databricks

Databricks Exam Guide

Databricks Certified Machine Learning Professional

Important note: This exam will change on September 30, 2025. See notes immediately below.

Provide Exam Guide Feedback

Purpose of this Exam Guide

The purpose of this exam guide is to give you an overview of the exam and what is covered on the exam to help you determine your exam readiness. This document will get updated anytime there are any changes to an exam (and when those changes will take effect on an exam) so that you can be prepared. **This version covers the currently live version up to and including September 29, 2025. Please see below for the exam guide for the exam starting on September 30, 2025.**

Audience Description

The Databricks Certified Machine Learning Professional certification exam assesses an individual's ability to use Databricks Machine Learning and its capabilities to perform advanced machine learning in production tasks. This includes the ability to track, version, and manage machine learning experiments and manage the machine learning model lifecycle. In addition, the certification exam assesses the ability to implement strategies for deploying machine learning models. Finally, test-takers will also be assessed on their ability to build monitoring solutions to detect data drift. Individuals to pass this certification exam can be expected to perform advanced machine learning engineering tasks using Databricks Machine Learning.

About the Exam

- Number of items: 59 multiple-choice questions
- Time limit: 120 minutes
- Registration fee: USD 200, plus applicable taxes as required per local law
- Delivery method: Online Proctored
- Test aides: none allowed
- Prerequisite: None required; course attendance and 1 year of hands-on experience in Databricks is highly recommended
- Validity: 2 years
- Recertification: Recertification is required every two years to maintain your certified status. To recertify, you must take the full exam that is currently live. Please review the "Getting Ready for the Exam" section on the exam webpage to prepare for taking the exam again.

- Unscored Content: Exams may include unscored items to gather statistical information for future use. These items are not identified on the form and do not impact your score, and additional time is factored into account for this content.

Recommended Training

- Instructor-led: Machine Learning at Scale and Advanced Machine Learning Operations
- Self-paced (available in Databricks Academy):
    - Machine Learning at Scale and Advanced Machine Learning Operations

Exam Outline

Section 1: Experimentation

Data Management
- Read and write a Delta table
- View Delta table history and load a previous version of a Delta table
- Create, overwrite, merge, and read Feature Store tables in machine learning workflows

Experiment Tracking
- Manually log parameters, models, and evaluation metrics using MLflow
- Programmatically access and use data, metadata, and models from MLflow experiments

Advanced Experiment Tracking
- Perform MLflow experiment tracking workflows using model signatures and input examples
- Identify the requirements for tracking nested runs
- Describe the process of enabling autologging, including with the use of Hyperopt
- Log and view artifacts like SHAP plots, custom visualizations, feature data, images, and metadata

Section 2: Model Lifecycle Management

Preprocessing Logic
- Describe an MLflow flavor and the benefits of using MLflow flavors
- Describe the advantages of using the pyfunc MLflow flavor
- Describe the process and benefits of including preprocessing logic and context in custom model classes and objects

Model Management
- Describe the basic purpose and user interactions with Model Registry
- Programmatically register a new model or new model version.

- Add metadata to a registered model and a registered model version
- Identify, compare, and contrast the available model stages
- Transition, archive, and delete model versions

Model Lifecycle Automation

- Identify the role of automated testing in ML CI/CD pipelines
- Describe how to automate the model lifecycle using Model Registry Webhooks and Databricks Jobs
- Identify advantages of using Job clusters over all-purpose clusters
- Describe how to create a Job that triggers when a model transitions between stages, given a scenario
- Describe how to connect a Webhook with a Job
- Identify which code block will trigger a shown webhook
- Identify a use case for HTTP webhooks and where the Webhook URL needs to come.
- Describe how to list all webhooks and how to delete a webhook

Section 3: Model Deployment

Batch

- Describe batch deployment as the appropriate use case for the vast majority of deployment use cases
- Identify how batch deployment computes predictions and saves them somewhere for later use
- Identify live serving benefits of querying precomputed batch predictions
- Identify less performant data storage as a solution for other use cases
- Load registered models with load_model
- Deploy a single-node model in parallel using spark_udf
- Identify z-ordering as a solution for reducing the amount of time to read predictions from a table
- Identify partitioning on a common column to speed up querying
- Describe the practical benefits of using the score_batch operation

