databricks

Databricks Exam Guide

Databricks Certified Machine Learning Professional

Provide Exam Guide Feedback

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: 59 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

Question answers are available at the end of this document.

Question 1

Objective: Apply the appropriate estimator and/or transformer given a use case.

A Machine Learning Engineer wants to build a model to predict loan approval. Their dataset contains 430 million records with both numerical features (e.g., income, loan amount) and categorical features (e.g., employment type, education level). They need to encode categorical variables and combine all features for model training, ensuring the workflow scales as data volume increases.

Which approach meets these requirements?

A. Use SparkML's **`StringIndexer`** and **`OneHotEncoder`** to transform categorical features, then **`VectorAssembler`** to combine all features before model training.

B. Use pandas' **`get_dummies`** for categorical encoding and concatenate features manually before model training.

C. Use scikit-learn's **`LabelEncoder`** and **`OneHotEncoder`**, then combine features with **`ColumnTransformer`**.

D. Use TensorFlow's **`tf.feature_column`** API to preprocess and combine features.

Question 2

Objective: Perform distributed hyperparameter tuning using Optuna and integrate it with MLflow.

A Data Scientist wants to use Optuna for automated hyperparameter optimization. To speed up the process, the scientist wants to run the optimization in a distributed manner and each trial's outcome should be logged to an MLflow experiment. They want to do this with minimal set up.

Which approach meets these requirements?

A. Provide the MLflow Experiment ID to Optuna's **`MLflowCallback`** function, then use Python's async functionality to asynchronously trigger multiple parallel optimization runs.

B. Provide the Mlflow Experiment ID to **`MLflowStorage`** and initialize **`MLflowSparkStudy`** to distribute the trials across the multi-node Databricks cluster.

C. Provide the MLflow Experiment ID to Optuna's **`MLflowCallback`** function and then use **`MlflowSparkStudy`** to distribute the trials across the multi-node Databricks cluster.

D. Provide the MLflow Experiment path to Optuna's **`MLflowCallback`** function**,** then distribute the trials across a multi-node Ray on Spark cluster.

Question 3

Objective: Utilize nested runs using MLflow for tracking complex experiments.

A Data Scientist is performing hyperparameter tuning using k-fold cross-validation, followed by retraining the best model on the full training set for final evaluation. They want to use MLflow to:

- Track performance metrics and hyperparameters for each configuration
- Associate the final model with its hyperparameter search process
- Enable easy comparison in the MLflow UI

Which approach will structure their MLflow runs to meet their needs?

A. Use flat runs with tags to indicate hyperparameter sets and fold numbers, storing all metrics and models at the same level.

B. Use nested runs with a parent run for the entire experiment, child runs for each hyperparameter set (logging aggregated CV metrics), and the final model evaluation as a separate run of the best-performing parameter set.

C. Use nested runs with a parent run for each cross-validation fold and child run for the hyperparameter set best-performing parameter set.

D. Use nested runs with a parent run for the entire experiment and child runs only for the final model training, logging all cross-validation results as metrics in the parent run.

Question 4

Objective: Map Databricks features to activities of the model lifecycle management process.

A Machine Learning Engineer at an advertising agency needs to update their recommendation model. They want to continuously compare the new model's performance against the existing production model. However, since development will take several months, they're concerned that the production model might be updated during that time.

How should they approach this problem?

A. Use the MLflow Client's `get_latest_version()` method to get the most recent version of the registered model.

B. Use the Databricks Model Serving SDK to query the served model when validating against the new model.

C. Implement a shadow deployment strategy to test the new model with real world data.

D. Update model deployment code to add an alias to the model version in production and use the model alias when validating against the new model.

Question 5

Objective: Design an integration test for machine learning systems that incorporates common pipelines: feature engineering, training, evaluation, deployment, and inference.

A Machine Learning Engineer has updated their existing MLOps pipeline by changing their neural network model's hyperparameters. The current pipeline creates features and writes them to the Databricks Feature Store, which is then used to create a training set for their recommendation model. Then, they log this model using the Feature Engineering Client and serve it.

Which of these integration tests, at a minimum, will need to be re-run due to the change in the

hyperparameters after updating their code?

A. Only the training integration test will be affected since only model parameters are being updated, while feature engineering and deployment processes remain unchanged.

B. Both training and evaluation integration tests will be affected, as changing model parameters impacts model performance metrics, but feature engineering and deployment tests remain unaffected.

C. Training, evaluation, and deployment integration tests will be affected, as changing model parameters requires retraining, re-evaluation, and redeployment of the model, but feature engineering tests remain unaffected.

