Google Cloud

# Professional Machine Learning Engineer

## Study Guide


v2.0

# Contents

# Introduction

The Google Cloud Professional Machine Learning Engineer training and exam are intended for individuals with a strong technical background and hands-on experience in machine learning (ML) who want to demonstrate their expertise in designing, building, and deploying machine learning solutions on Google Cloud.

The exam validates a candidate's ability to complete the following course objectives:

- Describe how to develop and implement machine learning solutions using low-code tools and services on Google Cloud.
- Explain how to effectively manage data, prototype models, and collaborate within and across teams to build robust ML solutions.
- Determine how to scale ML prototypes into production-ready models by selecting appropriate frameworks, training effectively, and choosing optimal hardware.
- Describe how to deploy and scale ML models in production using various serving strategies and infrastructure on Google Cloud.
- Explain how to automate and orchestrate end-to-end ML pipelines to streamline model development, deployment, and retraining.
- Identify the key tasks and considerations for monitoring, testing, and troubleshooting ML solutions to ensure performance, reliability, and responsible AI practices.

# Learn more about the exam

A Professional Machine Learning Engineer builds, deploys, and optimizes AI solutions on Google Cloud, leveraging strong programming skills and ML expertise to handle complex datasets and create scalable systems. They collaborate effectively with other teams, champion responsible AI practices, and enable the organization to leverage the power of AI.

The Professional Machine Learning Engineer Certification exam assesses the knowledge and skills required to effectively leverage Google Cloud's machine learning capabilities to solve real-world problems.

The Professional Machine Learning Engineer exam assesses knowledge in six areas:

- Architecting low-code AI solutions.
- Collaborating within and across teams to manage data and models.
- Scaling prototypes into ML models.
- Serving and scaling models.
- Automating and orchestrating ML pipelines.
- Monitoring AI solutions.

To get started, you can explore the updated Machine Learning Crash Course, which provides a foundation in machine learning fundamentals and offers guided, hands-on coding experience through Codelabs.

For a comprehensive learning journey towards the Professional Machine Learning Engineer certification, sign up for the Professional Machine Learning Engineer Learning Path through Google Cloud Skills Boost, Coursera, or Pluralsight.

Prepare for the exam with sample questions.

Learn more about how and where to take the exam on the Professional Machine Learning Engineer website.

# Outline of learning path content

*For learners who have taken Google Cloud's instructor-led training the terminology differs slightly. Each course listed here refers to one module and each module refers to a lesson.*

*To provide a concise overview of the core curriculum, introductory and summary modules that typically appear at the beginning and end of these courses have been excluded from this list. These modules generally recap the course scope and content.*

