DATABRICKS Machine Learning Associate Exam Questions

Total Questions: 70+
Demo Questions: 15
Version: Updated for 2025

Prepared and Verified by Cert Empire – Your Trusted IT Certification Partner

For Access to the full set of Updated Questions – Visit:
Databricks Machine Learning Associate Exam Questions by Cert Empire

Question: 1

Which of the following machine learning algorithms typically uses bagging?

A. Gradient boosted trees

B. K-means

C. Random forest

D. Linear regression

E. Decision tree

Answer:

C

Explanation:

Random Forest is an ensemble learning algorithm that operates by constructing a multitude of decision trees at training time. It is fundamentally based on the principle of bootstrap aggregating, or "bagging." In this process, each individual decision tree in the forest is trained on a different random subsample of the training data, drawn with replacement (bootstrapping). The final prediction is determined by aggregating the results of all individual trees-typically by majority vote for classification or by averaging for regression. This technique helps to decrease the model's variance and reduce overfitting, leading to improved accuracy and stability.

Why Incorrect Options are Wrong:

A. Gradient boosted trees: This algorithm uses boosting, a sequential process where each new model corrects the errors of its predecessor, not the parallel, independent training characteristic of bagging.
B. K-means: This is an unsupervised clustering algorithm used for partitioning data into groups. It does not use ensemble methods like bagging, which are typical for supervised learning tasks.
D. Linear regression: This is a single predictive model. While bagging can be applied to it, it is not an inherent or typical part of the standard linear regression algorithm.
E. Decision tree: This is a single model and serves as the base learner for a Random Forest. Bagging is a technique for combining multiple models, not a feature of a single decision tree.

References:

1. Databricks Official Documentation, "Ensemble tree methods": "Random forests combine many decision trees. To prevent the trees from being highly correlated with each other, random forests train each tree on a different sample of the data (using bootstrapping)..." This directly confirms that Random Forests use bootstrapping, the core of bagging.
2. Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32. DOI:

https://doi.org/10.1023/A:1010933404324: In the introduction (Section 1), the paper establishes that Random Forests are a method that adds a layer of randomness to "bagging," which involves growing each tree on a bootstrap sample of the training data.
3. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer. In Chapter 15, "Random Forests," Section 15.1 states: "Random forests is a substantial modification of bagging that builds a large collection of de-correlated trees, and then averages them." This explicitly links the algorithm to bagging.

Question: 2

The implementation of linear regression in Spark ML first attempts to solve the linear regression problem using matrix decomposition, but this method does not scale well to large datasets with a large number of variables. Which of the following approaches does Spark ML use to distribute the training of a linear regression model for large data?

A. Logistic regression

B. Spark ML cannot distribute linear regression training

C. Iterative optimization

D. Least-squares method

E. Singular value decomposition

Answer:

C

Explanation:

For large-scale datasets, solving the linear regression problem with a direct analytical method like the normal equation (which involves matrix decomposition) is computationally infeasible. The XT X matrix can become too large to fit in memory or too expensive to invert.

To overcome this, Spark ML's LinearRegression implementation utilizes iterative optimization algorithms. These methods, such as L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno), start with an initial guess for the model's weights and repeatedly update them in a direction that minimizes the loss function. The core computations, like calculating the gradient, can be efficiently distributed across the cluster, making this approach highly scalable for large data.

Why Incorrect Options are Wrong:

A. Logistic regression is a distinct algorithm used for classification tasks, not a method for training a linear regression model.
B. This is false. The primary purpose of Spark ML is to provide scalable, distributed implementations of machine learning algorithms, including linear regression.
D. The least-squares method describes the objective function (minimizing the sum of squared errors) that linear regression solves, not the computational strategy for distributed training.
E. Singular value decomposition is a matrix decomposition technique. The question's premise correctly states that such methods do not scale well for this problem.

