databricks

Databricks Exam Guide

Databricks Certified Machine Learning Associate

Important note: This exam will change on October 28, 2024. See notes immediately below.

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 be 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 October 27, 2024. Please see below for the exam guide for the exam starting on October 28, 2024.**

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 Machine Learning and its capabilities like AutoML, Feature Store, and select capabilities of MLflow. It also assesses the ability to make correct decisions in machine learning workflows and implement those workflows using Spark ML. Finally, the ability to understand advanced characteristics of scaling 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

- Number of items: 45 multiple-choice questions
- Time limit: 90 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 six months 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. Additional time is factored into account for this content.

Recommended Training

- Instructor-led: Machine Learning with Databricks
- Self-paced (available in Databricks Academy): Machine Learning with Databricks

Exam outline

Section 1: Databricks Machine Learning

Databricks ML

- Identify when a standard cluster is preferred over a single-node cluster and vice versa
- Connect a repo from an external Git provider to Databricks repos.
- Commit changes from a Databricks Repo to an external Git provider.
- Create a new branch and commit changes to an external Git provider.
- Pull changes from an external Git provider back to a Databricks workspace.
- Orchestrate multi-task ML workflows using Databricks jobs.

Databricks Runtime for Machine Learning

- Create a cluster with the Databricks Runtime for Machine Learning.
- Install a Python library to be available to all notebooks that run on a cluster.

AutoML

- Identify the steps of the machine learning workflow completed by AutoML.
- Identify how to locate the source code for the best model produced by AutoML.
- Identify which evaluation metrics AutoML can use for regression problems.
- Identify the key attributes of the data set using the AutoML data exploration notebook.

Feature Store

- Describe the benefits of using Feature Store to store and access features for machine learning pipelines.
- Create a feature store table.
- 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.

Managed MLflow

- Identify the best run using the MLflow Client API.
- Manually log metrics, artifacts, and models in an MLflow Run.
- Create a nested Run for deeper Tracking organization.
- Locate the time a run was executed in the MLflow UI.
- Locate the code that was executed with a run in the MLflow UI.
- Register a model using the MLflow Client API.

- Transition a model's stage using the Model Registry UI page.
- Transition a model's stage using the MLflow Client API.
- Request to transition a model's stage using the ML Registry UI page.

Section 2: ML Workflows

Exploratory Data Analysis

- Compute summary statistics on a Spark DataFrame using .summary()
- Compute summary statistics on a Spark DataFrame using dbutils data summaries.
- Remove outliers from a Spark DataFrame that are beyond or less than a designated threshold.

Feature Engineering

- Identify why it is important to add indicator variables for missing values that have been imputed or replaced.
- Describe when replacing missing values with the mode value is an appropriate way to handle missing values.
- Compare and contrast imputing missing values with the mean value or median value.
- Impute missing values with the mean or median value.
- Describe the process of one-hot encoding categorical features.
- Describe why one-hot encoding categorical features can be inefficient for tree-based models.

Training

- Perform random search as a method for tuning hyperparameters.
- Describe the basics of Bayesian methods for tuning hyperparameters.
- Describe why parallelizing sequential/iterative models can be difficult.
- Understand the balance between compute resources and parallelization.
- Parallelize the tuning of hyperparameters using Hyperopt and SparkTrials.
- Identify the usage of SparkTrials as the tool that enables parallelization for tuning single-node models.

Evaluation and Selection

- Describe cross-validation and the benefits of 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.
- Describe Recall and F1 as evaluation metrics.

- Identify the need to exponentiate the RMSE when the log of the label variable is used.
- Identify that the RMSE has not been exponentiated when the log of the label variable is used.

Section 3: Spark ML

Distributed ML Concepts

- Describe some of the difficulties associated with distributing machine learning models.
- Identify Spark ML as a key library for distributing traditional machine learning work.
- Identify scikit-learn as a single-node solution relative to Spark ML.

Spark ML Modeling APIs

- Split data using Spark ML.
- Identify key gotchas when splitting distributed data using Spark ML.
- Train / evaluate a machine learning model using Spark ML.
- Describe Spark ML estimator and Spark ML transformer.
- Develop a Pipeline using Spark ML.
- Identify key gotchas when developing a Spark ML Pipeline.

Hyperopt

- Identify Hyperopt as a solution for parallelizing the tuning of single-node models.
- Identify Hyperopt as a solution for Bayesian hyperparameter inference for distributed models.
- Parallelize the tuning of hyperparameters for Spark ML models using Hyperopt and Trials.
- Identify the relationship between the number of trials and model accuracy.

Pandas API on Spark

- Describe key differences between Spark DataFrames and Pandas on Spark DataFrames.
- Identify the usage of an InternalFrame making Pandas API on Spark not quite as fast as native Spark.
- Identify Pandas API on Spark as a solution for scaling data pipelines without much refactoring.
- Convert data between a PySpark DataFrame and a Pandas on Spark DataFrame.
- Identify how to import and use the Pandas on Spark APIs.

Pandas UDFs/Function APIs

- Identify Apache Arrow as the key to Pandas <-> Spark conversions.
- Describe why iterator UDFs are preferred for large data.
- Apply a model in parallel using a Pandas UDF.
- Identify that pandas code can be used inside of a UDF function.
- Train / apply group-specific models using the Pandas Function API.

Section 4: Scaling ML Models

Model Distribution

- Describe how Spark scales linear regression.
- Describe how Spark scales decision trees.

Ensembling Distribution

- Describe the basic concepts of ensemble learning.
- Compare and contrast bagging, boosting, and stacking.

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 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: Create a new branch and commit changes to an external Git provider.