## 01 Introduction to AI and Machine Learning on Google Cloud

**Module 1**: AI Foundations on Google Cloud

**Module 2**: AI Development on Google Cloud

**Module 3**: ML Workflow and Vertex AI

**Module 4:** Generative AI on Google Cloud

## 02 Prepare Data for ML APIs on Google Cloud

**Lab 1**: Vertex AI: Qwik Start

**Lab 2:** Dataprep: Qwik Start

**Lab 3:** Dataflow: Qwik Start - Templates

**Lab 4:** Dataflow: Qwik Start - Python

**Lab 5:** Dataproc: Qwik Start - Console

**Lab 6:** Dataproc: Qwik Start - Command Line

**Lab 7:** Cloud Natural Language API: Qwik Start

**Lab 8:** Speech-to-Text API: Qwik Start

**Lab 9:** Video Intelligence: Qwik Start

**Lab 10:** Prepare Data for ML APIs on Google Cloud: Challenge Lab

## 03 Working with Notebooks in Vertex AI

**Mini-course**: 8 lessons

## 04 Create ML Models with BigQuery ML

**Lab 1**: Getting Started with BigQuery ML

**Lab 2:** Predict Visitor Purchases with a Classification Model in BigQuery ML

**Lab 3:** Predict Taxi Fare with a BigQuery ML Forecasting Model

**Lab 4:** Bracketology with Google Machine Learning

**Lab 5:** Create ML Models with BigQuery ML: Challenge Lab

## 05 Engineer Data for Predictive Modeling with BigQuery ML

**Lab 1**: Creating a Data Transformation Pipeline with Cloud Dataprep

**Lab 2:** ETL Processing on Google Cloud Using Dataflow and BigQuery (Python)

**Lab 3:** Predict Visitor Purchases with a Classification Model in BigQuery ML

**Lab 4:** Engineer Data for Predictive Modeling with BigQuery ML: Challenge Lab

## 06 Feature Engineering

**Module 1**: Introduction to Vertex AI Feature Store

**Module 2**: Raw Data to Features

**Module 3**: Feature Engineering

**Module 4:** Preprocessing and Feature Creation

**Module 5:** Feature Crosses: TensorFlow Playground

**Module 6:** Introduction to TensorFlow Transform

## 07 TensorFlow on Google Cloud

**Module 1**: Introduction to the TensorFlow Ecosystem

**Module 2**: Design and Build an Input Data Pipeline

**Module 3**: Building Neural Networks with the TensorFlow and Keras API

**Module 4:** Training at Scale with Vertex AI

## 08 Production Machine Learning Systems

**Module 1**: Architecting Production ML System

**Module 2**: Designing Adaptable ML System Designing High-Performance ML Systems

**Module 3**: Designing High-Performance ML Systems

**Module 4**: Hybrid ML Systems

**Module 5**: Troubleshooting ML Production Systems

## 09 Machine Learning Operations (MLOps): Getting Started

**Module 1**: Employing Machine Learning Operations

**Module 2**: Vertex AI and MLOps on Vertex AI

## 10 Machine Learning Operations (MLOps) with Vertex AI: Manage Features

**Module 1:** Introduction to Vertex AI Feature Store

**Module 2:** An In-Depth Look

## 11 Introduction to Generative AI

**Mini-course**: 1 lesson

## 12 Introduction to Large Language Models

**Mini-course**: 1 lesson

## 13 Machine Learning Operations (MLOps) for Generative AI

**Mini Course:** 5 lessons

## 14 Machine Learning Operations (MLOps) with Vertex AI: Model Evaluation

**Module 1**: Introduction to Model Evaluation

**Module 2**: Model Evaluation for Generative AI

## 15 ML Pipelines on Google Cloud

**Module 1**: Introduction to TFX Pipelines

**Module 2**: Pipeline Orchestration with TFX

**Module 3**: Custom Components and CI/CD for TFX Pipelines

**Module 4**: ML Metadata with TFX

**Module 5**: Continuous Training with Multiple SDKs, KubeFlow & AI Platform Pipelines

**Module 6**: Continuous Training with Cloud Composer

**Module 7**: ML Pipelines with MLflow

## 16 Build and Deploy Machine Learning Solutions on Vertex AI

**Lab 1:** Vertex AI: Qwik Start

**Lab 2:** Identify Damaged Car Parts with Vertex AutoML Vision

**Lab 3:** Deploy a BigQuery ML Customer Churn Classifier to Vertex AI for Online Predictions

**Lab 4:** Vertex Pipelines: Qwik Start

**Lab 5:** Build and Deploy Machine Learning Solutions with Vertex AI: Challenge Lab

## 17 Build Generative AI Applications on Google Cloud

**Module 1:** Generative AI Applications

**Module 2:** Prompts

**Module 3:** Retrieval Augmented Generation (RAG)

## 18 Responsible AI for Developers: Fairness and Bias

**Module 1:** AI Interpretability and Transparency

**Module 2:** Modernizing Infrastructure in the Cloud

## 19 Responsible AI for Developers: Interpretability and Transparency

**Module 1**: AI Interpretability and Transparency

**Module 2**: Modernizing Infrastructure in the Cloud

## 20 Responsible AI for Developers: Privacy and Safety

**Module 1:** AI Privacy

**Module 2:** AI Safety

# Exam knowledge areas

## Section 1: Architecting low-code ML solutions

### 1.1: Developing ML models by using BigQuery ML

This section explores the practical application of BigQuery ML for solving real-world business problems. You learn to identify the right BigQuery ML model for different tasks, including linear and binary classification, regression, time series analysis, and more. We delve into feature engineering techniques within BigQuery ML to optimize model accuracy. Finally, you learn how to evaluate model performance by analyzing key metrics like R-squared, precision, recall, and F1-score, and generate both batch and online predictions using your trained models.

**Focus areas:**

- Building the appropriate BigQuery ML model (e.g., linear and binary classification, regression, time-series, matrix factorization, boosted trees, autoencoders) based on the business problem.
- Feature engineering or selection using BigQuery ML.
- Generating predictions by using BigQuery ML.

**Explore in the following courses:**

Introduction to AI and Machine Learning on Google Cloud (Module 2)

Creating ML Models with BigQuery ML

Engineer Data for Predictive Modeling with BigQuery ML

Prepare Data for ML APIs on Google Cloud (Lab 1)

Feature Engineering (Module 3)

Build and Deploy Machine Learning Solutions on Vertex AI (Lab 3)

## 1.2: Building AI solutions by using ML APIs or foundational models

This section provides a practical exploration of building AI-powered applications using pre-trained models and APIs available on Google Cloud. It describes how to select the appropriate Model Garden API for tasks like image classification and language translation, and then integrate it into your application. It also covers using industry-specific APIs for specialized tasks like document processing and retail recommendations. Finally, in this section you gain experience building Retrieval Augmented Generation (RAG) applications with Vertex AI Agent Builder, to leverage external knowledge sources for more comprehensive and informed AI solutions.

**Focus areas:**

- Building applications by using ML APIs (e.g., Cloud Vision API, Natural Language API, Cloud Speech API, Translation) from Model Garden.
- Building applications by using industry-specific APIs (e.g., Document AI API, Retail API).
- Implementing retrieval augmented generation (RAG) applications with Vertex AI Agent Builder by leveraging pre-built components and minimal coding for faster development, or utilizing visual, no-code tools without writing any code.

**Explore in the following courses:**

Introduction to AI and Machine Learning on Google Cloud (Module 3)

Prepare Data for ML APIs on Google Cloud (Lab 7, 8, 9, 10)

Build and Deploy Machine Learning Solutions on Vertex AI (Lab 2)

Create Generative AI Apps on Google Cloud

Build Generative AI Applications on Google Cloud (Lab 4, 5)

## 1.3: Training models by using AutoML

This section focuses on preparing your data for use with AutoML in Vertex AI. It describes how to organize various data types, including tabular, text, images, and videos, for optimal model training. The section also covers data management techniques within Vertex AI, preprocessing steps using tools like Dataflow and BigQuery, and the creation of feature stores. Additionally, it explains the crucial role of feature selection and data labeling in AutoML and explores responsible AI practices by examining privacy implications and how to handle sensitive data.

**Focus areas:**

- Preparing data for AutoML (e.g., feature selection, data labeling, Tabular Workflows on AutoML).
- Using available data (e.g., tabular, text, speech, images, videos) to train custom models.
- Using AutoML for tabular data.
- Creating forecasting models using AutoML.
- Configuring and debugging trained models.