References:

1. Official Apache Spark Documentation: The documentation for LinearRegression in MLlib explicitly details the optimization algorithms used. It states, "The implementation is based on the MLlib LBFGS optimizer for L2-regularized linear regression... For unregularized linear regression, the implementation uses a wrapper for the NormalEquation and Cholesky solvers... The normal equation solver is limited to at most 4096 features." This confirms that for a large number of variables (beyond 4096), the iterative L-BFGS optimizer is the method employed.
Source: Apache Spark 3.5.0 MLlib Guide, Classification and regression Linear methods Linear regression Mathematical detail.
2. Academic Publication: The foundational paper on Spark's machine learning library highlights the design choice of using iterative methods for scalability. "For many ML algorithms, we can express the optimization problem as a sum of loss terms... and solve it with gradient descent. The gradient can be computed on a cluster by summing gradients computed on subsets of the data in parallel... MLlib has a general-purpose gradient descent optimizer." L-BFGS is a more advanced quasi-Newton iterative optimization method built on these principles.
Source: Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., ... & Talwalkar, A. (2016). MLlib: Machine Learning in Apache Spark. Journal of Machine Learning Research, 17(34), 1-7. (Section 3.1: Implementation).
3. Official Databricks Documentation: The Databricks documentation on Linear Regression also confirms the use of iterative optimization. "The `training` algorithm uses the L-BFGS optimizer." This directly points to an iterative optimization approach as the standard for their implementation.
Source: Databricks Documentation, Machine Learning Guide MLflow MLflow models spark.mllib Linear Regression.

Question: 3

A machine learning engineer is converting a decision tree from sklearn to Spark ML. They notice that they are receiving different results despite all of their data and manually specified hyperparameter values being identical. Which of the following describes a reason that the single-node sklearn decision tree and the Spark ML decision tree can differ?

A. Spark ML decision trees test every feature variable in the splitting algorithm

B. Spark ML decision trees automatically prune overfit trees

C. Spark ML decision trees test more split candidates in the splitting algorithm

D. Spark ML decision trees test a random sample of feature variables in the splitting algorithm

E. Spark ML decision trees test binned features values as representative split candidates

Answer:

E

Explanation:

The primary reason for the difference in results between a scikit-learn decision tree and a Spark ML decision tree, even with identical data and hyperparameters, lies in how they handle continuous features. Spark ML's decision tree algorithm is designed for distributed computation. To optimize performance and minimize data shuffling across the cluster, it discretizes continuous features into a set number of "bins" (controlled by the maxBins parameter). The algorithm then only considers the boundaries of these bins as potential split points. In contrast, single-node implementations like scikit-learn can afford to be more exhaustive, typically sorting the unique values of a feature and considering every possible split point between them. This fundamental difference in the candidate split points evaluated leads to different tree structures and, consequently, different results.

Why Incorrect Options are Wrong:

A. Both libraries' standard decision tree algorithms typically test all available feature variables at each split; this is not a point of differentiation.
B. Pruning is a user-controlled process via hyperparameters (e.g., maxDepth), which the prompt states are identical, not an automatic, differing behavior.
C. Spark ML's binning strategy for continuous features means it tests fewer representative split candidates for efficiency, not more.
D. Randomly sampling feature variables at each split is the defining characteristic of a Random Forest, not a standard Decision Tree algorithm.

References:

1. Apache Spark Official Documentation. The MLlib guide explicitly describes this behavior.
Source: Apache Spark 3.5.0, MLlib Guide Classification and regression Decision trees.
Reference: In the "Parameters" and "Implementation (developer)" subsections, it states: "For continuous features, the number of bins is controlled by the maxBins parameter. The algorithm splits on bin boundaries... The number of bins is a trade-off between computation and communication." This confirms that Spark uses a binning strategy for continuous features, which is a different approach from the exhaustive search often used in single-node libraries.
2. Peer-Reviewed Academic Publication. The foundational paper on Spark's MLlib details this design choice.
Source: Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., ... & Talwalkar, A. (2016). MLlib: Machine learning in Apache Spark. Journal of Machine Learning Research, 17(34), 1-7.
Reference: Section 3.2, "Algorithms," discusses the implementation of decision trees. It highlights that for scalability, "For continuous features, we bin the feature values into a set of discrete ordered bins... This reduces the number of split candidates to consider at each node." This directly supports the explanation that binning is a core, differentiating feature of the Spark ML implementation.

Question: 4

A data scientist is using MLflow to track their machine learning experiment. As a part of each of their MLflow runs, they are performing hyperparameter tuning. The data scientist would like to have one parent run for the tuning process with a child run for each unique combination of hyperparameter values. All parent and child runs are being manually started with mlflow.startrun. Which of the following approaches can the data scientist use to accomplish this MLflow run organization?