D. All integration tests (feature engineering, training, evaluation, and deployment) will be affected, as changing model parameters requires validation across the entire ML pipeline to ensure end-to-end functionality.

Question 6

Objective: Develop a strategy for selecting top-performing models during automated retraining.

A Machine Learning Engineer is working on a real time fraud classification model that outputs probability of fraud that gets converted to an automated action if a certain confidence threshold is met:

- If the model predicts a fraudulent transaction with 95% confidence or higher, the credit card is immediately frozen and the transaction is declined.
- If the model predicts the transaction is non-fraudulent with 95% confidence or higher, the transaction is allowed with no action taken.
- Any other scenario allows the transaction, but notifies the owner that suspicious activity was observed on their card.

Which loss metric should the engineer use to select the top performing model during model retrains?

A. F1

B. Log Loss

C. AUROC

D. Accuracy

Question 7

Objective: Compare deployment strategies, (eg blue-green and canary) and evaluate their

suitability for high-traffic applications.

A Machine Learning Engineer has deployed a model with a Model Serving Endpoint to detect potentially fraudulent charges in real-time. The model supports a high-traffic business critical process, which if it should begin to fail, will cost their company a significant sum of money. New models need to be rolled out in a safe manner that minimizes the financial risk should there be an issue with the new model. They need to configure the Model Serving Endpoint to support the current scaling requirements while also minimizing the risk of deploying a new model.

Which set of actions will meet the requirements?

A.
- Set the Compute scale-out to allow for horizontal scaling to meet traffic requirements.
- Ensure Route Optimization is enabled to further improve the endpoints throughput.
- When deploying a new model, leverage a second Served Entity and incrementally increase traffic to that entity while monitoring that entity's performance.

B.
- Set the Compute scale-out to allow for horizontal scaling to meet traffic requirements.
- When deploying a new model, leverage a second Served Entity and incrementally increase traffic to that entity while monitoring that entity's performance.

C.
- Set the Compute scale-out to allow for horizontal scaling to meet traffic requirements.
- Ensure Route Optimization is enabled to further improve the endpoints throughput.
- When deploying a new model, leverage a second Served Entity and set traffic to the new model at 100% and the original model at 0%.

D.
- Set the Compute scale-out to allow for horizontal scaling to meet traffic requirements.
- When deploying a new model, leverage a second Served Entity and set traffic to the new model at 100% and the original model at 0%.

Question 8

Objective: Define and configure Databricks ML assets using DABs (Databricks Asset Bundles): model serving endpoints, MLflow experiments, ML registered models.

A Machine Learning team needs to implement a CI/CD pipeline for their ML workflow that includes MLflow experiment tracking, Unity Catalog model registration, and model serving endpoints. The DevOps team requires that all infrastructure components be version-controlled, reproducible, and deployable across multiple environments (dev, staging, production) using a streamlined deployment process.

Which deployment strategy meets these requirements?

A. Use MLflow Projects to package the code and Databricks Jobs to orchestrate the deployment of each component separately.

B. Configure all ML resources (experiments, models, endpoints) in a Databricks Asset Bundle file and deploy.

C. Create Terraform modules for infrastructure provisioning and use the Databricks CLI for ML component configuration.

D. Use the Databricks CLI to deploy Python scripts as Job Runs.

Question 9

Objective: Query custom models via REST API or MLflow Deployments SDK.

A Machine Learning Engineer wants to integrate a customer churn prediction model deployed on Databricks Model Serving into their data processing pipeline. The pipeline is written in Python and needs to send requests directly to the model endpoint using an SDK-based approach that simplifies authentication and payload handling.

Which method should they use to send data to the model and retrieve predictions?

A. Use the **`ai_query`** function from the Databricks SDK with the model's serving endpoint and input parameters.

B. Call the **`predict`** method from the MLflow Deployments class with the endpoint name and model inputs.

C. Use the MLflow Tracking client to log the request data and then retrieve predictions from the model serving endpoint.

D. Send a POST request to the model's REST API endpoint using the **`requests`** library and pass the access token and input data as query string parameters.

Question 10

Objective: Apply the appropriate estimator and/or transformer given a use case.

A data scientist is building a model to estimate the total sales revenue for each upcoming quarter based on factors such as advertising spend, seasonality, and economic indicators.

Which estimator is appropriate for this prediction task?

A. Binary Logistic Regression

B. Multinomial Naive Bayes

C. Linear Regression

D. Softmax Classifier

Question Answers

Question 1: A
Question 2: B
Question 3: B
Question 4: D
Question 5: C
Question 6: B
Question 7: A
Question 8: B
Question 9: B
Question 10: C