**Explore in the following courses:**

Introduction to AI and Machine Learning on Google Cloud

Working with Notebooks in Vertex AI

# Section 2: Collaborating within and across teams to manage data and models

## 2.1: Exploring and preprocessing organization-wide data (e.g., Cloud Storage, BigQuery, Spanner, Cloud SQL, Apache Spark, Apache Hadoop)

This section covers the crucial steps in preparing and managing your data for machine learning tasks on Google Cloud. It describes how to choose the most suitable storage service for different data types and volumes, considering factors like cost and access patterns. This section explores data preprocessing techniques using tools like Dataflow, TFX, and BigQuery, covering essential steps such as data cleaning, transformation, and feature engineering. Finally, the section emphasizes responsible AI practices by highlighting the importance of data privacy and security, particularly when dealing with sensitive information. It also explains anonymization techniques and Google Cloud tools that help ensure compliance with privacy regulations.

**Focus areas:**

- Organizing different types of data (e.g., tabular, text, speech, images, videos) for efficient training.
- Managing datasets in Vertex AI.
- Data preprocessing (e.g., Dataflow, TensorFlow Extended [TFX], BigQuery).
- Creating and consolidating features in Vertex AI Feature Store.
- Privacy implications of data usage and/or collection (e.g., handling sensitive data such as personally identifiable information [PII] and protected health information [PHI]).
- Ingesting different data sources (e.g., text documents) into Vertex AI for inference.

**Explore in the following courses:**

Introduction to AI and Machine Learning on Google Cloud (Module 2, Module 4)

Working with Notebooks in Vertex AI (Module 2)

Engineer Data for Predictive Modeling with BigQuery ML

Machine Learning Operations (MLOps) with Vertex AI: Manage Features

## 2.2: Model prototyping using Jupyter notebooks

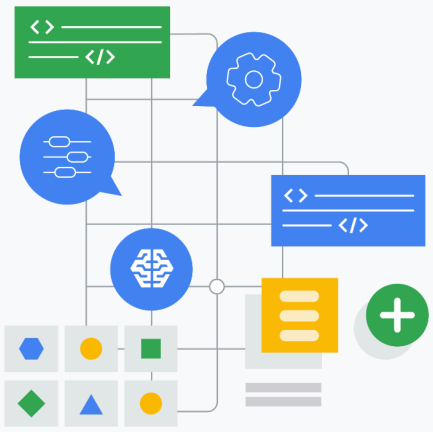
This section explores setting up and managing your machine learning development environment in Google Cloud. It explains the different Jupyter backend options, such as Vertex AI Workbench and Dataproc, and describes how to choose the best one for your needs. It covers essential security best practices in Vertex AI Workbench and Colab Enterprise to ensure your data and code remain protected. It also describes the advantages of using Spark kernels for large-scale data processing and how to integrate your notebooks with code repositories like Git for efficient version control and collaboration.

**Focus areas:**

- Choosing the appropriate Jupyter backend on Google Cloud (e.g., Vertex AI Workbench, notebooks on Dataproc).
- Applying security best practices in Vertex AI Workbench and Colab Enterprise.
- Using Spark kernels.
- Integration with code source repositories.
- Developing models in Vertex AI Workbench by using common frameworks (e.g., TensorFlow, PyTorch, Scikit-learn, Spark, JAX).
- Leveraging a variety of foundational and open-source models in Model Garden.

**Explore in the following courses:**

Working with Notebooks in Vertex AI
(Module 0, Module 1, Module 2)



## 2.3: Tracking and running ML experiments

This section focuses on building and evaluating machine learning models with a particular emphasis on generative AI. It explains how to select the optimal Google Cloud environment for your development and experimentation needs, choosing from options like Vertex AI Experiments, Kubeflow Pipelines, and Vertex AI TensorBoard. It delves into the nuances of evaluating generative AI solutions, considering factors like accuracy, creativity, bias, and ethical implications. It also gives practical experience integrating Vertex AI TensorBoard with popular frameworks like TensorFlow and PyTorch, enabling you to effectively visualize and analyze model performance, identify potential bottlenecks, and optimize your models for better results.

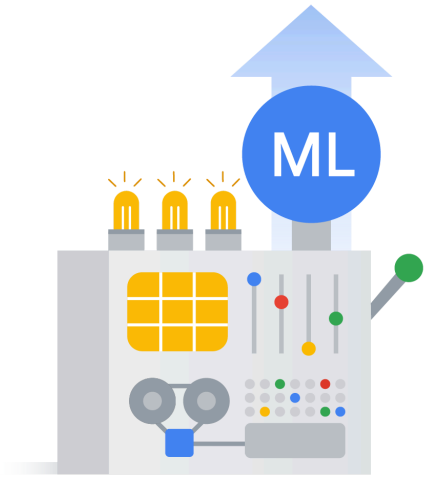
**Focus areas:**

- Choosing the appropriate Google Cloud environment for development and experimentation (e.g., Vertex AI Experiments, Kubeflow Pipelines, Vertex AI TensorBoard with TensorFlow and PyTorch) given the framework.
- Evaluating generative AI solutions.

**Explore in the following courses:**

Build and Deploy Machine Learning Solutions on Vertex AI (Lab 1)

Machine Learning Operations (MLOps) with Vertex AI: Model Evaluation (Module 2)



# Section 3: Scaling prototypes into ML models

## 3.1: Building models

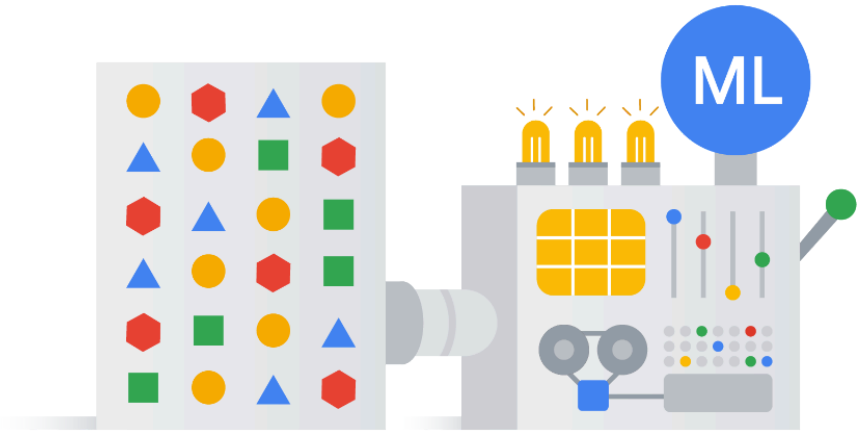
This section delves into the critical considerations for selecting the right tools and techniques for building interpretable machine learning models. It explains how to choose the most suitable ML framework for your project, considering factors like model development, training, and deployment. It also explores various modeling techniques and discusses how interpretability requirements can influence your choices, highlighting the trade-offs between model complexity and explainability.

