Google Cloud Certified Professional Machine Learning Language

Section 1: Architecting low-code ML solutions

1.1 Developing ML models by using BigQuery ML. Considerations include:

- 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 by using BigQuery ML
- Generating predictions by using BigQuery ML

1.2 Building AI solutions by using ML APIs. Considerations include:

- Building applications by using ML APIs (e.g., Cloud Vision API, Natural Language API, Cloud Speech API, Translation)
- Building applications by using industry-specific APIs (e.g., Document AI API, Retail API)

1.3 Training models by using AutoML. Considerations include:

- 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

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

2.1 Exploring and pre-processing organization-wide data (e.g., Cloud Storage, BigQuery, Spanner, Cloud SQL, Apache Spark, Apache Hadoop). Considerations include:

- Organizing different types of data (e.g., tabular, text, speech, images, videos) for efficient training
- Managing datasets in Vertex AI
- Data pre-processing (e.g., Dataflow, Tensor Flow 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])

2.2 Model prototyping using Jupyter notebooks. Considerations include:

- Choosing the appropriate Jupyter backend on Google Cloud (e.g., Vertex AI Workbench, notebooks on Dataproc)
- Applying security best practices in Vertex AI Workbench
- Using Spark kernels
- Integration with code source repositories
- Developing models in Vertex AI Workbench by using common frameworks (e.g., TensorFlow, PyTorch, sklearn, Spark, JAX)

2.3 Tracking and running ML experiments. Considerations include:

- Choosing the appropriate Google Cloud environment for development and experimentation (e.g., Vertex AI Experiments, Kube flow Pipelines, Vertex AI Tensor Board with TensorFlow and PyTorch) given the framework

Section 3: Scaling prototypes into ML models (~18% of the exam)

3.1 Building models. Considerations include:

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

3.2 Training models. Considerations include:

- 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
- Hyper parameter tuning
- Troubleshooting ML model training failures

3.3 Choosing appropriate hardware for training. Considerations include:

- 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)

Section 4: Serving and scaling models

4.1 Serving models. Considerations include:

- 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

4.2 Scaling online model serving. Considerations include:

- 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)

Section 5: Automating and orchestrating ML pipelines

5.1 Developing end-to-end ML pipelines. Considerations include:

- 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 multicloud strategies
- System design with TFX components or Kubeflow DSL (e.g., Dataflow)

5.2 Automating model retraining. Considerations include:

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

5.3 Tracking and auditing metadata. Considerations include:

- 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

Section 6: Monitoring ML solutions
6.1 Identifying risks to ML solutions. Considerations include:

- 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 explain ability on Vertex AI (e.g., Vertex AI Prediction)

6.2 Monitoring, testing, and troubleshooting ML solutions. Considerations include:

- 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