated into solutions like any plain managed API.

Nonetheless, custom models are often needed to achieve business objectives. Thus, many AWS AI services have both turnkey and AutoML capabilities. Some services like Amazon Fraud Detector and Amazon Forecast are designed around AutoML.

Working with AI service custom models

During the development phase, your team can choose to interface with these AI services using their respective SDK within SageMaker Studio, or use the AWS Management Console.

When you're ready to deploy your AutoML solutions into production, your team should deploy them as automated workflows. These workflows have a common pattern, as shown in Figure 46.

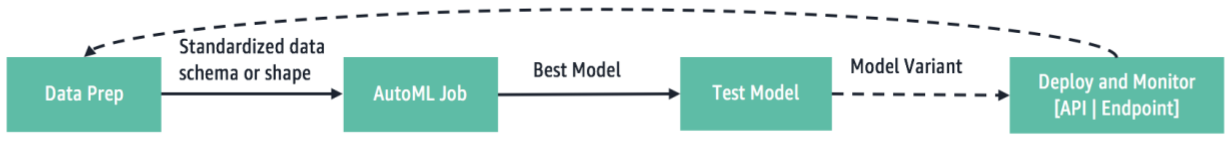

Data Prep
Standardized data schema or shape
AutoML Job
Best Model
Test Model
Model Variant
Deploy and Monitor [API | Endpoint]

Figure 46 – Logical workflow of an AutoML pipeline

Step Functions is a good choice for orchestrating these workflows to create serverless solutions. For human review workflows, A2I has turnkey support for Amazon Rekognition and Amazon Textract. The other AI services require customization.

For drift detection, you'll need to implement a custom solution. You can look to APN Partners to help fill the gap. For instance, Digital Customer Experience partners like Amplitude and Segment integrate with Amazon Personalize to provide marketing and product analytics. These partner technologies could aid in drift detection on recommender models.

As your ML initiatives mature your organization will adopt more ML tools and your team will face the challenge of managing many pipelining tools. AWS provides an MLOps Framework that provides the foundation for unifying workflow technologies like SageMaker Pipelines and Step Functions under a common management framework. The framework is designed to accelerate outcomes by serving workflow blueprints developed by AWS and our open source community.

AWS: Your journey ahead

The journey to MLOps maturity is an evolutionary process. The preceding solution is sometimes built iteratively and can deliver incremental ROI along the way. AWS is here to help you on this journey, along with our AWS Machine Learning Competency Partners who have been vetted as the most capable at delivering ML projects on AWS.

Among these partners are specialists in MLOps. Many of these consulting partners have achieved key competencies in DevOps, data and analytics, and machine learning. Information about these partners has been consolidated in the Resources section. Use the provided links to learn more about the partners and how to contact them.

Conclusion

The APN Machine Learning community provides customers with a choice of outstanding solutions for ML Ops. The following links can help you facilitate the next steps.

- **AWS** – Connect with vetted MLOps consultants and try Amazon SageMaker for free.
- **Alteryx** – Get started with the Alteryx Intelligence Suite Starter Kit.
- **Dataiku –** Launch a free-trial of Dataiku DSS from the AWS Marketplace.
- **Domino Data Lab** – Launch a free-trial of Domino Data Lab's managed offering.
- **KNIME** – Launch a free-trial of the KNIME Server from the AWS Marketplace.
- **ThoughtWorks** – Learn more about ThoughtWorks' CD4ML.

Contributors

- Christoph Windheuser, Global Head of Artificial Intelligence, ThoughtWorks
- Danilo Sato, Head of Data & AI Services UK, ThoughtWorks
- Alex Sadovsky, Senior Director of Product Management, Alteryx
- David Cooperberg, Senior Product Manager, Alteryx
- Greg Willis, Director of Solutions Architecture, Dataiku
- Josh Poduska, Chief Data Scientist, Domino Data Lab
- Jim Falgout, VP of Operations, KNIME
- Dylan Tong, Machine Learning Partner Solutions Architect, AWS

Resources

The following table lists AWS Machine Learning Competency Partners with MLOps expertise.

Partner	**Geo**	**Partner Brief**
Accenture	Global	Learn how Accenture Industrializes Machine Learning on AWS.
AllCloud	Israel, Germany	Learn about the AllCloud Data Practice.
DiUS	Australia, New Zealand, Asian Pacific	Learn about DiUS's ML expertise.
Deloitte	Global	Learn about Deloitte's MLOps expertise.
Inawisdom	UK, Middle East, Benelux	Learn about Inawisdom's DevOps practice for ML.

Partner	Geo	Partner Brief
Intellify	Australia, New Zealand, Asian Pacific	Learn about Intellify's expertise in ML on AWS.
Kainos	UK	Learn about Kaino's ML expertise.
Keepler	Spain, Portugal, Austria, Germany, Switzerland	Learn about Keepler's ML expertise and offerings on AWS.
Max Kelsen	Australia, New Zealand, Asian Pacific	Learn about MLOps Best Practices with Amazon SageMaker & Kubeflow and Max Kelsen's expertise.
1Strategy (TEKsystems)	North America	Learn about 1Strategy's MLOps expertise and offerings.
Onica	Global	Learn about Onica's MLOps Foundations offering.
Pariveda	North America	Learn about Pariveda's ML expertise on AWS.
Peak.ai	UK, US	Learn about Peak's MLOps expertise and offerings on AWS.
Provectus	North America	Learn about Provectus's MLOps expertise and offerings.
Quantiphi	Global	Learn about Quantiphi's MLOps expertise and offerings.
Reply	Italy, Germany, UK	Learn about Reply's offerings on AWS.
Slalom	Global	Learn about Slalom's MLOps expertise and offerings.
TensorIoT	Global	Learn about TensorIoT's ML expertise on AWS.

Document Revisions

Date	Description
December 2020	First publication

Notes

[1] Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation

[2] Jez Humble's Continuous Delivery website: continuousdelivery.com

[3] Continuous Delivery for Machine Learning: Automating the end-to-end lifecycle of Machine Learning applications