**Focus areas:**

- Choosing ML framework and model architecture.
- Modeling techniques given interpretability requirements.

**Explore in the following courses:**

Production Machine Learning Systems (Module 2 and 3)



## 3.2: Training models

This section provides a comprehensive guide to training machine learning models on Google Cloud. It explains how to organize and ingest various data types for training, utilize different SDKs like Vertex AI and Kubeflow, and implement distributed training for reliable pipelines. It covers crucial aspects of the training process, including hyperparameter tuning and troubleshooting common training failures. Finally, it explores techniques for fine-tuning foundational models from Model Garden using Vertex AI, enabling you to leverage pre-trained models for your specific needs.

**Focus areas:**

- Organizing training data (e.g., tabular, text, speech, images, videos) on Google Cloud (e.g., Cloud Storage, BigQuery).
- Ingestion of various file types (e.g., CSV, JSON, images, Hadoop, databases) into training.
- Training using different SDKs (e.g., Vertex AI custom training, Kubeflow on Google Kubernetes Engine, AutoML, tabular workflows).
- Using distributed training to organize reliable pipelines.
- Hyperparameter tuning.
- Troubleshooting ML model training failures.
- Fine-tuning foundational models (e.g., Vertex AI, Model Garden).

**Explore in the following courses:**

Introduction to AI and Machine Learning on Google Cloud (Module 4)

Introduction to Large Language Models

Machine Learning Operations (MLOps) for Generative AI

Production Machine Learning Systems (Module 3, Module 6)

Build and Deploy Machine Learning Solutions on Vertex AI (Lab 3)

## 3.3: Choosing appropriate hardware for training

This section focuses on optimizing your model training process through strategic hardware and infrastructure choices. It describes the diverse compute and accelerator options available on Google Cloud, including CPUs, GPUs, TPUs, and edge devices, and how to select the best fit for your model's needs. It delves into distributed training techniques using TPUs and GPUs, exploring tools like Reduction Server on Vertex AI and Horovod. The section also provides a comparative analysis of GPUs and TPUs, helping you understand their trade-offs and make informed decisions based on your model architecture, computational demands, and budget constraints.

**Focus areas:**

- Evaluation of compute and accelerator options (e.g., CPU, GPU, TPU, edge devices).
- Distributed training with TPUs and GPUs (e.g., Reduction Server on Vertex AI, Horovod).

**Explore in the following courses:**

Production Machine Learning Systems (Module 3)

# Section 4: Serving and scaling models

## 4.1: Serving models

This section explores the process of deploying and managing machine learning models for inference. It explains batch and online inference methods, comparing their strengths and weaknesses, and how to choose the right Google Cloud service for your needs, including Vertex AI, Dataflow, BigQuery ML, and Dataproc. It examines factors to consider when selecting hardware for low-latency predictions and explores options for serving models built with different frameworks, like PyTorch and XGBoost. Finally, the section covers the importance of organizing models within a model registry for version control and streamlined deployment management.

**Focus areas:**

- Batch and online inference (e.g., Vertex AI, Dataflow, BigQuery ML, Dataproc).
- Using different frameworks (e.g., PyTorch, XGBoost) to serve models.
- Organizing a model registry.
- A/B testing different versions of a model.

**Explore in the following courses:**

Feature Engineering (Module 1)

Production Machine Learning Systems (Module 2)

Build and Deploy Machine Learning Solutions on Vertex AI (Lab 3)

## 4.2: Scaling online model serving

This section delves into optimizing the performance and scalability of your deployed machine learning models. It describes how to leverage Vertex AI Feature Store for efficient feature access during online prediction requests and how to choose between public and private endpoints for secure model serving. It explores strategies for scaling your serving backend to handle increased traffic, including Vertex AI Prediction and containerized serving. This section also covers selecting appropriate hardware for serving, considering factors like model complexity and latency requirements. Finally, it covers techniques for tuning your models to optimize performance in a production environment, focusing on aspects like simplification, latency reduction, and memory optimization.

**Focus areas:**

- Vertex AI Feature Store.
- Vertex AI public and private endpoints.
- Choosing appropriate hardware (e.g., CPU, GPU, TPU, edge).
- Scaling the serving backend based on the throughput (e.g., Vertex AI Prediction, containerized serving).
- Tuning ML models for training and serving in production (e.g., simplification techniques, optimizing the ML solution for increased performance, latency, memory, throughput).

**Explore in the following courses:**

Feature Engineering (Module 1)

Machine Learning Operations (MLOps) with Vertex AI: Manage Features

Production Machine Learning Systems (Module 2)

Build and Deploy Machine Learning Solutions on Vertex AI (Labs 1-3)

Create ML Models with BigQuery ML (Lab 2, Lab 3)

# Section 5: Automating and orchestrating ML pipelines

## 5.1: Developing end-to-end ML pipelines

This section explores building and managing robust machine learning pipelines on Google Cloud. It explains the crucial role of data and model validation in ensuring reliable ML solutions and how to maintain consistent data preprocessing between training and serving stages. It delves into different orchestration frameworks like Kubeflow Pipelines, Vertex AI Pipelines, and Cloud Composer, comparing their strengths and weaknesses. This section also examines the advantages and challenges of hybrid and multi-cloud strategies for ML pipelines, providing a comprehensive view of building and deploying ML solutions in diverse environments.