A. They can turn on Databricks Autologging

B. They can specify nested=True when starting the child run for each unique combination of hyperparameter values

C. They can start each child run inside the parent run's indented code block using mlflow.start runO

D. They can start each child run with the same experiment ID as the parent run

E. They can specify nested=True when starting the parent run for the tuning process

Answer:

B

Explanation:

MLflow supports hierarchical run tracking. A run that is opened while another run is already active must be declared as nested by calling mlflow.startrun(nested=True); this new run is then stored as a child of the currently-active parent run. Therefore, the data scientist should 1) open a normal parent run with mlflow.startrun() and 2) open every hyper-parameter trial inside that context using mlflow.startrun(nested=True). No other flag or setting is required.

Why Incorrect Options are Wrong:

A. Autologging only records parameters/metrics automatically; it does not create parent-child relationships between runs.
C. Starting a second run inside an active run without nested=True raises mlflow.exceptions.MlflowException ("Run is already active").
D. Sharing an experiment ID merely groups runs in the same experiment; it does not establish parent-child linkage.
E. nested=True on the parent run has no effect; nested applies only to child runs, and the parent cannot itself be nested when first opened.

References:

1. MLflow Python API - mlflow.startrun, parameter "nested", Example for parent / child runs. MLflow Docs v2.9.1, Section "mlflow.startrun" (https://mlflow.org/docs/latest/pythonapi/mlflow.html#mlflow.startrun).
2. Databricks Documentation - "Track nested MLflow runs" in "MLflow guide" (Databricks Runtime 13.3 LTS ML), paragraph 1-2.
3. Zaharia et al., "Accelerating the Machine Learning Lifecycle with MLflow", IEEE Data Engineering Bulletin 2018, p.45-46 - describes nested runs concept.

Question: 5

Which of the following approaches can be used to view the notebook that was run to create an MLflow run?

A. Open the MLmodel artifact in the MLflow run paqe

B. Click the "Models" link in the row corresponding to the run in the MLflow experiment paqe

C. Click the "Source" link in the row corresponding to the run in the MLflow experiment page

D. Click the "Start Time" link in the row corresponding to the run in the MLflow experiment page

Answer:

C

Explanation:

The Databricks MLflow experiment UI is designed for traceability and reproducibility. When an MLflow run is initiated from a Databricks notebook, the UI automatically captures a link to that notebook. This link is displayed in the "Source" column on the experiment page. Clicking this link opens a snapshot of the notebook as it existed when the run was executed, allowing users to review the exact code that produced the logged metrics, parameters, and artifacts. This is a key feature for auditing and reproducing machine learning experiments.

Why Incorrect Options are Wrong:

A. The MLmodel artifact is a metadata file within the run's artifacts that defines the model's format and dependencies, not a link to the source notebook.
B. The "Models" link or column indicates a logged model artifact and, if registered, links to its page in the Model Registry, not the source code.
D. The "Start Time" link navigates to the detailed page for that specific run, which contains metrics and artifacts, but it is not the direct link to the source notebook from the experiment view.

References:

1. Databricks Official Documentation, "Organize training runs with MLflow experiments":
Reference: In the section describing the experiment UI table, the documentation states for the "Source" column: "Name of the notebook that created the run... Click the link to view the source." This directly confirms that the "Source" link is the correct method.
Location: https://docs.databricks.com/en/mlflow/experiments.html#view-an-experiment
2. Databricks Official Documentation, "Tutorial: ML end-to-end on Databricks":
Reference: This tutorial guides users through the ML lifecycle on Databricks. In the section "View the experiment and run," it explicitly instructs the user: "To view the notebook version that created the run, click the link in the Source field." This provides a practical, documented example of the

correct procedure.
Location: https://docs.databricks.com/en/machine-learning/end-to-end-example.html#view-the-experiment-and-run

Question: 6

A data scientist is developing a machine learning pipeline using AutoML on Databricks Machine Learning. Which of the following steps will the data scientist need to perform outside of their AutoML experiment?

A. Model tuning

B. Model evaluation

C. Model deployment

D. Exploratory data analysis

Answer:

C

Explanation:

Databricks AutoML is designed to automate the iterative and time-consuming tasks of the machine learning model development process. This includes data preprocessing, model training, hyperparameter tuning, and evaluation. The output of an AutoML experiment is a set of trained models ranked on a leaderboard, along with generated notebooks. While AutoML facilitates the next step by allowing one-click registration of the best model into the MLflow Model Registry, the actual deployment of that model-for example, creating a real-time serving endpoint or setting up a batch inference job-is a distinct, subsequent step in the MLOps lifecycle that must be performed by the user outside of the AutoML experiment itself.

