Google Professional

Machine Learning Engineer Practice Exam (PMLE)

(GOOG-PMLE-0010)

Cloud Certification Store

Google Professional Machine Learning Engineer

Cloud Certification Store

Google Cloud Professional Machine Learning Engineer - Practice Exam Questions (GOOG-PMLE-0010)

DISCLAIMER

- This practice test includes questions **compiled from various exam preparation platforms.**
- Important: **Questions and answers were AI-assisted and human-curated**. Verify accuracy with official Google documentation before relying on this material.
- **Users are strongly encouraged to** double-check all content against **official documentation and trusted sources** before using it for exam preparation or making important decisions.
- The creators of this material assume **no responsibility** for any errors, inaccuracies, or outcomes, including exam results, based on the use of this content.
- **Some questions might be duplicated or close** to previous ones, this is done on purpose as a way to re-inforce your learning.
- Single-user licence only
    - Includes one unique Payhip Licence Key per purchase, along with a Product Key.
    - Redistribution, resale, or public posting is prohibited. We can trace any file to the purchaser, with the use of the purchased License Key and Product Key.

Google Cloud Professional Machine Learning Enginner - Practice Questions (GOOG-PMLE-0010) - *PREVIEW 20 out of over 300 questions*

A Professional Machine Learning Engineer builds, evaluates, productionizes, and optimizes AI solutions by using Google Cloud capabilities and knowledge of conventional ML approaches. The ML Engineer handles large, complex datasets and creates repeatable, reusable code. The ML Engineer designs and operationalizes generative AI solutions based on foundational models. The ML Engineer considers responsible AI practices, and collaborates closely with other job roles to ensure the long-term success of AI-based applications. The ML Engineer has strong programming skills and experience with data platforms and distributed data processing tools. The ML Engineer is proficient in the areas of model architecture, data and ML pipeline creation, generative AI, and metrics interpretation. The ML Engineer is familiar with foundational concepts of MLOps, application development, infrastructure management, data engineering, and data governance. The ML Engineer enables teams across the organization to use AI solutions. By training, retraining, deploying, scheduling, monitoring, and improving models, the ML Engineer designs and creates scalable, performant solutions.

*Note: The exam does not directly assess coding skill. If you have a minimum proficiency in Python and Cloud SQL, you should be able to interpret any questions with code snippets.

The Professional Machine Learning Engineer exam assesses your ability to:

- ✔ Architect low-code AI solutions
- ✔ Collaborate within and across teams to manage data and models
- ✔ Scale prototypes into ML models
- ✔ Serve and scale models
- ✔ Automate and orchestrate ML pipelines
- ✔ Monitor AI solutions

This version of the Professional Machine Learning Engineer exam covers tasks related to generative AI, including building AI solutions using Model Garden and Vertex AI Agent Builder, and evaluating generative AI solutions.

To learn more about Google Cloud's generative AI services, go to Google Cloud Skills Boost to see the Introduction to Generative AI Learning Path (all audiences) or the Generative AI for Developers Learning Path (technical audience). If you are a partner, refer to the Gen AI partner courses: Introduction to Generative AI Learning Path , Generative AI for ML Engineers , and Generative AI for Developers. For additional learning, refer to product-specific Gen AI learning offerings, such as Explore and Evaluate Models using Model Garden , Vertex AI Agent Builder path (partners), and Integrate Search in Applications using Vertex AI Agent Builder .

About this certification exam

Length: Two hours

Registration fee: $200 (plus tax where applicable)

Language: English

Exam format: 50-60 multiple choice and multiple select questions

Exam delivery method:

a. Take the online-proctored exam from a remote location, review the online testing requirements.

b. Take the onsite-proctored exam at a testing center, locate a test center near you

Prerequisites: None

Recommended experience: 3+ years of industry experience including 1 or more years designing and managing solutions using Google Cloud.

Certification Renewal / Recertification: Candidates must recertify in order to maintain their certification status. Unless explicitly stated in the detailed exam descriptions, all Google Cloud certifications are valid for two years from the date of certification. Recertification is accomplished by retaking the exam during the recertification eligibility time period and achieving a passing score. You may attempt recertification starting 60 days prior to your certification expiration date.

Exam overview

Step 1: Get real world experience

Before attempting the Machine Learning Engineer exam, it's recommended that you have 3+ years of hands-on experience with Google Cloud products and solutions. Ready to start building? Explore the Google Cloud Free Tier for free usage (up to monthly limits) of select products.

Try the Google Cloud Free Tier

Step 2: Understand what's on the exam

The exam guide contains a complete list of topics that may be included on the exam. Review the exam guide to determine if your skills align with the topics on the exam.

See current exam guide

Step 3: Review the sample questions

Familiarize yourself with the format of questions and example content that may be covered on the Machine Learning Engineer exam.

Review sample questions

Step 4: Round out your skills with training

Collapse all

Follow the learning path

Prepare for the exam by following the Machine Learning Engineer learning path. Explore online training, in-person classes, hands-on labs, and other resources from Google Cloud.

Start preparing

Take a webinar

Prepare for the exam with Googlers and certified experts. Get valuable exam tips and tricks, as well as insights from industry experts.

Sign up

Additional resources

Explore Google Cloud documentation for in-depth discussions on the concepts and critical components of Google Cloud.

Learn about designing, training, building, deploying, and operationalizing secure ML applications on Google Cloud using the Official Google Cloud Certified Professional

Machine Learning Engineer Study Guide. This guide uses real-world scenarios to demonstrate how to use the Vertex AI platform and technologies such as TensorFlow, Kubeflow, and AutoML, as well as best practices on when to choose a pretrained or a custom model.

Step 5: Schedule an exam

Register and select the option to take the exam remotely or at a nearby testing center.

Review exam terms and conditions and data sharing policies.