**Focus areas:**

- Data and model validation.
- Ensuring consistent data pre-processing between training and serving.
- Hosting third-party pipelines on Google Cloud (e.g., MLFlow).
- Identifying components, parameters, triggers, and compute needs (e.g., Cloud Build, Cloud Run).
- Orchestration framework (e.g., Kubeflow Pipelines, Vertex AI Pipelines, Cloud Composer).
- Hybrid or multi cloud strategies.
- System design with TFX components or Kubeflow DSL (e.g., Dataflow).

**Explore in the following courses:**

Introduction to AI and Machine Learning on Google Cloud (Module 4)

Machine Learning Operations (MLOps): Getting Started (Module 2)

Production Machine Learning Systems (Module 4)

Build and Deploy Machine Learning Solutions on Vertex AI (Lab 4)

## 5.2: Automating model retraining

This section focuses on automating the crucial process of retraining your machine learning models to maintain their accuracy and effectiveness over time. It explains how to establish a robust retraining policy, considering factors that influence retraining frequency and triggers. It also explores the advantages of implementing continuous integration and continuous delivery (CI/CD) pipelines for automated model deployment, ensuring a streamlined workflow for building, testing, and deploying updated models. This section will equip you with the knowledge to keep your ML models performing optimally in a dynamic environment through automation.

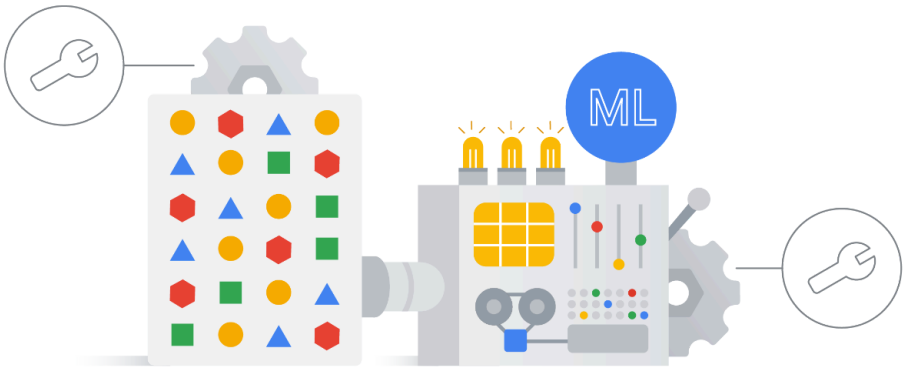
**Focus areas:**

- Determining an appropriate retraining policy.
- Continuous integration and continuous delivery (CI/CD) model deployment (e.g., Cloud Build, Jenkins).

**Explore in the following courses:**

Machine Learning Operations (MLOps) with Vertex AI: Manage Features (Module 0, Module 2, Module 3)

ML Pipelines on Google Cloud (Module 5)



## 5.3: Tracking and auditing metadata

This section focuses on tracking and auditing metadata within your machine learning pipelines for improved transparency and reproducibility. It describes how to track and compare model artifacts and versions using tools like Vertex AI Experiments and Vertex ML Metadata. It explores techniques for implementing model and dataset versioning, ensuring reproducibility and change tracking. Finally, this section covers the concept of model and data lineage, highlighting its importance in understanding and auditing ML pipelines, and how to effectively track lineage using Google Cloud tools.

**Focus areas:**

- Tracking and comparing model artifacts and versions (e.g., Vertex AI Experiments, Vertex ML Metadata).
- Hooking into model and dataset versioning.
- Model and data lineage.

**Explore in the following courses:**

Machine Learning Operations (MLOps): Getting Started (Module 2)

Machine Learning Operations (MLOps) with Vertex AI: Manage Features

# Section 6: Monitoring ML solutions

## 6.1: Identifying risks to ML solutions

This section focuses on identifying and mitigating potential risks associated with machine learning solutions. It covers security risks, including unintentional data exploitation and hacking, and how to address them through access control, encryption, and model hardening. It explores Google's Responsible AI practices, emphasizing fairness, privacy, transparency, and accountability in ML development. This section also covers identifying and mitigating biases in data, algorithms, and evaluations, and how to assess the overall readiness of your ML solution for production. Finally, it describes model explainability and how to leverage Vertex AI Explainable AI for gaining insights into model predictions and identifying potential biases or errors.

**Focus areas:**

- Building secure ML systems (e.g., protecting against unintentional exploitation of data or models, hacking).
- Aligning with Google's Responsible AI practices (e.g., biases).
- Assessing ML solution readiness (e.g., data bias, fairness).
- Model explainability on Vertex AI (e.g., Vertex AI Prediction).

**Explore in the following courses:**

Responsible AI for Developers: Privacy & Safety

Responsible AI for Developers: Fairness & Bias (Module 2)

Responsible AI for Developers: Interpretability & Transparency

Production Machine Learning Systems (Module 1, Module 2, Module 6)

## 6.2: Monitoring, testing, and troubleshooting ML solutions

This section focuses on monitoring, testing, and troubleshooting your deployed machine learning solutions to ensure ongoing performance and reliability. It explains how to establish continuous evaluation metrics using tools like Vertex AI Model Monitoring and Explainable AI to track model performance and identify potential issues. It explores concepts like training-serving skew and feature attribution drift, understanding their causes and mitigation strategies. This section also covers monitoring model performance against baselines and simpler models, and across time, to detect performance degradation or overfitting. Finally, it explores common training and serving errors and how to troubleshoot them effectively using various techniques, like log analysis and debugging tools.

