databricks

Databricks Exam Guide

Databricks Certified Machine Learning Associate



[Provide Exam Guide Feedback]

Update Information

Updates	Date
Exam Published	March 1, 2025
Exam Guide Updated	November 10, 2026
New exam live in Webassessor (cutover date)	December 9, 2026
Current exam available through	December 8, 2026

Please check the exam guide on our webpages two weeks before your exam date to ensure you are preparing for the current version of the exam.

Which exam will you take? This guide covers two versions of this exam. The exam changes in Webassessor on **December 9, 2026**. Use the version that matches *your* scheduled exam date.

- If your exam is on or before **December 8, 2026**, prepare with the **Current Exam** content — the "Current exam" column in About the Exam, and the Current Exam outline.
- If your exam is on or after **December 9, 2026**, prepare with the **New Exam** content — the "New exam" column in About the Exam, and the New Exam outline.

Everything not split into two versions below — registration, recertification, recommended preparation, and documentation — applies to both.

Purpose of this Exam Guide

The purpose of this exam guide is to give you an overview of the exam and what it covers, to help you determine your exam readiness. This document is updated whenever there are changes to the exam (and indicates when those changes take effect) so that you can prepare with confidence. Because this exam is changing, this guide covers both the exam that is live now and the exam that goes live on the cutover date listed above. Identify your exam date, then prepare against the matching version. Please check back two weeks before your exam to make sure you have the most current version.

Audience Description

The Databricks Certified Machine Learning Associate certification exam assesses an individual's ability to use Databricks to perform basic machine learning tasks. This includes an ability to understand and use Databricks and its machine learning capabilities like Unity Catalog and select features of MLflow. It also assesses the ability to explore data and perform feature engineering. Additionally, the exam assesses model building through training, tuning, evaluation, interpretation, and selection. Finally, an ability to deploy and monitor machine learning models is assessed. Individuals who pass this certification exam can be expected to complete basic machine learning tasks using Databricks and its associated tools.

About the Exam

Item	Current exam (through December 8, 2026)	New exam (from December 9, 2026)
Number of scored items	48 scored multiple-choice and multiple-selection questions	45 scored multiple-choice and multiple-selection questions
Possible unscored items	Up to 10 unscored items may be included (see "Unscored content" below)	Up to 10 unscored items may be included (see "Unscored content" below)
Total number of questions you may see	48 scored questions plus unscored items	45 scored questions plus unscored items
Time limit	90 minutes (additional time is already factored in to account for unscored items)	90 minutes (additional time is already factored in to account for unscored items)
Registration fee	USD 200, plus applicable taxes as required per local law	USD 200, plus applicable taxes as required per local law
Delivery method	Online proctored or test center	Online proctored or test center
Languages offered	English; Japanese; Brazilian Portuguese (pt-BR); Korean	English; other languages to follow.
Test aides	None allowed	None allowed
Prerequisite	None required; but related training highly recommended	None required; but related training highly recommended
Recommended experience	Six months of hands-on experience in Databricks Platform with the listed exam outline are highly recommended	Six months of hands-on experience in Databricks Platform with the listed exam outline are highly recommended
Validity	2 years	2 years

Recertification: Recertification is required every two years to maintain your certified status. To recertify, you must take the current live exam.

Unscored content: Exams may include unscored items to gather statistical information for future use. These items are not identified on the exam and do not affect your score, and additional time is factored into the time limit to account for them.

Recommended Preparation

Training

- Instructor-led: Machine Learning with Databricks
- Self-paced (available in Databricks Academy):
    - Data Preparation for Machine Learning
    - Machine Learning Model Deployment
    - Machine Learning Model Development
    - Machine Learning Operations

AI Prep Guide - Any Databricks Exam

The Databricks AI Prep Guide shows you how to turn the AI chatbot you already use into a personalized tutor for any Databricks certification: prime it with the official exam guide, catch the outdated answers AI tends to give, find your weak spots, and pair it all with the hands-on practice every Databricks exam demands.

If you are preparing for the new exam, prime the chatbot with the New Exam outline in this guide, not the current one.