A data scientist is developing a machine learning model. They made changes to their code in a text editor on their local machine, committed them to the project's Git repository, and pushed the changes to an online Git provider. Now, they want to load those changes into Databricks. The Databricks workspace contains an out-of-date version of the Git repository.

How can the data scientist complete this task?

A. Open the Repo Git dialog and enable automatic syncing.
B. Open the Repo Git dialog and click the "Sync" button.
C. Open the Repo Git dialog and click the "Merge" button.
D. Open the Repo Git dialog and enable automatic pulling.
E. Open the Repo Git dialog and click the "Pull" button.

Question 2

Objective: Write data to a feature store table.

A data scientist has computed updated rows that contain new feature values for primary keys already stored in the Feature Store table **`features`**. The updated feature values are stored in the DataFrame **`features_df`**. They want to update the rows in **`features`** if the associated primary key is in **`features_df`**. If a row's primary key is not in **`features_df`**, they want the row to remain unchanged in **`features`**.

Which code block using the Feature Store Client **`fs`** can be used to accomplish this task?

```
A. fs.write_table(
  name="features",
  df=features_df,
  mode="merge"
  )

B. fs.write_table(
  name="features",
  df=features_df,
  mode="overwrite"
  )


C. fs.write_table(
  name="features",
  df=features_df,
  )

D. fs.create_table(
  name="features",
  df=features_df,
  mode="append"
  )

E. fs.refresh_table(
  name="features",
  df=features_df,
  mode="overwrite"
  )
```

Question 3

Objective: Programmatically transition a model's stage.

A senior machine learning engineer is developing a machine learning pipeline. They set up the pipeline to automatically transition a new version of a registered model to the Production stage in the Model Registry once it passes all tests using the MLflow Client API **`client`**.

Which operation was used to transition the model to the Production stage?

```
A. Client.update_model_stage
B. client.transition_model_version_stage
C. client.transition_model_version
D. client.update_model_version
```

Question 4

Objective: Make a Python library **`newpackage`** *available in a given scenario.*

A machine learning team wants to use the Python library **`newpackage`** on all of their projects. They share a cluster for all of their projects.

Which approach makes the Python library **`newpackage`** available to all notebooks run on a cluster?

A. Edit the cluster to use the Databricks Runtime for Machine Learning
B. Set the **`runtime-version`** variable in their Spark session to **`"ml"`**
C. Running **`%pip install newpackage`** once on any notebook attached to the cluster
D. Adding **`/databricks/python/bin/pip install newpackage`** to the cluster's bash init script
E. There is no way to make the **`newpackage`** library available on a cluster

Question 5

Objective: identify code blocks for computing the accuracy of a model.

A data scientist has developed a two-class decision tree classifier using Spark ML and computed the predictions in a Spark DataFrame **`preds_df`** with the following schema:

```
prediction DOUBLE
actual DOUBLE
```

Which of the following code blocks can be used to compute the accuracy of the model according

to the data in **`preds_df`** and assign it to the **`accuracy`** variable?

A.
```
accuracy = RegressionEvaluator(
    predictionCol="prediction",
    labelCol="actual",
    metricName="accuracy"
)
```

B.
```
accuracy = MulticlassClassificationEvaluator(
    predictionCol="prediction",
    labelCol="actual",
    metricName="accuracy"
)
accuracy = classification_evaluator.evaluate(preds_df)
```

C.
```
classification_evaluator = BinaryClassificationEvaluator(
    predictionCol="prediction",
    labelCol="actual",
    metricName="accuracy"
)
```

D.
```
accuracy = Summarizer(
    predictionCol="prediction",
    labelCol="actual",
    metricName="accuracy"
)
```

E.
```
classification_evaluator = BinaryClassificationEvaluator(
    predictionCol="prediction",
    labelCol="actual",
    metricName="accuracy"
)
accuracy = classification_evaluator.evaluate(preds_df)
```

Answers:

Question 1: E

Question 2: A

Question 3: B

Question 4: D

Question 5: E

databricks

Databricks Exam Guide

Databricks Certified Machine Learning Associate

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 October 28, 2024. 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 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 AutoML, 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, and evaluation and selection. Finally, an ability to deploy 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

- Number of items: 48 scored multiple-choice or multiple-selection questions
- Time Limit: 90 minutes
- Registration fee: $200
- Delivery method: Online Proctored
- Test aides: None allowed
- Prerequisite: None required; course attendance and six months of hands-on experience performing the tasks mentioned in the below Exam Outline is highly recommended. Also, see Recommended Preparation in this document.
- 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. Additional time is factored into account for this content.

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

- 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

- 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

Sample Questions

These questions are similar to actual question items and give you a general sense of how questions are asked on this exam. They include exam 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: 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 working on a machine learning project to develop 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 that your 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: Identify how streaming inference is performed with Delta Live Tables.

A company has a podcast platform that has thousands of users. The company has implemented an anomaly detection algorithm to detect low podcast engagement based on a 10-minute running window of user events such as listening, pausing, and exiting the podcast. A machine learning engineer wants to deploy this model into a production data pipeline that needs to handle up to tens of thousands of events per second. As the volume of events fluctuates throughout the day, the engineer needs the pipeline compute to be resized dynamically.

Which pipeline design approach meets these requirements?

A. Create a Delta Live Tables pipeline that applies the algorithm as a Spark UDF.
B. Create a Structured Streaming Job that applies the algorithm as a Spark UDF.
C. Create a model serving endpoint, create a Delta Live Tables pipeline that calls a custom UDF which invokes the endpoint.
D. Create a model serving endpoint, create a Structured Streaming job that calls a custom UDF which invokes the endpoint.

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