**Focus areas:**

- Establishing continuous evaluation metrics (e.g., Vertex AI Model Monitoring, Explainable AI).
- Monitoring for training-serving skew.
- Monitoring for feature attribution drift.
- Monitoring model performance against baselines, simpler models, and across the time dimension.
- Common training and serving errors.

**Explore in the following courses:**

Build and Deploy Machine Learning Solutions on Vertex AI (Lab 1, Lab 3, Lab 5)

Production Machine Learning Systems (Module 2, Module 6)

TensorFlow on Google Cloud (Module 4)

# Glossary *

* This glossary provides a brief overview of key terms used in each section. For a comprehensive list of machine learning terminology with detailed explanations and examples, refer to the Google Machine Learning Glossary at developers.google.com/machine-learning/glossary.

## Section 1: Architecting low-code ML solutions

**AI (Artificial Intelligence)**: An umbrella term that includes anything related to computers mimicking human intelligence. Examples of AI applications include robots and self-driving cars.

**ML (Machine Learning)**: A subset of AI that allows computers to learn without being explicitly programmed, in contrast to traditional programming, where the computer is told explicitly what to do.

**AutoML:** Automated machine learning. AutoML aims to automate the process to develop and deploy an ML model.

**Batch prediction:** Making predictions without an endpoint.

**BigQuery ML (BQML):** BigQuery Machine Learning, allows users to use SQL (or Structured Query Language) to implement the model training and serving phases.

**Classification model:** A type of machine learning model that predicts a category from a fixed number of categories.

**Custom Training:** A code-based solution for building ML models that allows the user to code their own ML environment, giving them flexibility and control over the ML pipeline.

**Deep Learning packages:** A suite of preinstalled packages that include support for the TensorFlow and PyTorch frameworks.

**Pre-trained APIs:** Ready-to-use machine learning models requiring no training data.

**Transfer learning:** A technique where a pre-trained model is adapted for a new, related task.

**Hyperparameter:** A parameter whose value is set before the learning process begins.

**Large language models (LLM):** General-purpose language models that can be pre-trained and fine-tuned for specific purposes.

**Neural Architecture Search:** A technique for automating the design of artificial neural networks (ANNs).

**Deep Learning**: A subset of machine learning that adds layers in between input data and output results to make a machine learn at much depth.

**Generative AI**: Produces content and performs tasks based on requests. Generative AI relies on training extensive models like large language models, which are a type of deep learning model.

**Foundation model:** A large AI model trained on a massive dataset, capable of performing a wide range of tasks and serving as a basis for fine-tuning for specific applications.

**Fine-tuning:** The process of further training a pre-trained foundation model on a smaller, domain-specific dataset to adapt it to particular tasks or industries.

**Prompt design:** The process of crafting effective prompts to elicit desired responses from generative AI models.

**Supervised learning:** Uses labeled data to predict outcomes. Includes classification (predicting categories) and regression (predicting numbers).

**Unsupervised learning:** Uses unlabeled data to find patterns. Includes clustering (grouping data), association (finding relationships), and dimensionality reduction (simplifying data).

**MLOps (Machine Learning Operations)**: Turns ML experiments into production and helps deploy, monitor, and manage ML models.

**TPU (Tensor Processing Unit)**: Google's custom-developed application-specific integrated circuits (ASICs) used to accelerate machine learning workloads.

**TensorFlow Extended (TFX):** End-to-end platform for TensorFlow ML pipelines.

**Responsible AI**: The development and use of AI in a way that prioritizes ethical considerations, fairness, accountability, safety, and transparency.

## Section 2

**Accuracy:** A metric for evaluating classification models that measures the proportion of correct predictions out of the total number of predictions.

**Arbiter models:** Specialized language models used in model-based evaluation to mimic human evaluation and compare the quality of responses from different models.

**Bias and fairness:** Challenges in model evaluation related to inherent biases in data that can lead to discriminatory outcomes.

**Binary evaluation:** A simple evaluation type that involves a yes/no or pass/fail judgment.

**Categorical evaluation:** An evaluation type that offers more than two options for slightly more nuanced feedback.

**Continuous evaluation:** The ongoing process of evaluating a deployed model's performance using new data to identify any potential decline in effectiveness.

**Customization:** The ability to tailor workspaces to specific workflows, preferences, and project requirements.

**Diversity metrics:** Metrics that focus on measuring the variety and range of outputs a model can generate.

**Evaluation metrics:** Numerical scores used to quantify a model's performance on specific tasks.

**GPUs:** Graphical processing units that accelerate tasks like model training and inference, data analytics, and big data processing workloads by efficiently processing large datasets and performing parallel computations.

**Ground truth dataset:** A dataset containing the "correct" answers or labels, typically used as a reference for evaluating model performance.

**Jupyter Notebooks:** Interactive computing environments that allow users to create and share documents that contain live code, equations, visualizations, and narrative text.

**Model evaluation:** The process of assessing how well a machine learning model performs its intended task.

**Multi-task evaluation:** An evaluation type that combines multiple judgment types, such as numerical metrics and human ratings, for comprehensive evaluation.

**Numerical evaluation:** An evaluation type that assigns a quantitative score to model outputs.

**Notebook:** A document that combines executable code and rich text in a single document, used for editing and executing code.

**Persistent disk:** A type of storage that retains data even after a virtual machine (VM) instance is shut down or terminated. Persistent disks are typically used to store critical data that needs to be preserved, such as operating system files, application data, and user files.

**Principals:** Users, groups, domains, or service accounts that can be granted access to a resource in Colab Enterprise.

**Prompt engineering:** The process of designing and refining prompts or instructions given to LLMs to improve their performance and control their output.

**Roles:** Sets of permissions that determine what actions principals can take on a resource in Colab Enterprise.

**Structured data:** Data consisting of rows and columns.

**Text evaluation:** An evaluation type that uses human-generated feedback in the form of comments, critiques, or ratings to assess model output quality.