Why Incorrect Options are Wrong:

A. Model tuning: This is incorrect. A core feature of Databricks AutoML is to automatically perform hyperparameter tuning across a range of algorithms to find the best-performing model.
B. Model evaluation: This is incorrect. AutoML automatically evaluates each model trial using relevant metrics (e.g., F1 score for classification, RMSE for regression) and presents the results in a leaderboard for comparison.
D. Exploratory data analysis: This is incorrect. As part of its output, Databricks AutoML generates a data exploration notebook which includes a statistical summary and visualizations of the training dataset, thus performing a form of automated EDA.

References:

1. Databricks Official Documentation, "How Databricks AutoML works": This document outlines the automated steps performed by AutoML. The list includes: "Prepares the dataset for model training," "Iterates to train and tune multiple models," "Evaluates models," and "Provides a Python notebook with the source code... including a data exploration notebook." Model deployment is not

listed as an automated step.
Source: Databricks Machine Learning Guide AutoML How Databricks AutoML works.
2. Databricks Official Documentation, "Model serving with Databricks": This documentation describes model deployment as a separate process that occurs after a model has been trained and registered in the Model Registry. It details the steps to create and manage serving endpoints, which is distinct from the AutoML experiment workflow.
Source: Databricks Machine Learning Guide MLflow Model serving with Databricks.
3. The Big Book of MLOps (Databricks Ebook), Chapter 2, "A Modern MLOps Architecture": This official resource presents a diagram of the MLOps lifecycle. The "Build & Train" stage, which includes AutoML, is shown as separate and preceding the "Model Deployment" and "Model Monitoring" stages. This clearly delineates deployment as an activity outside of the automated training experiment.
Source: Databricks Ebooks, "The Big Book of MLOps", Page 15, Figure 2-1.

Question: 7

A machine learning engineer has grown tired of needing to install the MLflow Python library on each of their clusters. They ask a senior machine learning engineer how their notebooks can load the MLflow library without installing it each time. The senior machine learning engineer suggests that they use Databricks Runtime for Machine Learning. Which of the following approaches describes how the machine learning engineer can begin using Databricks Runtime for Machine Learning?

A. They can add a line enabling Databricks Runtime ML in their init script when creating their clusters.

B. They can check the Databricks Runtime ML box when creating their clusters.

C. They can select a Databricks Runtime ML version from the Databricks Runtime Version dropdown
when creating their clusters.

D. They can set the runtime-version variable in their Spark session to "ml".

Answer:

C

Explanation:

Databricks Runtime for Machine Learning (DBR ML) is a specialized, pre-packaged environment that includes optimized libraries for machine learning, such as MLflow, TensorFlow, PyTorch, and scikit-learn. To use this environment, a user must select a specific DBR ML version from the "Databricks Runtime Version" dropdown menu when configuring a new cluster. The ML runtimes are listed alongside the standard runtimes in this single dropdown (e.g., "14.3 LTS ML"). This action provisions the cluster with the pre-installed libraries, eliminating the need for manual installation via init scripts or notebook-scoped commands.

Why Incorrect Options are Wrong:

A. Init scripts are used for custom package installations and configurations, not for selecting the fundamental runtime environment of the cluster.
B. The user interface for cluster creation does not have a separate checkbox for DBR ML; it is an integrated option within the runtime version dropdown list.
D. Spark session configurations are applied after a cluster has been created and started; they cannot be used to change the underlying runtime version.

References:

1. Databricks Documentation. (2024). Databricks Runtime for Machine Learning. In "What is Databricks Runtime for Machine Learning?". The section "Create a cluster using Databricks Runtime for ML" states: "When you create a cluster, select a Databricks Runtime ML version from the Databricks Runtime Version dropdown."
Source: https://docs.databricks.com/en/runtime/mlruntime.html
2. Databricks Documentation. (2024). Compute configuration reference. In "Databricks runtime". This section describes the cluster configuration UI and clarifies that the runtime version, including ML variants, is selected from a single dropdown menu.
Source: https://docs.databricks.com/en/compute/configure.html#databricks-runtime

Question: 8

A data scientist is utilizing MLflow Autologging to automatically track their machine learning experiments. After completing a series of runs for the experiment experimentid, the data scientist wants to identify the runid of the run with the best root-mean-square error (RMSE). Which of the following lines of code can be used to identify the runid of the run with the best RMSE in experimentid? A)

```
mlflow.search_runs(
    experiment_id,
    order_by = ["metrics.rmse DESC"]
)["run_id"][0]
```

