Machine Learning in the Enterprise

Course Summary and Key Takeaways

Learning Objectives

- Describe data management, governance, and preprocessing options
- Identify when to use Vertex AutoML, BigQuery ML, and custom training
- Implement Vertex Vizier Hyperparameter Tuning
- Explain how to create batch and online predictions, setup model monitoring, and create pipelines using Vertex AI

Module Breakdown

- Module 0: Introduction
- Module 1: Understanding the ML Enterprise Workflow
- Module 2: Data in the Enterprise
- Module 3: Science of Machine Learning and Custom Training
- Module 4: Vertex Vizier Hyperparameter Tuning
- Module 5: Prediction and Model Monitoring Using Vertex AI
- Module 6: Vertex AI Pipelines
- Module 7: Best Practices for ML Development

Summary

This course encompasses a real-world practical approach to the ML Workflow: a case study approach that presents an ML team faced with several ML business requirements and use cases. This team must understand the tools required for data management and governance and consider the best approach for data preprocessing: from providing an overview of Dataflow and Dataprep to using BigQuery for preprocessing tasks.

The team is presented with three options to build machine learning models for two specific use cases. This course explains why the team would use AutoML, BigQuery ML, or custom training to achieve their objectives. A deeper dive into custom training is presented in this course. We describe custom training requirements from training code structure, storage, and loading large datasets to exporting a trained model.

You will build a custom training machine learning model, which allows you to build a container image with little knowledge of Docker.

The case study team examines hyperparameter tuning using Vertex Vizier and how it can be used to improve model performance. To understand more about model improvement, we dive into a bit of theory: we discuss regularization, dealing with sparsity, and many other essential concepts and principles. We end with an overview of prediction and model monitoring and how Vertex AI can be used to manage ML models.

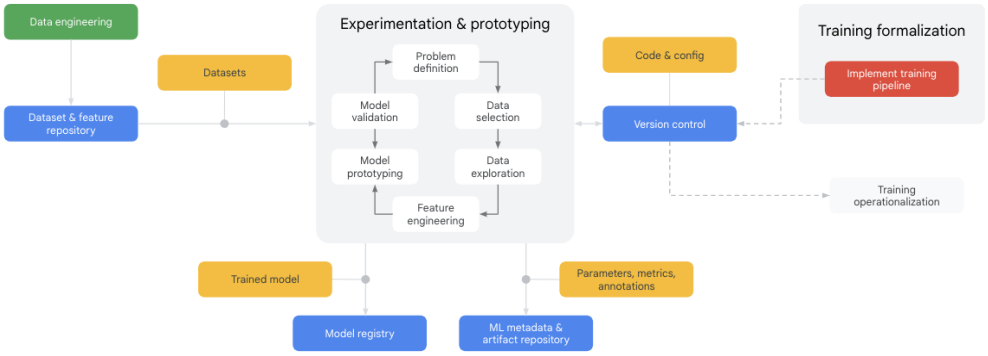
Key takeaways

Module 1: Understanding the ML Enterprise Workflow

ML development typically involves two activities: experimentation and training operationalization.

Data scientists develop models on an **experimentation** platform, iteratively going through the process of: problem refinement, data selection, data exploration, feature engineering, model prototyping, and model validation. This experimentation also can lead to refining the problem definition.

If the model needs to be repeatedly retrained in the future, an automated training pipeline is also developed. We refer to this task as **training operationalization**.



After model development, you then move to **model deployment**.

When the model is deployed to its target environment as a service, it serves predictions to various consumers in the following forms:

- Online inference: in real time, as a REST API
- Streaming inference: near real time, such as in an event processing pipeline
- Batch inference: offline, usually integrated with your ETL processes
- Embedded inference: on an embedded system

All of these aspects of the ML development workflow are supported by **Vertex AI**, from Experimentation to Training Operationalization.

Module 2: Data in the Enterprise

Tools for data management and governance

Feature Store

Three challenges create feature management pain points:

- Features are hard to share and reuse.
- Reliably serving in production with low latency is a challenge.
- Inadvertent skew in feature values between training and serving is common.