**Unstructured data:** Data such as pictures, audio, or video.

**Vertex AI Notebooks**: Jupyter-based fully managed, scalable, enterprise-ready compute infrastructure with easily enforceable policies and user management.

**Vertex AI Pipelines**: Automates the deployment process and ensures consistency between training and production environments.

## Section 3

**AI Foundations:** Building blocks for AI solutions, including cloud essentials, data tools, and AI development processes.

**Dialog tuning:** A specialized form of instruction tuning where language models are trained to engage in conversations by predicting the most appropriate response in a given context. Dialog-tuned models excel in handling back-and-forth interactions and are well-suited for chatbot applications.

**Fine tuning:** Adapting a pre-trained foundation model to a specific task or domain using a smaller dataset.

**Foundation models:** Large-scale, pre-trained AI models that can be adapted for various tasks, serving as a base for building specific applications.

**Hybrid machine learning models:** Models that work well both on-premises and in the cloud.

**Inference:** The process of using a trained machine learning model to make predictions on new data.

**Instruction tuning:** A training approach where language models are trained to follow instructions and generate responses based on the given input. Instruction-tuned models are more adaptable to various tasks and can perform well even with limited training data.

**Model Garden:** A repository of generative AI models, both from Google and open-source, that developers can access and utilize.

**Multimodal model:** A model capable of processing and generating content in multiple modalities, such as text, images, and video.

**Parameter-Efficient Tuning:** An approach to fine-tuning AI models that focuses on modifying a smaller subset of parameters, reducing computational requirements.

**Predictive AI:** AI models that focus on predicting future outcomes based on patterns in data.

**Transformer model:** Neural network architecture revolutionizing natural language processing.

## Section 4

**Batch serving:** Serving features for high throughput and serving large volumes of data for offline processing.

**Container:** An abstraction that packages applications and libraries together so that the applications can run on a greater variety of hardware and operating systems.

**Drift:** The change in an entity relative to a baseline. In the case of production ML models, this is the change between the real-time production data and a baseline data set, likely the training set, that is representative of the task the model is intended to perform.

**Entity:** A specific instance of an entity type.

**Entity type:** A group of features that are related to each other in some way.

**Entity view:** Contains the feature values that you requested.

**Extrapolation:** When models are asked to make predictions on points in feature space that are far away from the training data.

**Feature:** A measurable or recordable property of an entity, passed as input to a machine learning model.

**Feature ingestion:** The process of importing feature values computed by your feature engineering jobs into a feature store.

**Feature serving:** The process of exporting stored feature values for training or inference.

**Feature Store:** A centralized repository for organizing, storing, and serving machine learning features.

**Feature value:** Feature Store captures feature values for a feature at a specific point in time.

**Model Registry:** A tool for registering, organizing, tracking, and versioning trained and deployed ML models.

**Modular design:** Programs are more maintainable, as well as easier to reuse, test, and fix because they allow engineers to focus on small pieces of code rather than the entire program.

**Monolithic design:** A software program that is not modular. Few software programs use this design.

**Online serving:** Low-latency data retrieval of small batches of data for real-time processing, like for online predictions.

## Section 5

**AI Platform Pipelines:** A service that makes it easy to schedule Kubeflow pipeline runs, either one-off or recurring.

**AI Platform Training:** A service that allows users to submit training images directly to benefit from distributed training on the cloud.

**Cloud ML Engine:** A serverless execution environment that allows data scientists to not worry about infrastructure.

**Composability:** The ability to compose a bunch of microservices together, and the option to use what makes sense for your problem.

**Continuous training:** The practice of periodically retraining machine learning models to maintain their performance as data distributions change.

**Custom job:** A way to run your training code on Vertex AI. When creating a custom job, you specify settings such as the location, job name, Python package URIs, and worker pool specifications.

**Custom model:** A type of machine learning model that is created and trained by the user to meet specific needs.

**Federated Learning:** A new frontier in machine learning that continuously trains the model on device, and then combines model updates from a federation of users' devices to update the overall model.

**Inference-on-the-Edge:** Doing predictions on the edge, on the device itself, due to connectivity constraints.

**Kubeflow:** An open-source platform for building and deploying machine learning workflows, including pipelines.

**Kubeflow Pipelines:** A series of steps or operations in a machine learning workflow, often including data preprocessing, model training, and model deployment.

**ML Metadata:** Data about machine learning models and datasets, such as model architecture, training data, and evaluation metrics.

**model.py:** Contains the core machine learning logic for your training job, invoked by task.py.

**Parameter-efficient tuning:** A method for fine-tuning large language models that involves training only a subset of the model's parameters, reducing computational costs and resources.

**Pre-built ML APIs:** Pre-made machine learning models that are ready to use for common tasks, such as image recognition, natural language processing, and translation.

**Quantization:** A technique that compresses each float value to an 8-bit integer, reducing the size of the files and the resources required to handle the data.

**task.py:** The entry point to your code when sending training jobs to Vertex AI. It handles job-level details like parsing command-line arguments, run time, output locations, and hyperparameter tuning.

**Training pipeline:** A series of steps to train an ML model by using a dataset.

**Vertex AI Model Monitoring:** A service for monitoring the performance of machine learning models in production.

## Section 6

**Ablation analysis:** Comparing a model's performance with and without a specific feature to assess its value.

**Adversarial testing:** Evaluating AI models with malicious or harmful input to identify vulnerabilities.

**AI interpretability**: Understanding machine learning model behaviors.

**AI safety:** Building AI systems that are safe, secure, and beneficial to society, aligning with principles like fairness, accountability, and privacy.

**AI transparency**: Obtaining trust from stakeholders by making documentation for each system and communicating based on that information.