Take the next step

Follow the learning path: Start Learning

Earn a skill badge in machine learning
Start now

New to Google Cloud?
Get started

Take a cert prep webinar
Watch Cloud OnAir

Practice Questions

Question 1

You built a custom ML model using scikit-learn. Training time is taking longer than expected. You decide to migrate your model to Vertex AI Training, and you want to improve the model's training time. What should you try out first?

A. Train your model in a distributed mode using multiple Compute Engine VMs.
B. Train your model using Vertex AI Training with CPUs.
C. Migrate your model to TensorFlow, and train it using Vertex AI Training.
D. Train your model using Vertex AI Training with GPUs.

✅ **Correct answer: B. Train your model using Vertex AI Training with CPUs**
📌 **Explanation:** Before making major changes to your model or infrastructure, start by running your scikit-learn training on Vertex AI's optimized environment using standard CPU instances. Vertex AI's deep learning VM images come with optimized math libraries (like Intel MKL) for NumPy/SciPy, which can accelerate scikit-learn computations.

This approach requires minimal code changes and leverages efficient vectorized operations, potentially speeding up training significantly. It's a simpler first step than rewriting your model in TensorFlow or implementing complex distributed training, and it lets you gauge performance improvements from a tuned environment. If training is still slow, you can then consider more involved options (like GPUs or distributed training).

Incorrect answers:

❌ **A. Train in distributed mode on multiple VMs** – Running scikit-learn algorithms across multiple machines is non-trivial. Scikit-learn doesn't natively distribute training across nodes, so this would add complexity and overhead with uncertain benefit. It's not the first thing to try when a simpler environment change (optimized single-node training) might suffice.

❌ **C. Migrate to TensorFlow and retrain** – Converting your scikit-learn model to TensorFlow would be time-consuming and isn't guaranteed to solve the speed issue. You'd only consider

this if you needed GPU acceleration or distributed training that scikit-learn can't handle. It's a heavy lift to rewrite the model, so it shouldn't be the first step.

❌ **D. Use GPUs on Vertex AI Training** – Many scikit-learn algorithms do not automatically benefit from GPU acceleration, as they run on CPU by design. Without specialized libraries, simply switching to a GPU instance might not improve training time for scikit-learn models. It's better to first optimize on CPUs. (If your algorithm can utilize GPU-enabled libraries, you might explore this later, but it's not the default assumption for scikit-learn.)

Question 2

You work for a gaming company that has millions of customers worldwide. All games offer a chat feature allowing real-time communication in over 20 languages, translated on-the-fly via the Cloud Translation API. You built an ML system to automatically moderate chat messages for toxicity. However, the model's performance varies greatly by language – it fails more often on certain languages. The factory has no reliable internet, and faster defect detection is a priority. (Your company wants to implement the new ML model ASAP.) How should you improve the model's performance across languages?

A. Add a regularization term (e.g., MinDiff) to the loss function to reduce bias.
B. Train a separate classifier using the chat messages in their original languages (no translation).
C. Replace the in-house word2vec embeddings with a large multilingual model like GPT-3 or T5.
D. Remove moderation for languages where the false positive rate is too high.

✅ **Correct answer: D. Remove moderation for languages for which the false positive rate is too high**

📌 **Explanation:** The simplest way to ensure uniform user experience when certain languages are yielding many moderation errors is to disable automated moderation for those languages. If the model is over-flagging benign content in underrepresented languages, turning off or limiting the model's actions on those languages prevents unfair blocking of users. This approach immediately eliminates the model's false positives for those languages, achieving uniform (if

conservative) treatment across all languages without requiring infrastructure changes or long development cycles. Given the urgent need and lack of time to retrain a better model, focusing moderation on languages where the model is reliable and omitting it where it's unreliable is a practical stop-gap solution.

Incorrect answers:

❌ **A. Add MinDiff regularization** – MinDiff is used to reduce *unfair bias* by penalizing performance gaps between predefined groups, but it requires defining those groups and retraining the modelruslanmv.com. This is a complex change and wouldn't quickly fix the immediate issue of poor performance on certain languages, especially if training data for those languages is limited.

❌ **B. Train on original languages** – Building separate models or a single multilingual model that processes messages in their original language could improve accuracy, but it's time and resource intensive. It requires either collecting labeled data for each language or using multilingual NLP techniques. The question scenario emphasizes implementing a solution *as soon as possible*, so retraining a new multi-language model is not the fastest remedy.

❌ **C. Use GPT-3 or T5 embeddings** – Swapping in a massive pretrained model (like GPT-3/T5) for embeddings could improve cross-language understanding, but those models are extremely resource-heavy. Using them would likely violate the "no infrastructure change" constraint and add significant latency or cost. Additionally, integrating such models is a non-trivial engineering effort. This option is neither quick nor cost-effective for an immediate fix.

Question 3

You need to ensure real-time ingestion of user activity data from a mobile app into BigQuery for analysis and ML experimentation. Your team will use BigQuery for data analysis and ML modeling. What should you do to ingest the streaming data into BigQuery with minimal latency?

A. Configure Pub/Sub to stream the data into BigQuery.
B. Run an Apache Spark streaming job on Dataproc to ingest the data into BigQuery.

C. Run a Dataflow streaming job to ingest the data into BigQuery.

D. Use Pub/Sub together with a Dataflow streaming job to ingest the data into BigQuery.

✅ **Correct answer: A. Configure Pub/Sub to stream the data into BigQuery**

📌 **Explanation:** The easiest and most efficient solution is to use **Pub/Sub's BigQuery subscription feature**, which writes messages directly from Pub/Sub into a BigQuery table. This native integration enables real-time streaming inserts with minimal setupinfoq.cominfoq.com. It eliminates the need to manage a separate processing job, providing low-latency ingestion out-of-the-box. By configuring a Pub/Sub topic for the app events and creating a BigQuery subscription on that topic, the data will flow continuously into BigQuery as it arrives, satisfying the real-time requirement.