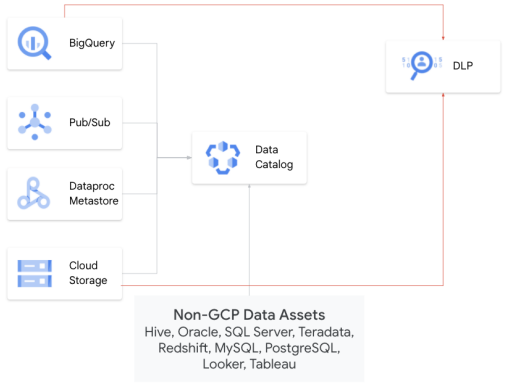
Feature Store is a fully managed solution; it manages and scales the underlying infrastructure for you such as storage and compute resources. This solution means that your data scientists can focus on the feature computation logic instead of worrying about the challenges of deploying features into production.

Benefits of Feature Store:

- Share and reuse ML features across use cases
- Serve ML features at scale with low latency
- Alleviate training serving skew
- Batch and streaming feature Ingestion

Data Catalog

Most organizations today are dealing with a large and growing number of data assets. Data stakeholders (consumers, producers, and administrators) within an organization face a number of challenges in searching for insightful data, understanding data, and making data useful.



Data Catalog can catalog the native metadata on data assets from Google Cloud system sources You can also use Data Catalog APIs to create and manage entries for custom data resource types.

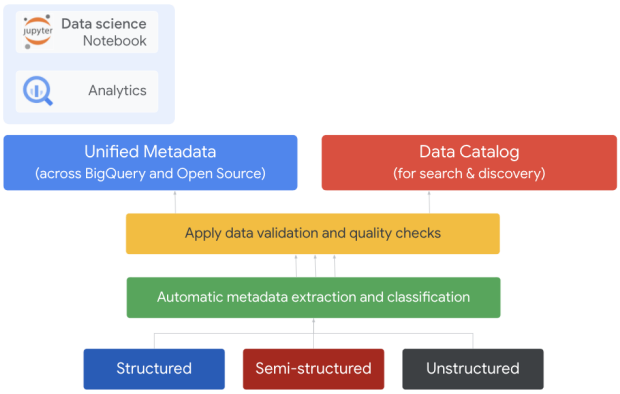
Dataplex

Dataplex's intelligent data fabric that enables organizations to centrally manage, monitor, and govern their data across data lakes, data warehouses, and data marts with consistent controls, providing access to trusted data and powering analytics at scale.

With Dataplex, you can:

- Achieve **freedom of choice** to store data wherever you want for the right price/performance and choose the best analytics tools for the job, including Google Cloud and open source analytics technologies such as Apache Spark and Presto.
- Enforce **consistent controls** across your data to ensure unified security and governance
- Take advantage of the **built-in data intelligence** using Google's best in class AI/ML capabilities to automate much of the manual toil around data management and get access to higher quality data.
- Organize and manage your data in a way that makes sense for your business, without data movement or duplication.

One of the biggest differentiators for Dataplex is our data intelligence capabilities using Google's best in class AI/ML technologies.

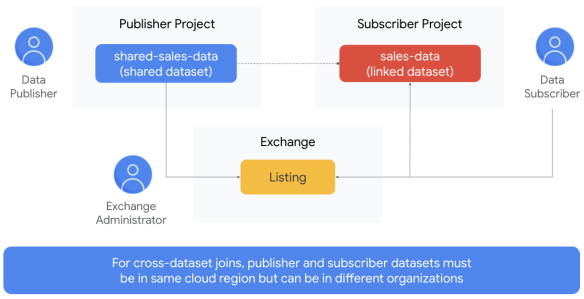


- As you bring the data under management, Dataplex will automatically harvest the metadata for both structured and unstructured data, with built-in data quality checks.
- All of the metadata is automatically registered in a unified metastore, and made available for search and discovery.

Analytics Hub

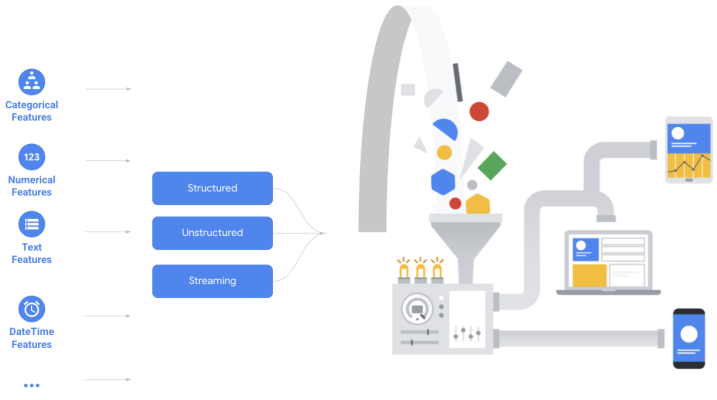
Analytics Hub efficiently and securely exchanges data analytics assets across organizations to address challenges of data reliability and cost. You can create and access a curated library of internal and external assets, including unique datasets like Google Trends, backed by the power of BigQuery.