Streaming

- Describe Structured Streaming as a common processing tool for ETL pipelines
- Identify structured streaming as a continuous inference solution on incoming data
- Describe why complex business logic must be handled in streaming deployments
- Identify that data can arrive out-of-order with structured streaming
- Identify continuous predictions in time-based prediction store as a scenario for streaming deployments
- Convert a batch deployment pipeline inference to a streaming deployment pipeline
- Convert a batch deployment pipeline writing to a streaming deployment pipeline

Real-time

- Describe the benefits of using real-time inference for a small number of records or when fast prediction computations are needed

- Identify JIT feature values as a need for real-time deployment
- Query a Model Serving enabled model in the Production stage and Staging stage
- Identify how cloud-provided RESTful services in containers is the best solution for production-grade real-time deployments

Section 4: Solution and Data Monitoring

Drift Types

- Compare and contrast label drift and feature drift
- Identify scenarios in which feature drift and/or label drift are likely to occur
- Describe concept drift and its impact on model efficacy

Drift Tests and Monitoring

- Describe summary statistic monitoring as a simple solution for numeric feature drift
- Describe mode, unique values, and missing values as simple solutions for categorical feature drift
- Describe tests as more robust monitoring solutions for numeric feature drift than simple summary statistics
- Describe tests as more robust monitoring solutions for categorical feature drift than simple summary statistics
- Compare and contrast Jenson-Shannon divergence and Kolmogorov-Smirnov tests for numerical drift detection
- Identify a scenario in which a chi-square test would be useful

Comprehensive Drift Solutions

- Describe a common workflow for measuring concept drift and feature drift
- Identify when retraining and deploying an updated model is a probable solution to drift
- Test whether the updated model performs better on the more recent data

Sample Questions

These questions are retired from a previous version of the exam. The purpose is to show you the objectives as they are stated on the exam guide, and give you a sample question that aligns with the objective. The exam guide lists all of the objectives that could be covered on an exam. The best way to prepare for a certification exam is to review the exam outline in the exam guide.

Question 1

Objective: Identify how to restore objects in a given scenario.

A data scientist has developed and logged a scikit-learn gradient boosting regressor model **`model`**, and then they ended their Spark session and terminated their cluster. After starting a new cluster, they want to review the **`estimators_`** of the original **`model`** object to analyze each of the trees in the trained model.

Which line of code can be used to restore the **`model`** object so that **`estimators_`** is available?

A. `mlflow.sklearn.load_model(model_uri)`
B. `client.pyfunc.load_model(model_uri)`
C. `mlflow.load_model(model_uri)`
D. `client.list_artifacts(run_id)["estimators.csv"]`

Question 2

Objective: Move models from None stage to Staging stage using MLflow Client

A machine learning engineer wants to move their model version **`model_version`** for the MLflow Model Registry model **`model`** from the None stage to the Staging stage using MLflow Client **`client`**.

Which code block should the engineer use to accomplish the task?

```
A.   client.transition_model_version_stage(
         name=model,
         version=model_version,
         stage="Staging"
     )

B.   client.transition_model_version_stage(
         name=model,
         version=model_version,
         stage="None"
     )

C.   client.transition_model_stage(
         name=model,
         version=model_version,
         stage="Staging"
     )
```

D.
```
        client.transition_model__stage(
            name=model,
            version=model_version,
            from="None",
            to="Staging"
        )
```

Question 3

Objective: Deploy a model by performing batch inference on a Spark DataFrame.

A machine learning engineer has developed a decision tree model using scikit-learn, logged the model using MLflow as **`decision_tree_model`**, and stored its run ID in the **`run_id`** Python variable. They now want to deploy that model by performing batch inference on a Spark DataFrame **`spark_df`**.

Which code block should the engineer use to create a function called **`predict`** that they can use to complete the task?