Hands-On Practice

Not sure what to practice? The AI Prep Guide (see the AI Prep Guide section above) includes a prompt under “Step 5 — Hands-on minimum” that has an AI chatbot generate a personalized checklist of 6–10 hands-on tasks mapped to this exam’s objectives, each small enough to complete in Databricks Free Edition.

Databricks Documentation

As you prepare, study the official Databricks documentation for the platform areas covered on this exam. It is the most current, authoritative source. Use it to judge how deeply to study each topic and to ensure you are learning up-to-date, non-deprecated features rather than out-of-date material.

Official documentation: Databricks documentation

Exam Outline — Current Exam (through December 8, 2026)

Section 1: Databricks Machine Learning (38%)

- Identify the best practices of an MLOps strategy
- Identify the advantages of using ML runtimes
- Identify how AutoML facilitates model/feature selection
- Identify the advantages AutoML brings to the model development process
- Identify the benefits of creating feature store tables at the account level in Unity Catalog in Databricks vs at the workspace level
- Create a feature store table in Unity Catalog
- Write data to a feature store table
- Train a model with features from a feature store table
- Score a model using features from a feature store table
- Describe the differences between online and offline feature tables
- Identify the best run using the MLflow Client API
- Manually log metrics, artifacts, and models in an MLflow Run
- Identify information available in the MLflow UI
- Register a model using the MLflow Client API in the Unity Catalog registry
- Identify benefits of registering models in the Unity Catalog registry over the workspace registry
- Identify scenarios where promoting code is preferred over promoting models and vice versa
- Set or remove a tag for a model
- Promote a challenger model to a champion model using aliases

Section 2: Data Processing (19%)

- Compute summary statistics on a Spark DataFrame using .summary() or dbutils data summaries
- Remove outliers from a Spark DataFrame based on standard deviation or IQR
- Create visualizations for categorical or continuous features
- Compare two categorical or two continuous features using the appropriate method
- Compare and contrast imputing missing values with the mean or median or mode value
- Impute missing values with the mode, mean, or median value
- Use one-hot encoding for categorical features
- Identify and explain the model types or data sets for which one-hot encoding is or is not appropriate
- Identify scenarios where log scale transformation is appropriate

Section 3: Model Development (31%)

- Use ML foundations to select the appropriate algorithm for a given model scenario
- Identify methods to mitigate data imbalance in training data
- Compare estimators and transformers
- Develop a training pipeline
- Use Hyperopt's fmin operation to tune a model's hyperparameters
- Perform random or grid search or Bayesian search as a method for tuning hyperparameters

- Parallelize single node models for hyperparameter tuning
- Describe the benefits and downsides of using cross-validation over a train-validation split
- Perform cross-validation as a part of model fitting
- Identify the number of models being trained in conjunction with a grid-search and cross-validation process
- Use common classification metrics: F1, Log Loss, ROC/AUC, etc
- Use common regression metrics: RMSE, MAE, R-squared, etc
- Choose the most appropriate metric for a given scenario objective
- Identify the need to exponentiate log-transformed variables before calculating evaluation metrics or interpreting predictions
- Assess the impact of model complexity and the bias variance tradeoff on model performance

Section 4: Model Deployment (12%)

- Identify the differences and advantages of model serving approaches: batch, realtime, and streaming
- Deploy a custom model to a model endpoint
- Use pandas to perform batch inference
- Identify how streaming inference is performed with Delta Live Tables
- Deploy and query a model for realtime inference
- Split data between endpoints for realtime inference

Total: 100%

Exam Outline — New Exam (from December 9, 2026)

Section 1: Databricks Machine Learning (36%)

- Identify the best practices of an MLOps strategy
- Identify the advantages of using ML runtimes
- Identify the benefits of creating feature store tables at the account level in Unity Catalog in Databricks vs at the workspace level
- Create a feature store table in Unity Catalog
- Write data to a feature store table
- Train a model with features from a feature store table
- Score a model using features from a feature store table
- Describe the differences between online and offline feature tables
- Identify the best run using the MLflow Client API
- Manually log metrics, artifacts, and models in an MLflow Run
- Identify information available in the MLflow UI
- Register a model using the MLflow Client API in the Unity Catalog registry
- Identify benefits of registering models in the Unity Catalog registry over the workspace registry
- Identify scenarios where promoting code is preferred over promoting models and vice versa
- Set or remove a tag for a model
- Promote a challenger model to a champion model using aliases
- Compare the performance of two MLflow runs to select a model for production