- Easier for you to publish, discover, and subscribe to valuable datasets that you can combine with your own data to derive unique insights.
- Three roles in Analytics Hub
    a. Data Publisher
    b. Exchange Administrator
    c. Data Subscriber
- Three components
    a. Publisher Project
    b. Subscriber Project
    c. Exchange



Data preprocessing options

If you're using tabular data, use **BigQuery** for data processing and transformation steps. When you're working with ML, use BigQuery ML in BigQuery. Perform the transformation as a normal BigQuery query, then save the results to a permanent table.



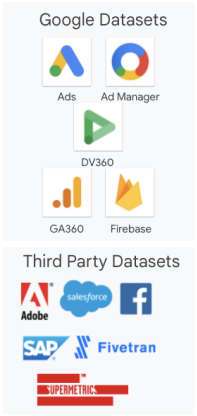
Use **Dataflow** to convert the unstructured data into binary data formats like TFRecord, which can improve performance of data ingestion during training. If you need to perform transformations that are not expressible in Cloud SQL or are for streaming, you can use a combination of Dataflow and the pandas library.

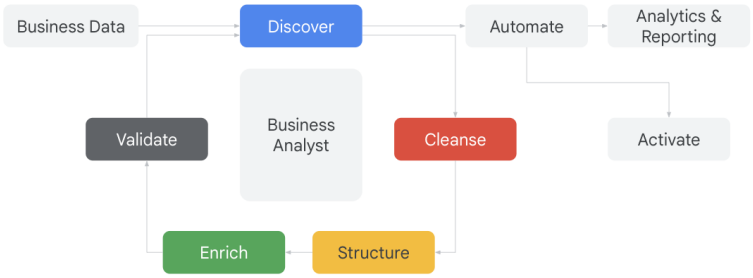
Dataproc is recommended for customers with existing implementations using Hadoop with Spark to perform ETL, or who want to leverage their experience with Hadoop on-premises to create a cloud-based solution.

If you're using TensorFlow for model development, use **TensorFlow Extended** to prepare your data for training. TensorFlow Transform is the TensorFlow component that enables defining and executing a preprocessing function to transform your data.

Dataprep

1. Business data is brought in from various sources.
2. Discover is about finding anomalies and correlations among the data
3. Cleanse is the process of detecting and correcting (or removing) corrupt or inaccurate data and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting it
4. Structure is about changing the format of data to conform to the correct or desired state. For example, changing a string to an actual date, pivoting a rows into columns or extracting data from JSON into a relational model
5. Enrich - is about combining sources and deriving new values that augment the data with further information for analysis
6. Validate is about ensuring the data conforms to the required data set for analysis





Module 3: The Science of Machine Learning and Custom Training

Learning rate and batch size

Learning rate controls the size of the step in the weight space.

- If the steps are too small, training will take a long time.
- If the steps are too large, we will bounce around and could even miss the optimal point.

Batch size controls the number of samples that the gradient is calculated on.

- If the batch size is too small, we could be bouncing around because the batch may not be a good enough representation of the input.
- If the batch size is too large, training will take a very long time.

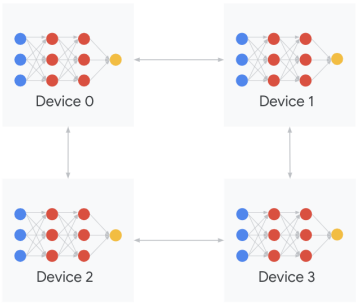
Model performance is very sensitive to learning rate and batch size. Larger batch sizes require smaller learning rates. However, deep learning is still an art, and there are no hard and fast rules for picking a learning rate and batch size.

