databricks

Databricks Exam Guide

Databricks Certified Machine Learning Associate

BETA EXAM ONLY

Beta Note

The next beta test of this exam is currently scheduled for August 28 - September 11, 2024. You can apply for inclusion in the beta test with its one free exam attempt by filling out this form. before August 16, 2024. Note that participation is not guaranteed and dates and conditions are subject to change.

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 version covers the Beta Exam only and is subject to change at any time.**

Audience Description

The Databricks Certified Machine Learning Associate certification exam assesses an individual's ability to use Databricks to perform basic machine learning tasks to build and deploy performant predictive models. This includes an ability to understand and use Databricks features that support machine learning like AutoML, Feature Store, Unity Catalog and select capabilities of MLflow. It also assesses the basic machine learning skill set required for model development like data exploration, feature engineering, as well as model tuning, evaluation, and selection. Finally, it assesses the basics of model deployment. Individuals who pass this certification exam can be expected to complete basic machine learning tasks using Databricks and associated tools.

About the Exam

- Number of items: **up to 135 multiple-choice or multiple-selection questions**, including standard items and extra beta-test items **for the beta test only**
- Time Limit: 180 minutes, **for the beta test only**
- Registration fee: The single exam attempt for the beta test is free of charge Delivery method: Online Proctored
- Test aides: None allowed
- Prerequisite: None required; course attendance and six months of hands-on experience in Databricks is highly recommended. Also, see Recommended Preparation in this document. Validity: 2 years. **Beta Note: Results will not be immediately available at exam attempt end but beta testers who are successful in their will receive the full credential at**

project end. Results can take 4-6 weeks.

- 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.

Recommended Preparation

- Instructor-led: Machine Learning with Databricks
- Self-paced (available in Databricks Academy): Machine Learning with Databricks
- Working knowledge of Python and major libraries that support machine learning like scikit-learn and SparkML
- Working knowledge of Unity Catalog and other Databricks data management features like Delta Live Tables
- Familiarity with the major topics in machine learning in Databricks documentation

Exam outline

Section 1: Databricks Machine Learning

- 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

- 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

- Train a machine learning model
- Evaluate a machine learning model
- Compare estimators and transformers
- Develop a 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

Section 4: Model Deployment

- 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 interference