- Use Autologging to simplify the capture of performance metrics and hyperparameters

Section 2: Data Processing (20%)

- Compute summary statistics on a Spark DataFrame using .summary() or dbutils data summaries
- Remove outliers from a Spark DataFrame based on standard deviation or IQR
- Create visualizations for categorical or continuous features
- Compare two categorical or two continuous features using the appropriate method
- Compare and contrast imputing missing values with the mean or median or mode value
- Impute missing values with the mode, mean, or median value
- Use one-hot encoding for categorical features
- Identify and explain the model types or data sets for which one-hot encoding is or is not appropriate
- Identify scenarios where log scale transformation is appropriate

Section 3: Model Development (31%)

- Use ML foundations to select the appropriate algorithm for a given model scenario
- Identify methods to mitigate data imbalance in training data
- Compare estimators and transformers
- Develop a training pipeline
- Perform random or grid search or Bayesian search as a method for tuning hyperparameters
- Parallelize single node models for hyperparameter tuning
- Describe the benefits and downsides of using cross-validation over a train-validation split
- Perform cross-validation as a part of model fitting
- Identify the number of models being trained in conjunction with a grid-search and cross-validation process
- Use common classification metrics: F1, Log Loss, ROC/AUC, etc
- Use common regression metrics: RMSE, MAE, R-squared, etc
- Choose the most appropriate metric for a given scenario objective
- Identify the need to exponentiate log-transformed variables before calculating evaluation metrics or interpreting predictions
- Assess the impact of model complexity and the bias variance tradeoff on model performance
- Use Optuna to perform distributed hyperparameter tuning
- Assess whether a machine learning approach is appropriate for a business problem based on data availability, baseline performance, and ROI considerations
- Compare the computational requirements, training time, and data needs of deep learning versus traditional machine learning algorithms for a given problem
- Use SHAP values to explain individual model predictions and identify the most influential features for stakeholder communication
- Use feature importance scores to perform feature selection and reduce model complexity

Section 4: Model Deployment and Monitoring (13%)

- Identify the differences and advantages of model serving approaches: batch, realtime, and streaming

- Deploy a custom model to a model endpoint
- Use pandas to perform batch inference
- Identify how streaming inference is performed with Lakeflow pipelines
- Deploy and query a model for realtime inference
- Split data between endpoints for realtime inference
- Detect various drift types using Databricks Data profiling
- Create a custom metric to support a given ML modeling need
- Choose a deploy code or deploy model approach based on model size, training cost, and organizational complexity

Total: 100%

Sample Questions

The purpose of sample questions is to show you the objectives as they are stated in the exam guide and to give you a sample question that aligns with each objective. The exam guide lists the objectives that could be covered on an exam; the best way to prepare is to review the full exam outline for your exam date above.

The sample questions below align to the New Exam outline. If your exam is on or before **December 8, 2026**, these questions are still useful practice, but a few may cover objectives that are not on your exam. Your exam outline is the authoritative list of what you will be tested on.

Question 1

Objective: Create a feature store table in Unity Catalog.

A data scientist wants to create a feature table to use in their models. They are working in a workspace with Unity Catalog enabled and want this feature table to be stored and governed by it. What is the correct way of creating this feature table?

A. Create a Delta table with data in it, as usual, then use the **`register_table`** method from the **`FeatureStoreClient`** in Python to register it as a feature table in Unity Catalog.
B. Create an empty Delta table on Unity Catalog with the **`AS FEATURE STORE`** clause via SQL, then write data to it.
C. Use the **`create_table`** method of the **`FeatureEngineeringClient`** in Python to create the table, then write data to it.
D. Create a Delta table with data in it in Unity Catalog then use the **`ALTER TABLE`** command in SQL to configure it as a feature table with the **`SET AS FEATURE STORE`** clause.

Question 2

Objective: Impute missing values with the mode, mean, or median value.

A data scientist needs to impute the missing values in a continuous feature. They want to do this with the least amount of effort but with correct results. Which strategy will do this?