Heterogeneous systems require our code to work anywhere. If you want high-performance training and scalable inference, chances are that the code that you write has to work on:

- CPU
- GPU
- TPU

- Android/iOS
- Edge TPU
- Raspberry Pi

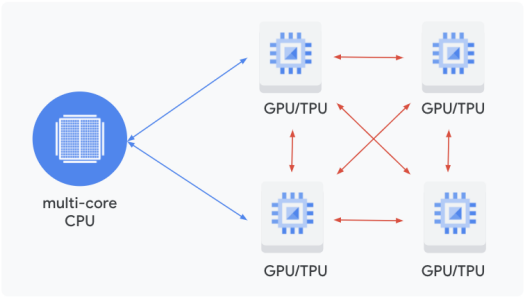
Training complex networks with large amounts of data can often take a long time.



Scaling with **distributed training**

You can go from using one machine with a single device to a machine with multiple devices attached to it.

And to multiple machines with possibly multiple devices each, connected over a network. Eventually, with various approaches you can scale up to 100s of devices (and that's indeed what we do in several google systems).

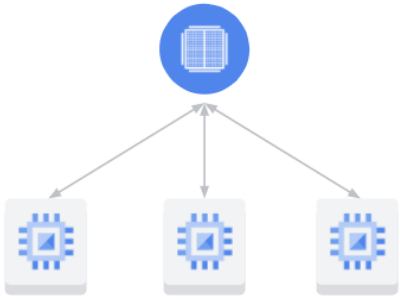


Async parameter versus Allreduce parameter

Async parameter server

Consider async parameter server when you:

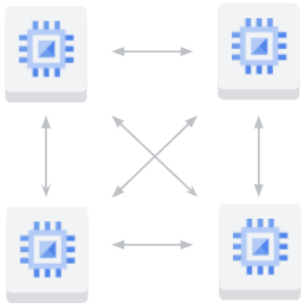
- Have a large number of not-so-powerful or unreliable workers, such as a cluster of machines with just CPUs
- Want a more mature approach
- Have a model that is primarily I/O bound



Allreduce

Consider Allreduce when you:

- Have fast devices with strong communication links, such as multiple GPUs on one host, or TPUs
- Have multiple GPUs
- Have a model that is more computer bound



When to use AutoML and when to use custom training

	AutoML	Custom training
Data science expertise needed	No.	Yes; to develop the training application and also to do some of the data preparation, like feature engineering.
Programming ability needed	No; AutoML is codeless.	Yes; to develop the training application.
Time to trained model	Lower; less data preparation is required, and no development is needed.	Higher; more data preparation is required, and training application development is needed.
Limits on machine learning objectives	Yes; you must target one of AutoML's predefined objectives.	No.
Can manually optimize model performance with hyperparameter tuning	No; AutoML does some automated hyperparameter tuning, but you can't modify the values used.	Yes; you can tune the model during each training run for experimentation and comparison.
Can control aspects of the training environment	Limited; for image and tabular datasets, you can specify the number of node hours to train for and whether to allow early stopping of training. number of nodes.	Yes; you can specify aspects of the environment, such as Compute Engine machine type, disk size, machine learning framework, and number of nodes.
Limits on data size	Yes; AutoML uses managed datasets, and data size limitations vary depending on the type of dataset. Refer to one of the following topics for specifics:	For unmanaged datasets, no. Managed datasets have the same limits as Vertex AI datasets that are used to train AutoML models.

Benefits of using the Vertex AI custom training service

- **Local training first:** Instead of training your model directly within your notebook instance, you can submit a training job from your notebook.
- **Modularize architecture:** Put your training code into a container to operate as a portable unit.
- **Cloud logging:** Each training job is tracked with inputs, outputs, and the container image used.
- **Distributed training:** You can train models across multiple nodes in parallel.

Training code requirements

- Training code structure: Use the Python training application with a pre-built container OR custom container image.
- Load input data: Don't store training data together with your code.
- Load a large dataset: Read data that is too large to fit in memory, stream the data, or read it incrementally.

“Custom” training code requirements

- Export a trained model: After training is complete, export model artifacts to a Cloud Storage bucket.
- Cloud Storage: Read and write Cloud Storage files with Cloud Storage FUSE.
- Optional training: To use certain optional custom training features, change your training code.
- Custom examples: Write code for hyperparameter tuning, distributed training, GPUs, or TensorBoard.