Incorrect answers:

❌ **B. Use Spark on Dataproc** – Setting up a Spark streaming job introduces unnecessary complexity. You'd have to manage a Spark cluster and code the ingestion logic. This is heavyweight for simply piping events into BigQuery, and it likely adds more latency compared to Google's native streaming mechanisms.

❌ **C. Use Dataflow streaming** – While Dataflow can stream data to BigQuery, it's an extra layer here. If no transformation is needed on the data, using Dataflow is overkill for just shuttling data from Pub/Sub to BigQuery. The Pub/Sub→BigQuery direct path is simpler and managed for you, whereas Dataflow would require developing and maintaining a pipeline (and D is an even more explicit version of this approach).

❌ **D. Pub/Sub with Dataflow** – This classic combination (Pub/Sub as source, Dataflow pipeline to sink into BigQuery) is traditionally used for streaming ingestion, but Google's newer **BigQuery subscription** removes the need for the Dataflow step. Option D would work, but it's not the quickest or most efficient route. It incurs more cost and maintenance effort compared to simply letting Pub/Sub forward messages to BigQuery directly.

Question 4

You recently trained a deep learning model using Keras on a large dataset. After a few epochs, you notice the training and validation losses barely change – the model isn't learning. You want to debug and fix your model quickly. What should you do first?

A. Verify that your model can achieve a low loss on a small subset of the dataset.
B. Add handcrafted features to inject your domain knowledge into the model.
C. Use Vertex AI's hyperparameter tuning service to find a better learning rate.
D. Switch to hardware accelerators and train the model for more epochs.

✅ **Correct answer: A. Verify that your model can obtain a low loss on a small subset of the dataset**

📌 **Explanation:** When a model is not learning (loss not decreasing), a key first step is to **check for fundamental issues by training on a very small sample**. If your model can't overfit a tiny dataset, something is likely wrong with the model architecture, data processing, or training setup. By using a small subset (even just a few batches) and seeing if the model can drive loss near zero, you verify that the training pipeline is working at a basic level. This fast sanity check helps distinguish between a model capacity/optimization problem and a bug or data issue. It's a quick, low-effort diagnostic step to perform before trying more complex fixes.

Incorrect answers:

❌ **B. Add handcrafted features** – Introducing manual features is not the first line of action for a non-learning model. If the model isn't learning from existing features at all (loss plateaus immediately), adding features won't help until you resolve why learning stalled (which could be due to bugs, normalization issues, etc.). Handcrafted features are more relevant if the model is learning but not achieving required accuracy, not when it's basically stuck.

❌ **C. Hyperparameter tuning for learning rate** – While an improper learning rate can cause training issues (too high can cause oscillation, too low can stall learning), if losses are *barely changing at all*, it might indicate a more fundamental issue (like a bug). It's wise to first verify the model can learn in a simple scenario. Only after that would you systematically tune hyperparameters. Jumping straight to automated tuning without basic debugging could waste time.

❌ **D. Use accelerators and train longer** – If the model isn't improving, simply training for more epochs or on faster hardware won't magically fix it. Faster hardware (GPUs/TPUs) just does the same computations quicker; it doesn't address why loss is stagnant. Without diagnosing the root cause, you might just more quickly reach the same plateau. It's better to troubleshoot with short, controlled experiments (like option A) before scaling up compute.

Question 5

You are experimenting with an XGBoost classification model in Vertex AI Workbench. You split your BigQuery data into training and validation sets using these SQL queries:

pgsql

```
CREATE OR REPLACE TABLE training AS
(SELECT * FROM mytable WHERE RAND() <= 0.8);

CREATE OR REPLACE TABLE validation AS
(SELECT * FROM mytable WHERE RAND() <= 0.2);
```

After training, your model's AUC ROC is 0.80 on validation, but when deployed in production its AUC drops to 0.65. What is the most likely cause?

A. There is training-serving skew in your production environment.
B. The training dataset was too small to generalize well.
C. The training and validation tables share some records, meaning not all data was used properly.
D. The RAND() function caused every record in validation to also be in training.

✅ **Correct answer: A. There is training-serving skew in your production environment.**
📌 **Explanation:** The sudden performance drop from validation to production strongly suggests **training-serving skew**, meaning the model is seeing data in production that has a different

distribution or features than the data it was validated onfreecram.net. In this scenario, the method of splitting data is flawed: using separate `RAND()` filters can lead to overlapping datasets (many records appear in both training and validation)freecram.netfreecram.net. This overlap would make validation metrics overly optimistic (since the model inadvertently "saw" validation data during training), thus the high 0.80 AUC. Once deployed, the model faces truly unseen data and its performance (0.65 AUC) reflects the true generalization ability. This discrepancy – excellent metrics during validation vs poor in production – is a classic symptom of training-serving skew or data leakage.

Incorrect answers:

❌ **B. Training dataset insufficient** – If data volume were the issue, you'd likely see high variance (unstable metrics) or overfitting on training vs validation. Here the validation looked good, but production did not, indicating a shift in data rather than simply not enough data. Also, nothing in the scenario suggests an extremely small dataset; the issue is how the split was done.

❌ **C. Training and validation tables share some records** – This is true (they *do* share records given the RAND logic), but the option says "not using all data in initial table," which is a bit off. The bigger problem is the overlap itself causing misleading validation results, which is essentially part of training-serving skew. However, answer A names the broader issue more accurately. (Option C is on the right track but doesn't explicitly connect to the observed performance drop in production; the key issue is the skewed evaluation.)