A.
```
predict = spark.spark_udf(
    f"runs:/{run_id}/decision_tree_model"
)
```

B.
```
predict = mlflow.pyfunc.spark_udf(
    spark_df,
    f"runs:/{run_id}/decision_tree_model"
)
```

C.
```
predict = sklearn.spark_udf(
    spark_df,
    f"runs:/{run_id}/decision_tree_model"
)
```

D.
```
predict = mlflow.pyfunc.spark_udf(
    spark,
    f"runs:/{run_id}/decision_tree_model"
)
```

Question 4

Objective: Identify types of drift in different scenarios.

A data scientist has developed a model to predict whether or not it will rain using the expected temperature and expected cloud coverage. However, the relationship between expected temperature and whether or not it rains has changed dramatically since the time period of the data on which the model was trained

Which type of drift is present in the above scenario?

A. Label drift
B. Feature drift
C. Concept drift
D. Prediction drift

Question 5
Objective: Identify how to implement elements of Feature Store.

A data scientist has suggested that their team start using Feature Store in Databricks Machine Learning. The data scientist claims that using Feature Store will meet a number of their feature management needs.

What will the team need to implement because it is not automatically provided by Feature Store?

A. Share features across workspaces
B. Measure the drift for individual features
C. Discover features used throughout the organization
D. Track where specific feature tables are used

Answers
Question 1: A
Question 2: A
Question 3: D
Question 4: C
Question 5: B

databricks

Databricks Exam Guide

Databricks Certified Machine Learning Professional

Purpose of this Exam Guide

This exam guide gives you an overview of the exam and what it covers to help you determine your exam readiness. This document will get updated anytime there are any changes to an exam (and when those changes will take effect on an exam) so that you can be prepared. **This version covers the currently live version as of September 30, 2025. Please check back two weeks before you take your exam to make sure you have the most current version.**

Audience Description

The Databricks Certified Machine Learning Professional certification exam assesses an individual's ability to design, implement, and manage enterprise-scale machine learning solutions using advanced Databricks platform capabilities. This includes proficiency in building scalable ML pipelines with SparkML, implementing distributed training and hyperparameter tuning, leveraging advanced MLflow features, and utilizing Feature Store concepts for automated feature pipelines. The certification exam evaluates expertise in MLOps practices, including testing strategies, environment management with Databricks Asset Bundles, automated retraining workflows, and monitoring using Lakehouse Monitoring for drift detection. Additionally, test-takers are assessed on their ability to implement deployment strategies, custom model serving, and model rollout management. Individuals who pass this certification exam can be expected to perform advanced machine learning engineering tasks at enterprise scale, implementing production-ready ML systems with comprehensive monitoring, testing, and deployment practices using the full feature set of Databricks.

About the Exam

- Number of items: 60 scored multiple-choice questions
- Time limit: 120 minutes
- Registration fee: USD 200, plus applicable taxes as required per local law
- Delivery method: Online Proctored
  Test aides: none allowed
- Prerequisite: None required; course attendance and 1 year of hands-on experience in Databricks is highly recommended
- Validity: 2 years
- Recertification: Recertification is required every two years to maintain your certified status.

- To recertify, you must take the full exam that is currently live. Please review the “Getting Ready for the Exam” section on the exam webpage to prepare for taking the exam again.
- Unscored Content: Exams may include unscored items to gather statistical information for future use. These items are not identified on the form and do not impact your score, and additional time is factored into account for this content

Recommended Preparation

- Instructor-led: Machine Learning at Scale and Advanced Machine Learning Operations
- Self-paced (available in Databricks Academy): Machine Learning at Scale and Advanced Machine Learning Operations
- Working knowledge of Python and major libraries that support machine learning like scikit-learn, SparkML, and MLflow
- Working knowledge of Lakehouse Monitoring and Databricks Model Serving
- Familiarity with the major topics in machine learning in Databricks documentation

Exam outline

Section 1: Model Development

Using Spark ML

- Identify when SparkML is recommended based on the data, model, and use case requirements.
- Construct an ML pipeline using SparkML.
- Apply the appropriate estimator and/or transformer given a use case.
- Tune a SparkML model using MLlib.
- Evaluate a SparkML model.
- Score a Spark ML model for a batch or streaming use case.
- Select SparkML model or single node model for an inference based on type: batch, real-time, streaming.