Dependencies

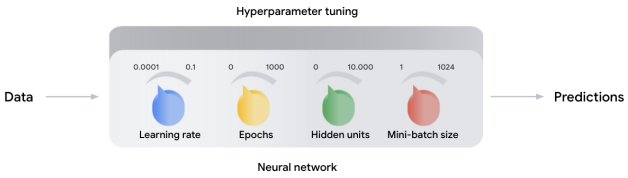
- Using a requirements.txt file for Python dependencies
- Using a setup.py file for Python dependencies
- Specifying individual PyPI dependencies
- Specifying local Python dependencies
- Including other files

Train with Vertex AI

Train with	Data Analyst	Software developer	Data scientist	Use when
AutoML	✔	✔	✔	• Your problem fits into one of the types AutoML supports. Offers a point-and-click workflow. • Natural Language or Video models are served from Google Cloud; Vision and Tables support edge/ • downloadable models.
BigQueryML	✔	✔	✔	• All your data is contained in BigQuery. • Users are most comfortable with SQL. • The set of models available in BigQuery ML • matches the problem you're trying to solve.
VertexAI Training			✔	• You need more flexibility and customization than the criteria listed below for BigQuery ML or AutoML. • You're already running training on-premises or on another cloud, and you need consistency across the platforms.

Module 4: Vertex Vizier Hyperparameter Tuning

Although machine learning models automatically learn from data, they still require user-defined knobs to guide the learning process. These knobs, commonly known as hyperparameters, control, for example, the tradeoff between training accuracy and generalizability.



Examples of hyperparameters are the optimizer, its learning rate, epochs, regularization parameters, the number of hidden layers in a DNN, and their sizes. Setting hyperparameters to their optimal values for a given dataset can make a huge difference in model quality.

Vertex Vizier is a black-box optimization service that helps you tune hyperparameters in complex machine learning (ML) models.

It offers three types of hyperparameter tuning:

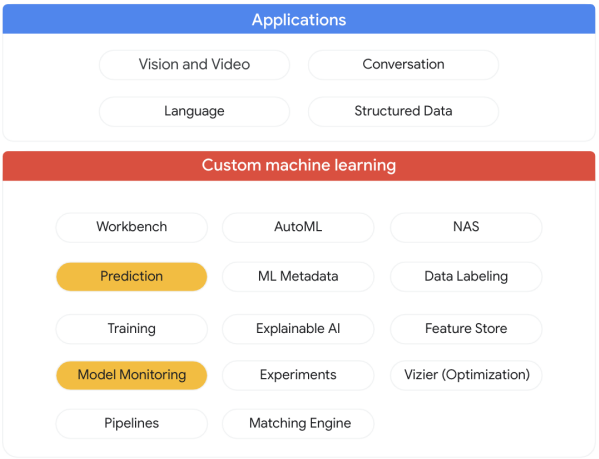
Grid Search	Random Search	Bayesian optimization
Very traditional technique for hyperparameter tuning. In the Grid Search method, we can set up a grid of specific model hyperparameters and then train/test our problem statement model on every combination of values.	Set up a grid of specific model hyperparameter values (the same as with the grid search), but here we select the combination of hyperparameter values randomly. A Random Search tuning technique is faster than a Grid Search, but a Grid Search is more effective than a Random Search because a Random Search misses a few combinations.	Another method of hyperparameter tuning that takes into account past evaluations when choosing which hyperparameter set to evaluate next. This approach typically requires fewer iterations to get to the optimal set of hyperparameter values, most notably because it disregards those areas of the parameter space that it believes won't produce useful results. This in turn limits the number of times a model needs to be trained for validation, because only those settings that are expected to generate a higher validation score are passed through for evaluation.

How to do hyperparameter tuning with vertex Vizier:

1. Environment setup: Enable APIs (Compute Engine, Vertex API)
2. Launch Vertex Workbench: User-managed notebook
3. Containerize the application code
4. Run a hyperparameter tuning job on Vertex AI

Module 5: Prediction and Model Monitoring Using Vertex AI

Prediction and Model Monitoring both allow you to request either batch and online predictions—via a BigQuery table, a CSV file, or from a pre-built or custom container—and to monitor training-serving skew.