❌ **D. Every validation record is in training** – It's an exaggeration to say *every* record overlaps (though theoretically any record with RAND() ≤ 0.2 would also satisfy ≤ 0.8). In practice, a large portion (roughly 20% of data) ended up in both setsfreecram.net. This data leakage is problematic, but stating it as "RAND caused every record to be in both" is not strictly accurate. The root cause is the splitting method leading to overlapping data, which falls under training-serving skew. Thus A is the more encompassing description of the problem.

Question 6

During batch training of a neural network, you notice the loss oscillates (fluctuates up and down) instead of steadily decreasing. What adjustment will most likely ensure the model converges?

A. Decrease the size of each training batch.
B. Decrease the learning rate hyperparameter.
C. Increase the learning rate hyperparameter.
D. Increase the size of the training batch.

✅ **Correct answer: B. Decrease the learning rate hyperparameter.**
📌 **Explanation:** Loss oscillation during training is a strong sign that the learning rate is too high, causing the optimizer to overshoot minima and bounce around the loss surfacefreecram.netfreecram.net. By lowering the learning rate, you make the weight updates smaller and more fine-grained, which helps the model descend the loss curve more smoothly rather than jumping back and forthfreecram.net. A smaller learning rate often stabilizes training and allows the network to converge to a minimum. This is a common remedy for oscillating or diverging losses in neural network trainingmattermodeling.stackexchange.com.

Incorrect answers:

❌ **A. Decrease batch size** – Reducing batch size increases the noisiness of the gradient estimates, which generally causes *more* oscillation in the loss, not less. A very small batch might even worsen convergence or require an even smaller learning rate. The primary cause of oscillation here is likely the step size (learning rate), not the batch size.

❌ **C. Increase learning rate** – This would exacerbate the problem. If the loss is already oscillating, a higher learning rate would cause even larger weight updates, potentially making the training diverge completely (loss blowing up). It's the opposite of the needed adjustment.

❌ **D. Increase batch size** – While larger batches give more stable gradients (reducing noise), simply using a bigger batch does not directly fix oscillation caused by an overly large learning rate. There's usually an optimal batch size for throughput, but convergence issues are more effectively addressed by tuning learning rate or using techniques like momentum/adaptive optimizers. In fact, if oscillation is severe, you'd still need to lower the learning rate even with a bigger batch.

Question 7

You are working on a predictive maintenance model for factory machines. It's a binary classifier that predicts whether a crucial machine will fail in the next 3 days (class "1" means failure is predicted). Only 4% of training examples are actual failures, so missing a failure is far more costly than a false alarm. You evaluate several models on a test set and want to choose the one that **prioritizes detection** of failures while ensuring that more than 50% of its failure predictions are correct. Which model performance criterion should you prioritize in selecting the model?

A. The model with the highest AUC ROC, given its precision is above 0.5.
B. The model with the lowest RMSE and recall above 0.5.
C. The model with the highest recall, with precision above 0.5.
D. The model with the highest precision, with recall above 0.5.

✅ **Correct answer: C. The model with the highest recall, with precision above 0.5.**
📌 **Explanation: Recall** (sensitivity) measures how many of the actual failures your model catches. "Prioritizing detection" means you want to catch as close to 100% of failures as possible, so high recall is the top priority. However, you also require that when the model does predict a failure, it's correct more than half the time – in other words, precision > 50%. Among models that meet the precision > 0.5 threshold, you should pick the one with the highest recall. This ensures you're maximizing the captured failures (fewer missed failures), while keeping false alarms to a manageable rate (at least half of alarms are true issues).

Incorrect answers:

❌ **A. Highest AUC with precision > 0.5** – AUC ROC is a useful overall metric, but it doesn't directly address the operating threshold or the trade-off between missing failures and raising false alarms. A model could have a great AUC yet still have subpar recall at the chosen threshold. The specific requirement here is about recall and precision, not the entire ROC curve. Focusing on AUC could lead to choosing a model that performs well on average but not optimally for catching failures.

❌ **B. Lowest RMSE with recall > 0.5** – RMSE (Root Mean Squared Error) is a regression metric and not applicable to classification performance. Including RMSE doesn't make sense in this context. We need classification metrics like precision/recall, not a regression error metric.

❌ **D. Highest precision with recall > 0.5** – This would choose a model that makes very few false alarms (high precision), as long as it at least catches half of failures (recall > 0.5). But the problem stated that missing a failure is very costly, implying recall is more critical. A model that is extremely precise might be too conservative and miss many failures (as long as it catches just over 50%, it passes the recall > 0.5 check). That's not desirable here – we'd rather tolerate more false alarms if it means catching nearly all failures. Thus, recall should be the primary focus.

Question 8

You have a highly imbalanced dataset: 96% of the images do **not** contain your company's logo, and only 4% do. You're training a binary image classifier to detect the presence of the logo. Which evaluation metric will give you the most confidence that the model is performing well?

A. Precision
B. Recall
C. RMSE
D. F₁ score

✅ **Correct answer: D. F₁ score**
📌 **Explanation:** The **F₁ score** is the harmonic mean of precision and recall, and it's well-suited for imbalanced classification problems where you care about both false positives and false negatives. In this scenario, if you rely on accuracy, a trivial model that always predicts "no logo" would be 96% accurate but useless. The F₁ score will only be high if the model achieves a good balance of precision and recall – meaning it finds a reasonable fraction of logos *and* doesn't flood you with false alarms. This gives a more informative picture of performance on the minority class than either precision or recall alone. Essentially, a strong F₁ indicates the model is handling the skewed dataset well by capturing logos without too many mistakes.

Incorrect answers:

❌ **A. Precision** – Precision alone tells you, "When the model says *logo present*, how often is it correct?" A high precision could be achieved by a model that rarely ever flags a logo (it might miss most logos but be right on the few it detects). In an imbalanced setting, focusing solely on precision might lead to a model that ignores many positive cases. It doesn't ensure logos are being detected adequately.