B)

```
mlflow.best_run(
    experiment_id,
    order_by = ["metrics.rmse"]
)
```

C)

```
mlflow.search_runs(
    experiment_id,
    order_by = ["metrics.rmse"]
)["run_id"][0]
```

D)

```
mlflow.best_run(
    experiment_id,
    order_by = ["metrics.rmse DESC"]
)
```

A. Option A

B. Option B

C. Option C

D. Option D

Answer:

C

Explanation:

mlflow.searchruns() returns a pandas DataFrame of all runs in an experiment.
By passing the experimentid list and ordering by the metric key in ascending
order - orderby="metrics.rmse ASC" - the first row holds the run with the
lowest (best) RMSE. Selecting .iloc0.runid therefore yields the
runid of the best-performing run.

Why Incorrect Options are Wrong:

A Uses an undefined function/attribute and does not specify ordering; cannot guarantee lowest RMSE.
B Orders by DESC, which returns the worst (largest) RMSE first.
D Filters on a fixed RMSE threshold instead of sorting, so may miss the overall best run or return many runs.

References:

1. MLflow Python API - mlflow.searchruns (Databricks documentation). Section "Parameters: orderby". https://docs.databricks.com/en/mlflow/python-api.html#mlflow-searchruns
2. MLflow Tracking User Guide, "Searching for runs" - example ordering by metric ascending. Databricks Guide v2024-03, Section"Sort by metrics".
3. N. Meng et al., "MLflow: A Platform for the Complete Machine Learning Lifecycle", SysML 2020 Tutorial, slide 32 ("searchruns orderby supports 'metrics. ASCDESC'").

Question: 9

A machine learning engineer has been notified that a new Staging version of a model registered to the MLflow Model Registry has passed all tests. As a result, the machine learning engineer wants to put this model into production by transitioning it to the Production stage in the Model Registry. From which of the following pages in Databricks Machine Learning can the machine learning engineer accomplish this task?

A. The home page of the MLflow Model Registry

B. The experiment page in the Experiments observatory

C. The model version page in the MLflow Model Registry

D. The model page in the MLflow Model Registry

Answer:

C

Explanation:

The MLflow Model Registry is used to manage the lifecycle of a model. Each registered model can have multiple versions, and each version can be assigned a stage (e.g., Staging, Production, Archived). To change the stage of a specific model version that has passed testing, the machine learning engineer must navigate to the page dedicated to that particular version. The model version page contains the controls, typically a dropdown menu, to transition the version from one stage to another, such as from Staging to Production.

Why Incorrect Options are Wrong:

A. The home page of the MLflow Model Registry: This page provides a high-level list of all registered models. It does not contain the specific controls to manage the stage of an individual model version.
B. The experiment page in the Experiments observatory: This page is for viewing MLflow experiment runs and their associated artifacts and parameters. While a model version originates from a run, its lifecycle stage is managed within the Model Registry, not the Experiments page.
D. The model page in the MLflow Model Registry: This page lists all versions of a single registered model. While you can see the stages of all versions here and initiate a transition, the most direct and specific location to manage a single version's properties is its own dedicated page.

References:

1. Databricks Official Documentation, "Manage model lifecycle in Model Registry":
Section: Transition a model stage
Content: The documentation explicitly states, "You can transition a model version's stage from the model version page." It provides a step-by-step guide: "1. In the sidebar, click Models. 2. Click the name of a model to open the registered model page. 3. Click a model version to open the model version page. 4. Use the Stage drop-down menu at the top-right to transition the model version's stage." This directly supports that the action is performed on the model version page.
Source:
https://docs.databricks.com/en/mlflow/model-registry.html#transition-a-model-versions-stage
2. Databricks Official Documentation, "Model Registry on Databricks":
Section: Model version page
Content: This section describes the components of the model version page, stating: "The model version page displays information about a specific registered model version... You can also perform tasks from this page, such as deleting a model version or transitioning a model version to another stage." This confirms the model version page is the correct location for this task.
Source: https://docs.databricks.com/en/mlflow/model-registry.html#model-version-page

Question: 10

A machine learning engineer has identified the best run from an MLflow Experiment. They have stored the run ID in the runid variable and identified the logged model name as "model". They now want to register that model in the MLflow Model Registry with the name "bestmodel". Which lines of code can they use to register the model associated with runid to the MLflow Model Registry?