A. Use sklearn **`SimpleImputer`**, which automatically selects the best methodology based on the feature distribution.

B. Examine the distribution of the values and select the appropriate imputation upon review.
C. Use **`.mean()`**, which is the most appropriate imputation on continuous columns.
D. Use **`.mode()`**, which is the most appropriate imputation on continuous columns.

Question 3

Objective: Identify methods to mitigate data imbalance in training data.

A data scientist is developing a model that predicts whether a customer will churn from a subscription service. The dataset is highly imbalanced, with only 10% of the instances representing customers who churn. They want to ensure the model effectively identifies the minority class without being biased towards the majority class. Which strategy directly mitigates the model's bias towards the non-churn customers due to class imbalance?

A. Normalize the features to ensure they are on the same scale, improving model performance.
B. Use cost-sensitive learning by assigning a higher misclassification cost to the minority class during model training.
C. Increase the size of the training dataset by collecting more data on non-churn customers.
D. Use a simpler model to reduce overfitting, ensuring it generalizes better to the minority class.

Question 4

Objective: Identify the number of models being trained in conjunction with a grid-search and cross-validation process.

A data scientist is tuning a Support Vector Machine (SVM) model using 5-fold cross-validation and **`GridSearchCV`** in scikit-learn. The parameter grid includes three hyperparameters to optimize: `C` with values **`[0.1, 1, 10]`**, **`kernel`** with choices **`['linear', 'rbf']`**, and **`gamma`** with values **`[0.01, 0.1, 1]`**. How many different models will be trained in total?

A. 90
B. 18
C. 1
D. None of the above.

Question 5

Objective: Use Optuna to perform distributed hyperparameter tuning.

A ML engineer is building a distributed Optuna hyperparameter tuning study on Databricks to compare **`SVR`** and **`RandomForestRegressor`** for a regression task. The objective function must allow Optuna to select between the two model types on each trial so that both are explored across the parallel workers. The engineer is writing the objective function and needs to declare the model-selection step correctly before branching on the result. Which code line should the engineer use to declare the model-selection hyperparameter inside the objective function?

A. `regressor_name = trial.suggest_categorical('regressor', ['SVR', 'RandomForestRegressor'])`
B. `regressor_name = trial.suggest_float('regressor', ['SVR', 'RandomForestRegressor'])`

C. `regressor_name = trial.suggest_int('regressor', 0, 1)`
D. `regressor_name = trial.suggest_uniform('regressor', ['SVR', 'RandomForestRegressor'])`

Question 6

Objective: Promote a challenger model to a champion model using aliases.

A team registers successive versions of a model in the Unity Catalog model registry. After validation shows that a newly registered version outperforms the version currently serving production traffic, they want to mark the new version as the one that production reads, following the recommended approach. What should they do?

A. Use **`MlflowClient().transition_model_version_stage()`** to move the new version to the “Production” stage.
B. Use **`MlflowClient().set_registered_model_alias()`** to assign the “champion” alias to the new model version.
C. Delete the current champion model version so that the new version is the only one available to serve.
D. Use **`MlflowClient().set_model_version_tag()`** to set a “champion” = “true” tag on the new model version.

Question 7

Objective: Identify the best run using the MLflow Client API.

A data scientist has logged dozens of MLflow runs to a single experiment, each recording a **`validation_rmse`** metric. They want to programmatically retrieve the single run with the lowest **`validation_rmse`** using the MLflow Client API. Which approach accomplishes this?

A. Call **`search_runs`** with the experiment ID, **`order_by=["metrics.validation_rmse ASC"]`**, and take the first returned run.
B. Open the experiment in the MLflow UI, sort the runs table by **`validation_rmse`**, and read the top row.
C. List the runs with the client and select the run with the most recent start time.
D. Call **`get_run`** with the experiment name to return the experiment’s best run directly.

Question 8

Objective: Identify scenarios where log scale transformation is appropriate.

A data scientist is preparing features for a house-price regression model. One numeric feature and the price target are both strongly right-skewed, with a small number of very large values spanning several orders of magnitude. In which situation is applying a log-scale transformation most appropriate?

A. When a categorical feature has many distinct levels and its cardinality needs to be reduced.
B. When the feature is already approximately normally distributed and centered near zero.
C. When a feature or target is right-skewed and spans several orders of magnitude, to compress large values and make the distribution more symmetric.
D. When a feature contains negative values that need to be shifted into the positive range.