❌ **B. Recall** – Recall tells you "What percentage of actual logos did the model detect?" High recall alone could be achieved by flagging lots of images (catching most logos but also producing many false positives). By itself, recall doesn't account for precision. In practice, you want both: find the logos (recall) and be accurate when you flag one (precision). That's why F₁ is preferred – it combines both aspects.

❌ **C. RMSE** – Root Mean Squared Error is a regression metric, not used for evaluating classification performance. It doesn't apply here since the task is categorical (logo vs no logo). Even if one treats the problem as predicting probabilities, other metrics like log-loss or AUC would be more relevant than RMSE. In short, RMSE is the wrong tool for classification.

Question 9

You are creating an ML pipeline to ingest hundreds of millions of rows from BigQuery and train TensorFlow models on Vertex AI. You want to minimize data ingestion bottlenecks and ensure the solution is scalable. What is the best way to feed the BigQuery data into your TensorFlow training job?

A. Use the BigQuery Python client to download the data into a pandas DataFrame, then use `tf.data.Dataset.from_tensor_slices()` on it.

B. Export the BigQuery data to CSV files in Cloud Storage, then use `tf.data.TextLineDataset()` to read the CSVs.

C. Convert the BigQuery data to TFRecord format, store in Cloud Storage, then use `tf.data.TFRecordDataset()` to read it.

D. Use **TensorFlow I/O's BigQuery Reader** to read directly from BigQuery into the training pipeline.

✅ **Correct answer: D. Use TensorFlow I/O's BigQuery Reader to directly read the data.**

📌 **Explanation: TensorFlow I/O's BigQuery integration** allows you to stream data from BigQuery to TensorFlow efficiently, without intermediate storage. This is the most scalable and maintenance-free option for large datasets. It avoids the overhead of exporting or downloading data and leverages BigQuery's ability to supply data in parallel to your training job. In short, option D provides a direct, end-to-end pipeline with minimal bottlenecks – it was designed for exactly this purpose, enabling TensorFlow to consume BigQuery data in a distributed manner.

Incorrect answers:

❌ **A. BigQuery client to DataFrame, then from_tensor_slices** – Loading hundreds of millions of rows into a pandas DataFrame will likely run out of memory or be painfully slow. Even if it succeeded, wrapping a DataFrame with `from_tensor_slices` would force all data into memory and not leverage streaming or parallel prefetching. This approach doesn't scale to very large data.

❌ **B. Export to CSV and use TextLineDataset** – This introduces an extra step (exporting) and large CSV files, which are bulky to parse. Reading CSVs line by line is relatively slow compared to using a binary format. It also incurs additional storage and I/O overhead. It's a viable workaround for smaller datasets, but not the most efficient for massive data.

❌ **C. Use TFRecords via Cloud Storage** – TFRecord is a binary format that is efficient for TensorFlow, and using it would be scalable. However, the process of converting "hundreds of millions" of rows to TFRecords still means you have to run an export job (or Dataflow pipeline) to produce those records. That's an extra heavy step to manage. Option D skips the conversion and directly feeds data to training, simplifying the workflow. If you already had data in TFRecords, great – but here the data lives in BigQuery, so reading it directly is faster and easier than building a separate TFRecord export pipeline.

Question 10

You manage server maintenance at a data center and want to build a predictive maintenance model to detect machines likely to fail. You have lots of sensor data, but **no labeled incidents** (no examples of "failure" vs "normal" labeled yet). What is the first thing you should do?

A. Train a time-series model to predict each machine's performance metrics and trigger an alert when actual readings deviate significantly from predictions.
B. Develop a simple rule (e.g. a z-score threshold) to label historical data as "anomalous" or not, and use that rule in real-time to monitor machines.
C. Use a simple rule (e.g. z-score) on historical data to create labels, then train a supervised model on that newly labeled dataset to detect anomalies.
D. Hire experts to manually label the historical sensor data for past failures, then train a model on those labels.

✅ **Correct answer: A. Train a time-series model to predict the machines' performance values, and alert on significant deviations.**
📌 **Explanation:** With no labeled failure data, the best starting point is an **unsupervised or self-supervised anomaly detection approach**. Training a time-series forecasting model on each machine's normal behavior, then flagging large prediction errors, is a common technique. Essentially, the model learns the expected patterns (temperature, vibration, etc.), and if a machine's actual readings diverge sharply from the model's prediction, it likely indicates a potential issue. This approach doesn't require explicit failure labels and can start providing actionable insights immediately. It prioritizes catching any unusual behavior (potential failures) without needing a labeled dataset upfront.

Incorrect answers:

❌ **B. Heuristic rule for real-time monitoring** – Using a simple statistical threshold (like "if metric > 3σ from mean, flag it") might catch gross anomalies, but it's a very crude method. It doesn't learn the nuanced patterns of each machine or adapt over time. Also, by itself it doesn't improve your understanding of the data or create a model – it's just a one-off rule. It's often better to incorporate such heuristics into a model or use them to validate a model's alerts, rather than as the primary solution.

❌ **C. Label with heuristic then train a model** – This is effectively two steps: use a rule to generate pseudo-labels for anomalies, then train a supervised model on those labels. The issue is that if your initial heuristic is poor, your model will just learn that flawed heuristic. It's a bit circular. Given no true labels, it's usually more effective to directly use unsupervised anomaly detection (like option A) than to assume your made-up labels are ground truth. That said, once you have some confidence (or some actual failure examples), you could refine with supervised learning – but that's not the *first* thing to do.