**Blocklists:** Lists of blocked words or phrases to prevent unsafe input.

**Bucketing:** A de-identification technique that generalizes a sensitive value by replacing it with a range of values. It is not reversible and does not maintain referential integrity.

**Cloud Data Loss Prevention (DLP) API:** A fully managed service designed to help discover, classify, and protect valuable data assets with ease. It provides de-identification, masking, tokenization, and bucketing, as well as the ability to measure re-identification risk, and sensitive data intelligence for security assessments.

**Cloud Key Management Service (KMS):** A cloud-hosted key management service that lets you manage encryption keys, allowing you to create, import, and manage cryptographic keys, and perform cryptographic operations in a single centralized cloud service.

**Constitutional AI:** Scaling supervision using AI to evaluate and tune models based on safety principles.

**Data Cards**: Structured summaries of essential facts about ML datasets.

**Data leakage:** When the target variable leaks into training data, leading to unrealistic model performance.

**Data security:** The protection of sensitive data used in AI systems, with the goal of minimizing the use of sensitive data through de-identification and randomization techniques.

**De-identification:** The process of removing or modifying personally identifiable information (PII) from data to protect individual privacy. Techniques include redaction, replacement, masking, tokenization, bucketing, and shifting.

**Differential privacy:** A rigorous approach that adds noise to data or model parameters to ensure that the inclusion or exclusion of any individual's data does not significantly affect the output or result of the analysis. Key parameters include the privacy parameter (epsilon) and sensitivity.

**Encryption:** The process of converting data into a secret code to prevent unauthorized access. Google Cloud provides default encryption at rest and in transit, as well as Cloud KMS for additional control over encryption keys.

**Federated Learning:** A distributed machine learning approach that trains models using decentralized data on devices like smartphones, preserving data privacy by avoiding the need to share raw data. It can be used for personalization, model updates, and federated analytics.

**Input/output safeguards:** Protective measures to ensure AI behavior aligns with safety and ethical standards.

**Instruction tuning and RLHF:** Fine-tuning language models using instructions and human feedback to embed safety concepts and align with human values.

**Intrinsic interpretability**: Intrinsic interpretability refers to models that are inherently transparent and can be understood without additional tools, such as linear regression models.

**Masking:** A de-identification technique that replaces some or all characters of a sensitive value with a surrogate value. It is not reversible and does not maintain referential integrity.

**Model agnostic**: Model agnostic methods can be applied to a wide range of machine learning models, as they analyze how changes in input features affect model output.

**Model Cards**: Short documents accompanying trained machine learning models that provide benchmarked evaluation.

**Model specific**: Model specific methods are restricted to specific types of machine learning models, relying on the internal details of the model.

**PII (Personally Identifiable Information):** Any information that can be used to identify an individual, such as full names, date of birth, address, phone number, and email address.

**Post-hoc interpretability**: Post-hoc interpretability refers to methods applied after a model is trained to provide insights into its behavior and explain its predictions.

**Redaction:** A de-identification technique that deletes all or parts of a sensitive value. It is not reversible and does not maintain referential integrity.

**Residuals:** The difference between the model's predictions and the actual target values.

**Safety classifiers:** Machine learning systems that classify input text as safe or unsafe.

**Training-serving skew:** Discrepancies between training and serving data distributions, potentially causing model performance issues.

# List of Google products and solutions

Learn more about Google Cloud products at cloud.google.com/products.

Learn more about Google Cloud solutions at cloud.google.com/solutions.

Learn more about reference architectures, design guidance, and best practices for building, migrating, and managing your cloud workloads at cloud.google.com/architecture/all-reference-architectures.

**App Engine:** A platform for building scalable web applications and mobile backends.

**BigQuery:** The primary data analytics tool on Google Cloud; a fully managed data warehouse that provides two services in one: storage plus analytics.

**Bigtable:** Scalable NoSQL solution for analytical workloads.

**Cloud Monitoring:** Collects and monitors metrics, events, and metadata from Google Cloud.

**Cloud Run:** A fully managed environment for running containerized apps.

**Cloud SQL:** Google Cloud's database service (relational database management service).

**Cloud Storage:** Google Cloud's object storage service for structured, semi-structured, and unstructured data.

**Cloud VMware Engine:** An engine for migrating and running VMware workloads natively on Google Cloud.

**Colab Enterprise:** A notebook solution appropriate for users who don't want to worry about managing compute, where they need Zero configuration and serverless infrastructure.

**Compute Engine:** Virtual machines running in Google's data center.

**Dataflow:** A fully managed streaming analytics service that creates a pipeline to process both streaming data and batch data.

**Dataproc:** A fully managed cloud service for running big data processing, analytics, and machine learning workloads on Google Cloud.

**Feature Store:** A managed service that simplifies machine learning feature management by storing and serving feature data directly from BigQuery.

**Firebase:** An app development software to build, improve, and grow mobile and web apps.

**Gemini:** Google's most recent foundation model, capable of handling multimodal data like text, images, and video.

**Google Kubernetes Engine:** An open source container orchestration system for automating computer application deployment, scaling, and management.

**JupyterLab:** A web-based interactive development environment for notebooks, code, and data.

**Model Garden:** A model library within Vertex AI that allows users to search, discover, and interact with a variety of generative AI models, including both Google and open-source models.

**Spanner:** A fully managed Google Cloud database service designed for global scale.

**TensorFlow:** An end-to-end open source platform for machine learning, with a comprehensive, flexible ecosystem of tools, libraries and community resources, originally created by Google.

**Vertex AI:** A unified platform for training, hosting and managing ML models. Features include AutoML and custom training.

**Vertex AI Studio:** An intuitive interface within Vertex AI that provides tools for experimenting with, tuning, and deploying generative AI models.

**Vertex AI Workbench:** A Jupyter notebook-based environment provided through virtual machine (VM) instances with features that support the entire data science workflow.