databricks

Databricks Exam Guide

Databricks Certified Machine Learning Professional

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 as of August 1, 2023

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: 60 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
- 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 in Production
- Self-paced (available in Databricks Academy): Machine Learning in Production

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
- 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
- Describe model serving deploys and endpoint for every stage
- Identify how model serving uses one all-purpose cluster for a model 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 objectives as they are stated on the exam guide, and give you a sample question that aligns to 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"]`
E. This can only be viewed in the MLflow Experiments UI

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 of the following code blocks can they 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"
     )
```

```
E.   client.transition_model_version_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 of the following code blocks can they 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"
)
```

E. It is not possible to deploy a scikit-learn model on a Spark DataFrame.

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
E. None of these

Question 5

Objective: Identify how to implement elements of Feature Store.

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

Which of the following 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
E. Monitor the freshness of feature tables

Answers

Question 1: C

Question 2: A

Question 3: D

Question 4: C

Question 5: B