❌ **D. Manually label historical data** – In predictive maintenance, true failures might be rare, and it may not be even possible for humans to correctly label "when did this sensor reading indicate a future failure?" without hindsight. Moreover, it's time-consuming and costly to have analysts label tons of sensor logs, and they might still miss subtle precursors to failure. It's better initially to use algorithms to detect anomalies. Human labeling could come into play later, once you have alerts, to confirm which alerts corresponded to real issues and then improve the model. But at the outset, it's impractical to label everything.

Question 11

You are experimenting with a built-in distributed XGBoost model in Vertex AI Workbench user-managed notebooks. You use BigQuery to split your data into training and validation sets using the following queries:

```
CREATE OR REPLACE TABLE ‘myproject.mydataset.training‘ AS
 (SELECT * FROM ‘myproject.mydataset.mytable‘ WHERE RAND() <= 0.8);

CREATE OR REPLACE TABLE ‘myproject.mydataset.validation‘ AS
 (SELECT * FROM ‘myproject.mydataset.mytable‘ WHERE RAND() <= 0.2);
```

After training the model, you achieve an area under the receiver operating characteristic curve (AUC ROC) value of 0.8, but after deploying the model to production, you notice that your model performance has dropped to an AUC ROC value of 0.65. What problem is most likely occurring?

A. There is training-serving skew in your production environment.
B. There is not a sufficient amount of training data.
C. The tables that you created to hold your training and validation records share some records, and you may not be using all the data in your initial table.
D. The RAND() function generated a number that is less than 0.2 in both instances, so every record in the validation table will also be in the training table.

Correct answer: A. There is training-serving skew in your production environment.

📌 AUC ROC degradation from 0.8 to 0.65 in production typically indicates that the data seen at serve time differs from the training distribution—i.e., training-serving skew Google - Professional M….

Incorrect answers:

❌ B. There is not a sufficient amount of training data. – New data would impact both training and production, not just serving.
❌ C. The tables … share some records … – That causes validation leakage, which would inflate validation performance, not degrade production relative to training.
❌ D. The RAND() function … – While possible, this would affect both training and validation splits, not cause skew between training and serving.

Question 12

Your team needs to build a model that predicts whether images contain a driver's license, passport, or credit card. The data engineering team already built the pipeline and generated a dataset composed of 10,000 images with driver's licenses, 1,000 images with passports, and 1,000 images with credit cards. You now have to train a model with the following label map: [`drivers_license`, `passport`, `credit_card`]. Which loss function should you use?

A. Categorical hinge
B. Binary cross-entropy

C. Categorical cross-entropy
D. Sparse categorical cross-entropy

✅ **Correct answer: C. Categorical cross-entropy**
📌 For a multiclass classification problem with three mutually exclusive classes, use categorical cross-entropy. It measures the difference between the true one-hot label distribution and the predicted probability distribution over all classes.

Incorrect answers:
❌ **A. Categorical hinge** – Used for "max-margin" loss in SVM-style multiclass; not standard for neural networks.
❌ **B. Binary cross-entropy** – Applies to independent binary labels (multi-label), not mutually exclusive multiclass.
❌ **D. Sparse categorical cross-entropy** – Also multiclass, but expects integer class IDs rather than one-hot encoded vectors. The question implies one-hot encoding via a label map.

Question 13

You are an ML engineer at a manufacturing company. You need to build a model that identifies defects in products based on images taken at the end of the assembly line. You want your model to preprocess the images with lower computation to quickly extract features of defects. Which approach should you use to build the model?

A. Reinforcement learning
B. Recommender system
C. Recurrent Neural Networks (RNN)
D. Convolutional Neural Networks (CNN)

✅ **Correct answer: D. Convolutional Neural Networks (CNN)**
📌 CNNs are purpose-built for image processing. Their convolutional layers extract spatial features efficiently, enabling fast inference on defects.

Incorrect answers:

❌ **A. Reinforcement learning** – For sequential decision making, not static image classification.

❌ **B. Recommender system** – For suggesting items or content, not image analysis.

❌ **C. RNN** – Designed for sequential data (text/time series), not spatial feature extraction in images.

Question 14

You are developing an ML model intended to classify whether X-ray images indicate bone fracture risk. You have trained a ResNet architecture on Vertex AI using a TPU accelerator, but you're unsatisfied with training time and memory usage. You want to quickly iterate your training code without major changes and minimize impact on accuracy. What should you do?

A. Reduce the number of layers in the model architecture.
B. Reduce the global batch size from 1024 to 256.
C. Reduce the dimensions of the images used in the model.
D. Configure your model to use bfloat16 instead of float32.

✅ **Correct answer: D. Configure your model to use bfloat16 instead of float32**

📌 Switching to bfloat16 halves memory usage and doubles throughput on TPUs with minimal code change. Model accuracy remains nearly unchanged because bfloat16 preserves dynamic range.

Incorrect answers:

❌ **A. Reduce layers** – Alters model capacity and may degrade accuracy; requires code refactoring.

❌ **B. Reduce batch size** – Eases memory pressure but slows overall throughput and convergence dynamics.

❌ **C. Reduce image dimensions** – May lose critical diagnostic details, risking accuracy.

Question 15

You have successfully deployed to production a complex TensorFlow model trained on tabular data. You want to predict the lifetime value (LTV) field for each subscription stored in the BigQuery table subscription.subscriptionPurchase in project my-fortune500-company-project. You organized your entire TFX pipeline (preprocessing → validation → training → deployment) as a Vertex AI pipeline. You need to prevent prediction drift—i.e., feature distributions changing significantly over time—without retraining too often. What should you do?

A. Implement continuous retraining of the model daily using Vertex AI Pipelines.
B. Add a model monitoring job where 10% of incoming predictions are sampled once every 24 hours.
C. Add a model monitoring job where 90% of incoming predictions are sampled once every 24 hours.
D. Add a model monitoring job where 10% of incoming predictions are sampled every hour.