Question 9

Objective: Assess the impact of model complexity and the bias variance tradeoff on model performance.

During evaluation, a data scientist observes that their model achieves very low error on the training set but substantially higher error on the validation set. Based on the bias-variance tradeoff, what is happening and what is an appropriate response?

A. The model has high bias (underfitting); the scientist should increase model complexity.
B. The model has high bias caused by too few features; the scientist should engineer additional features.
C. Both training and validation error are high, indicating high variance; the scientist should train the model for longer.
D. The model has high variance (overfitting); the scientist should reduce model complexity or add regularization.

Question 10

Objective: Identify the differences and advantages of model serving approaches: batch, realtime, and streaming.

A retailer scores its entire customer base once each night to produce marketing segments for the next day. There is no requirement for low-latency, per-request predictions. Which model serving approach is the most appropriate and cost-effective for this workload?

A. Batch inference that scores all records on a nightly schedule.
B. Real-time inference behind a REST model serving endpoint queried once per customer.
C. Streaming inference using a continuously running Structured Streaming pipeline.
D. An always-on GPU model serving endpoint sized for the lowest possible latency.

Answer Key & Explanations

#	Answer	Explanation
1	C	In a Unity Catalog-enabled workspace, use the **`FeatureEngineeringClient.create_table`** method to create the feature table, then write data to it. There is no **`AS FEATURE STORE`** SQL clause or **`ALTER TABLE ... SET AS FEATURE STORE`**, and **`register_table`** is not the current creation path.
2	B	Correct results require examining the feature's distribution and choosing the appropriate imputation. **`SimpleImputer`** does not auto-select a strategy from the distribution, and blindly applying mean (or mode) can be inappropriate for a given continuous feature.
3	B	Cost-sensitive learning assigns a higher misclassification cost to the minority (churn) class, directly countering the model's bias toward the majority. Normalizing

		features, collecting more majority data, or using a simpler model do not address class imbalance.
4	A	The grid has 3 x 2 x 3 = 18 hyperparameter combinations, and 5-fold cross-validation trains a model per fold for each, so 18 x 5 = 90 models are trained in total.
5	A	**`trial.suggest_categorical`** is the correct Optuna API for selecting among a discrete set of named choices such as model types. It accepts a parameter name and a list of candidate values, allowing Optuna to sample one value per trial and enabling the objective function to branch on the result. **`trial.suggest_float`** and **`trial.suggest_uniform`** require numeric bounds, not a list of strings, so they raise a **`TypeError`** when passed a list. **`trial.suggest_int`** samples an integer index, not a model name string, so the downstream branching logic would receive **`0`** or **`1`** rather than a recognizable model identifier.
6	B	In the Unity Catalog model registry, the promotion workflow uses aliases rather than the legacy stage transitions. Assigning the “champion” alias to the new version with **`set_registered_model_alias`** makes it the version production reads. Stage transitions belong to the deprecated workspace registry, deleting the prior version is destructive and unnecessary, and a tag is descriptive metadata that does not change which version is served.
7	A	The MLflow Client’s **`search_runs`** (like **`mlflow.search_runs`**) supports server-side ordering, so ordering by **`metrics.validation_rmse`** ascending and taking the first result returns the best run programmatically. The MLflow UI is not the Client API, the most recently started run is not necessarily the best, and **`get_run`** requires a specific **`run_id`** and does not rank runs.
8	C	A log transformation compresses large values and pulls in a long right tail, making a right-skewed feature or target that spans several orders of magnitude more symmetric. It is not a cardinality-reduction technique, provides no benefit for an already-normal feature, and is undefined for zero or negative values, so it is not used to shift negatives into the positive range.
9	D	Low training error together with much higher validation error is the signature of high variance, or overfitting. The appropriate response is to reduce model complexity or add regularization (or gather more training data). High bias/underfitting would instead show high error on both the training and validation sets.
10	A	When predictions are generated on a fixed schedule over the full dataset with no low-latency requirement, batch inference is the simplest and most cost-effective approach. Real-time and always-on GPU endpoints add infrastructure and cost for latency the workload does not need, and streaming inference targets continuously arriving data rather than a nightly bulk scoring job.