Scaling and Tuning

- Scale distributed training pipelines using SparkML and pandas Function APIs/UDFs.
- Perform distributed hyperparameter tuning using Optuna and integrate it with MLflow.
- Perform distributed hyperparameter tuning using Ray.
- Evaluate the trade-offs between vertical and horizontal scaling for machine learning workloads in Databricks environments.
- Evaluate and select appropriate parallelization (model parallelism, data parallelism) strategies for large-scale ML training.
- Compare Ray and Spark for distributing ML training workloads
- Use the Pandas Function API to parallelize group-specific model training and perform inference

Advanced MLflow Usage

- Utilize nested runs using MLflow for tracking complex experiments.
- Log custom metrics, parameters, and artifacts programmatically in MLflow to track advanced experimentation workflows.
- Create custom model objects using real-time feature engineering.

Advanced Feature Store Concepts

- Ensure point-in-time correctness in feature lookups to prevent data leakage during model training and inference.
- Build automated pipelines for feature computation using the FeatureEngineering Client
- Configure online tables for low-latency applications using Databricks SDK.
- Design scalable solutions for ingesting and processing streaming data to generate features in real time.
- Develop on-demand features using feature serving for consistent use across training and production environments.

Section 2: MLOps

Model Lifecycle Management

- Describe and implement the architecture components of model lifecycle pipelines used to manage environment transitions in the deploy code strategy.
- Map Databricks features to activities of the model lifecycle management process.

Validation Testing

- Implement unit tests for individual functions in Databricks notebooks to ensure they produce expected outputs when given specific inputs.
- Identify types of testing performed (unit and integration) in various environment stages (dev, test, prod, etc.).
- Design an integration test for machine learning systems that incorporates common pipelines: feature engineering, training, evaluation, deployment, and inference.
- Compare the benefits and challenges of approaches for organizing functions and unit tests.

Environment Architectures

- Design and implement scalable Databricks environments for machine learning projects using best practices.
- Define and configure Databricks ML assets using DABs (Databricks Asset Bundles): model serving endpoints, MLflow experiments, ML registered models.

Automated Retraining

- Implement automated retraining workflows that can be triggered by data drift detection or performance degradation alerts.
- Develop a strategy for selecting top-performing models during automated retraining.

Drift Detection and Lakehouse Monitoring

- Apply any statistical tests from the drift metrics table in Lakehouse Monitoring to detect drift in numerical and categorical data and evaluate the significance of observed changes.

- Identify the data table type and Lakehouse Monitoring feature that will resolve a use case need and explain why.
- Build a monitor for a snapshot, time series, or inference table using Lakehouse Monitoring.
- Identify the key components of common monitoring pipelines: logging, drift detection, model performance, model health, etc.
- Design and configure alerting mechanisms to notify stakeholders when drift metrics exceed predefined thresholds.
- Detect data drift by comparing current data distributions to a known baseline or between successive time windows.
- Evaluate model performance trends over time using an inference table.
- Define custom metrics in Lakehouse Monitoring metrics tables.
- Evaluate metrics based on different data granularities and feature slicing.
- Monitor endpoint health by tracking infrastructure metrics such as latency, request rate, error rate, CPU usage, and memory usage.

Section 3: Model Deployment
Deployment Strategies
- Compare deployment strategies (e.g. blue-green and canary) and evaluate their suitability for high-traffic applications.
- Implement a model rollout strategy using Databricks Model Serving.

Custom Model Serving
- Register a custom PyFunc model and log custom artifacts in Unity Catalog.
- Query custom models via REST API or MLflow Deployments SDK.
- Deploy custom model objects using MLflow deployments SDK, REST API or user interface.

Sample Questions

Sample questions will be made available here in this document at a later date.