✅ **Correct answer: D. Add a model monitoring job where 10% of incoming predictions are sampled every hour.**
📌 Hourly sampling at 10% balances cost and timeliness, enabling you to detect feature-drift quickly so you can trigger retraining only when necessary.

Incorrect answers:
❌ **A. Continuous daily retraining** – High cost and unnecessary if data distributions haven't shifted.
❌ **B. 10% sampled once per day** – May delay detection of drift by up to 24 hours, risking stale predictions.
❌ **C. 90% sampled daily** – High cost for little extra benefit; sampling rate can remain low if frequency is high.

Question 16

You recently developed a deep learning model using Keras, experimenting with training strategies. You first trained on a single GPU, but it was too slow. Next you distributed training

across 4 GPUs using `tf.distribute.MirroredStrategy` without other changes, but saw no speedup. What should you do?

A. Distribute the dataset with `tf.distribute.Strategy.experimental_distribute_dataset`
B. Create a custom training loop.
C. Use a TPU with `tf.distribute.TPUStrategy`
D. Increase the batch size.

✅ **Correct answer: A. Distribute the dataset with `tf.distribute.Strategy.experimental_distribute_dataset`**
📌 MirroredStrategy requires that you explicitly shard and distribute your input dataset. Calling `experimental_distribute_dataset` ensures each GPU gets a slice of each batch, enabling true parallelism without code structure changes.

Incorrect answers:
❌ **B. Custom training loop** – Unnecessary complexity; the standard Keras loop works once data is distributed correctly.
❌ **C. TPU with TPUStrategy** – A different accelerator; doesn't address why the GPUs weren't utilized.
❌ **D. Increase batch size** – Might help GPU utilization, but if the data isn't distributed, batch size has no effect on multi-GPU scaling.

Question 17

You work for a gaming company that develops massively multiplayer online (MMO) games. You built a TensorFlow model that predicts whether players will make in-app purchases of > $10 in the next two weeks. The model's predictions adapt each user's game experience. User data is stored in BigQuery. How should you serve your model while optimizing cost, user experience, and ease of management?

A. Import the model into BigQuery ML. Make predictions using batch reads from BigQuery, then push results to Cloud SQL.
B. Deploy the model to Vertex AI Prediction. Make predictions using batch reads from Cloud Bigtable, then push results to Cloud SQL.
C. Embed the model in the mobile application. Make predictions after each in-app purchase event is published to Pub/Sub, then push results to Cloud SQL.
D. Embed the model in a streaming Dataflow pipeline. Make predictions after each in-app purchase event is published to Pub/Sub, then push results to Cloud SQL.

✅ **Correct answer: D. Embed the model in a streaming Dataflow pipeline.**
📌 A Dataflow streaming pipeline can consume Pub/Sub events in real time, call the TensorFlow model for low-latency predictions, and write enriched results to Cloud SQL. It keeps infrastructure serverless and managed, providing scale and minimal client-side complexity.

Incorrect answers:
❌ **A. BigQuery ML batch** – Imposes high latency; not real time.
❌ **B. Vertex AI batch on Bigtable** – Also batch, not suitable for real-time adaptation.
❌ **C. Embedding in mobile app** – Exposes model code to clients, complicates updates and security.

Question 18

You are building a linear regression model on BigQuery ML to predict a customer's likelihood of purchasing your company's products. The model uses a city name variable as a key predictive component. To train and serve the model, data must be in columns. You want the least coding while preserving predictive power. What should you do?

A. Use TensorFlow to create a categorical variable with a vocabulary list. Create the vocabulary file, and upload it as part of your model to BigQuery ML.
B. Create a new BigQuery view that omits city information.
C. Use Cloud Data Fusion to assign each city to a region labeled 1–5, then use that number as the city feature.
D. Use Dataprep to one-hot encode the state column and make each city a binary column.

✅ **Correct answer: A. Use TensorFlow to create a categorical variable with a vocabulary list. Create the vocabulary file, and upload it as part of your model to BigQuery ML.**

📌 BigQuery ML supports supplying a GCS-hosted vocabulary file for STRING features. You define a CSV of all city names, upload it, and BigQuery ML automatically one-hot encodes using that vocabulary—no ETL pipelines or manual column explosion required.

Incorrect answers:

❌ **B. Drop city** – Loses the key predictive variable entirely.

❌ **C. Region bucketing** – Reduces granularity, likely harming predictive power by grouping distinct cities.

❌ **D. Dataprep one-hot** – Creates hundreds or thousands of columns (one per city), introducing schema bloat and requiring manual file export—not minimal coding.

Question 19

You are an ML engineer at a bank that has a mobile app. Management wants biometric authentication via fingerprint. Fingerprints are highly sensitive PII and cannot be downloaded or stored. Which learning strategy should you recommend to train and deploy this ML model?

A. Data Loss Prevention API
B. Federated learning
C. MD5 to encrypt data
D. Differential privacy

✅ **Correct answer: B. Federated learning**

📌 Federated learning keeps raw fingerprint data on-device while training a global model by aggregating weight updates. No sensitive data leaves the device, preserving privacy and meeting compliance without central storage of PII.

Incorrect answers:

❌ **A. DLP API** – For detecting/shielding existing PII, not training a model on-device data.

❌ **C. MD5 encryption** – Irreversible hash; useless for training biometric models.

❌ **D. Differential privacy** – Adds noise to protect individual records but still requires central data aggregation; doesn't keep data fully on-device.

Question 20

You are an ML engineer in the contact center of a large enterprise. You need to build a sentiment analysis tool that predicts customer sentiment from recorded phone conversations. You must ensure gender, age, and cultural differences do not bias any stage of development or results. What should you do?