Prediction

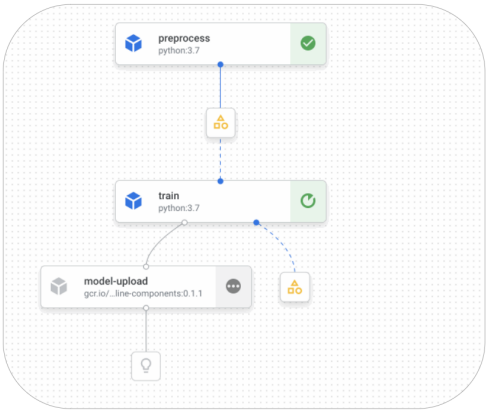
1. **Batch prediction** lets you make many prediction requests at once. Batch prediction is asynchronous, which means that the model will wait until it processes all of the prediction requests before returning a CSV file or BigQuery Table with prediction values.
2. **Online prediction** is useful if your model is part of an application, and parts of your system are dependent on a quick prediction turnaround.

Vertex AI Model Monitoring is a service that helps you manage the performance of your ML models.

1. Detect drift in data quality
2. Identify skew in training versus serving data
3. Monitor feature attribution
4. Use the UI to visualize monitoring metrics

Module 6: Vertex AI Pipelines

Vertex AI Pipelines lets you orchestrate your machine learning workflows in a serverless manner. ML pipelines are portable and scalable ML workflows that are based on containers and Google Cloud services.



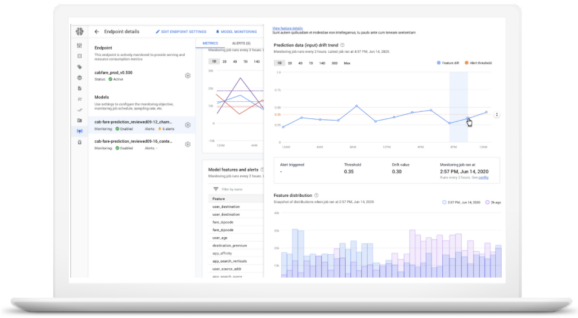
You must describe your workflow as a pipeline.

- A pipeline is composed of modular pieces/components
- It offers automation and orchestration
- Components are chained with dsl to form a pipeline

Run pipelines built using the **Kubeflow Pipelines SDK** v1.8.9 or higher, or **TensorFlow Extended** v0.30.0 or higher.

Module 7: Best Practices for ML Development

ML model deployment and serving best practices



Model deployment and serving refers to putting a model into production. The output of the training job is a model file stored on Cloud Storage, which you can upload to Vertex AI so the file can be used for prediction serving.

For best practices for model deployment and serving:

- Specify the number and types of machines you need
- Plan inputs to the model
- Turn on automatic scaling
- Define what is good and bad performance

ML model monitoring best practices

Best practices for model monitoring include:

- Using skew detection
- Fine tuning alert thresholds
- Using feature attributions to detect data drift or skew
- Tracking outliers

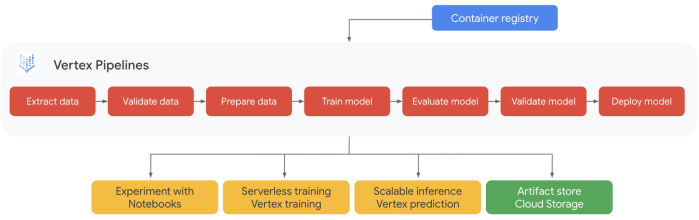
Note: Model monitoring works for structured data, like numerical and categorical features, but not for unstructured data, like images.

Vertex AI Pipeline best practices

Vertex AI Pipelines helps you to automate, monitor, and govern your ML systems by orchestrating your ML workflow in a serverless manner. Pipelines automates the training and deployment of models.

When you run a pipeline using Vertex AI Pipelines, the artifacts and metadata of your pipeline run are stored using Vertex ML Metadata. As a best practice:

- Assess perfection: Why did a certain pipeline run produce an especially accurate model?
- Compare pipelines: Which pipeline run produced the most accurate model, and what hyperparameters were used to train the model?



- Implement system governance: Depending on the steps in your pipeline, you may be able to use Vertex ML Metadata to answer system governance questions. For example, you could use metadata to determine which version of your model was in production at a given point in time.

ML model deployment and serving best practices

Artifacts are outputs resulting from each step in the ML workflow. It is a best practice to organize them in a standardized way.

- Use a Git repo for pipeline definitions and training code.

- Use Artifact Registry to store, manage, and secure your Docker container images without making them publicly visible.
    - Source control repo location, where artifacts such as notebooks and pipeline source code can be stored
    - Experiments and ML Metadata, where artifacts such as experiments, parameters, and metrics can be stored
    - Artifact Registry, where artifacts such as pipeline containers and custom training environments are stored