A. Convert the speech to text and extract sentiment based on sentences.
B. Convert the speech to text and build a model based on the words.
C. Extract sentiment directly from the voice recordings.
D. Convert the speech to text and extract sentiment using syntactical analysis.

✅ **Correct answer: C. Extract sentiment directly from the voice recordings.**
📌 Voice-based sentiment analysis (tone, pitch, prosody) bypasses text biases due to colloquialisms, gender/age speech patterns, or cultural language differences. By analyzing acoustic features, you avoid textual biases in ASR and NLP pipelines.

Incorrect answers:
❌ **A. Sentence-level text sentiment** – Relies on transcripts; subject to transcription errors and linguistic bias.
❌ **B. Word-level text model** – Similarly biased by vocabulary and translation nuances across demographics.
❌ **D. Syntactic text analysis** – Focuses on grammar and structure, still vulnerable to textual biases in language use.

Final Review Checklist & Exam Readiness Scorecard tailored for your Practice Exam

Before you schedule your exam, use these tools to ensure you're truly ready.

🗒️ Final Review Checklist

Use this checklist to validate that you're ready across all key exam domains:

🧠 Framing ML Problems & Business Requirements

☐ Can map business challenges to ML problem types (classification, regression, etc.)
☐ Understand ethical ML practices, stakeholder goals, and data governance needs
☐ Familiar with responsible AI principles, fairness, and explainability

🧪 Data Preparation & Feature Engineering

☐ Can clean, normalize, transform, and split datasets correctly
☐ Understand feature importance, dimensionality reduction, and encoding techniques
☐ Know how to use BigQuery, Dataflow, and Vertex AI Feature Store

⚙️ Model Development

☐ Understand when to use AutoML, custom training, and pre-trained models
☐ Comfortable with scikit-learn, TensorFlow, XGBoost, and BigQuery ML
☐ Know how to evaluate models (precision, recall, AUC, F1, confusion matrix)

🔁 ML Pipelines & Automation (MLOps)

☐ Can design training pipelines using Vertex AI Pipelines or Kubeflow
☐ Understand how to use Cloud Build, Artifact Registry, and CI/CD for ML workflows
☐ Familiar with model versioning, retraining triggers, and continuous delivery

📦 Deployment & Monitoring

☐ Know how to deploy models to Vertex AI Prediction (online + batch)
☐ Understand A/B testing, canary deployments, and rollback strategies
☐ Can monitor performance drift, model bias, and service uptime

🔐 Security, Compliance, and Cost Optimization

☐ Familiar with IAM roles for ML workloads and VPC-SC usage
☐ Understand data encryption at rest/in transit, DLP, and audit logging
☐ Can estimate costs using Vertex AI and optimize training/deployment pipelines

---

📊 Exam Readiness Scorecard

Rate your confidence per domain and list anything that needs review:

Domain	Confidence Level (1-5)	Notes / Gaps to Review
Framing ML Problems	⭐⭐⭐⭐☆	Revisit business metric alignment
Data Preparation & Feature Engineering	⭐⭐⭐⭐⭐	Fully confident
Model Development	⭐⭐⭐⭐☆	Review evaluation metrics tradeoffs
ML Pipelines & Automation	⭐⭐⭐☆☆	Need to rewatch Kubeflow pipeline demos
Deployment & Monitoring	⭐⭐⭐⭐☆	A/B rollout strategy review needed
Security & Cost Optimization	⭐⭐⭐⭐☆	Slightly unsure about VPC-SC scenarios

You're exam-ready when you're consistently hitting **4+ stars across all domains** and scoring **85%+ on full-length timed practice exams**.

💫 **Congratulations!! You are on the right path to certification, you made it to 20 questions so far. You're my kind of audience! BUT… you need all the questions to pass.**

Note: ***This is a courtesy PREVIEW DOC of around 20 questions. The complete practice exam with over 300 questions can be purchased at our Cloud Certification Store at*** *https://cloudcertificationstore.com/b/Y1OyW*

We personally took this exam recently, and quite a few of us, plus many of our buyers that leave a review, assure you, we had more than 90% of the same questions in the recent exam (read the reviews).

So go ahead, invest in your future, and find the complete exam for this particular certification, or more practice exams at our Cloud Certification Store at ***https://cloudcertificationstore.com/collection/all***

2025 Google Cloud Digital Leader Practice Exam Questions (GOOG-CDL-0010)

$9.00

NEW! Google Cloud Certified Generative AI Leader Practice Exam Questions (GOOG-GAIL-0010)

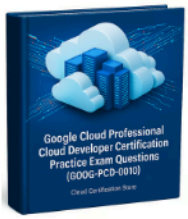
$9.00

2025 Google Cloud Associate Cloud Engineer Exam Questions (GOOG-ACE-0010)

$9.00

2025 Google Cloud Professional Cloud Architect Exam Questions (GOOG-PCA-0010)

$9.00



2025 Google Cloud Professional Cloud Developer Certification Practice Exam Questions (GOOG-PCD-0010)

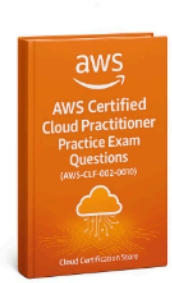
$9.00

2025 Google Cloud Professional Data Engineer Certification Practice Exam Questions (GOOG-PDE-0010)

$9.00

2025 Google Cloud Professional Cloud Database Engineer Practice Exam Questions (GOOG-PCDE-0010)

$9.00



2025 AWS Certified Cloud Practitioner Practice Exam Questions (AWS-CLF-002-0010)

$9.00

2025 Google Cloud Professional Machine Learning Engineer Practice Exam Questions (GOOG-PMLE-0010)

$9.00

NEW! 2025 AWS Certified Solutions Architect - Associate SAA-C0300 Practice Exam Questions (AWS-SAA-C0300-0010)

$9.00