A. mlflow.registermodel(runid, "bestmodel")

B. mlflow.registermodel(f"runs:/runid/model", "bestmodel")

C. millow.registermodel(f"runs:/runid)/model")

D. mlflow.registermodel(f"runs:/runid/bestmodel", "model")

Answer:

B

Explanation:

The mlflow.registermodel() function is used to create a new model version in the MLflow Model Registry from a previously logged model artifact. The function's signature requires two key arguments: modeluri and name.

The modeluri must point to the location of the logged model artifact. For a model within a specific run, the URI scheme is runs://. In this scenario, the run ID is in the runid variable and the model artifact path is "model". Therefore, the correct modeluri is f"runs:/runid/model".

The name argument specifies the name under which the model will be registered in the registry, which is "bestmodel" as required by the question.

Why Incorrect Options are Wrong:

A. This option provides only the runid as the first argument, which is not a valid model URI. The function requires a URI string like runs:/....

C. This option is missing the required name argument. The function must be told what name to register the model under in the Model Registry.

D. This option incorrectly swaps the artifact path and the desired registered model name. It points to a non-existent artifact ("bestmodel") and tries to register it under the wrong name ("model").

References:

1. Databricks Documentation, "Log, load, register, and deploy MLflow Models": This document explicitly shows the syntax for registering a model from a run. In the "Register a model" section, it provides the example: mlflow.registermodel(modeluri="runs:/runid/artifactpath", name=modelname). This directly validates the structure used in option B.

Source: Databricks Official Documentation Machine Learning MLflow Log, load, register, and

deploy MLflow Models.
2. MLflow Official Documentation, mlflow.registermodel API Reference: The official MLflow API documentation defines the function as mlflow.registermodel(modeluri, name, ...). It specifies that modeluri is the location of the saved model and name is the name for the registered model. It also describes the runs:/ URI scheme for referencing artifacts from a run.
Source: mlflow.org Docs Latest API Reference mlflow.
3. Databricks Documentation, "Manage model lifecycle in Model Registry": This guide provides a comprehensive example notebook. In "Step 4: Register the model to the MLflow Model Registry", the code cell demonstrates the exact pattern: result = mlflow.registermodel(modeluri, modelname), where modeluri is constructed as f"runs:/run.info.runid/model".
Source: Databricks Official Documentation Machine Learning MLflow Model Registry Manage model lifecycle in Model Registry.

Question: 11

A new data scientist has started working on an existing machine learning project. The project is a scheduled Job that retrains every day. The project currently exists in a Repo in Databricks. The data scientist has been tasked with improving the feature engineering of the pipeline's preprocessing stage. The data scientist wants to make necessary updates to the code that can be easily adopted into the project without changing what is being run each day. Which approach should the data scientist take to complete this task?

A. They can create a new branch in Databricks, commit their changes, and push those changes to the
Git provider.

B. They can clone the notebooks in the repository into a Databricks Workspace folder and make the
necessary changes.

C. They can create a new Git repository, import it into Databricks, and copy and paste the existing code from the original repository before making changes.

D. They can clone the notebooks in the repository into a new Databricks Repo and make the necessary changes.

Answer:

A

Explanation:

The project is managed within a Databricks Repo, which is integrated with a Git provider. The standard and best practice for developing new features or making changes without disrupting the main production code is to use a new Git branch. This isolates the development work from the main branch, which the scheduled Job is presumably running from. The data scientist can create a branch, commit their changes, and push them to the remote Git provider. Once the work is complete and tested, it can be merged into the main branch through a pull request, ensuring a controlled and reviewable process for adoption into the daily job.

Why Incorrect Options are Wrong:

B. Cloning notebooks to a Workspace folder removes them from Git version control, making integration of changes difficult and breaking the established project workflow.
C. Creating an entirely new Git repository disconnects the work from the original project's history and makes merging changes back a manual, complex process.
D. Cloning into a new Databricks Repo creates a separate project fork, which is an unnecessarily complex approach compared to using a branch for feature development.

References:

1. Databricks Official Documentation, "Git operations with Databricks Repos": This document outlines the standard Git workflow within Databricks. The section "Create a new branch" explicitly describes the procedure for this task: "You can create a new branch based on an existing branch from within a repo... This is the best practice for developing your new work." This directly supports the methodology in option A as the recommended approach for isolated development.
Source: Databricks Documentation, docs/en/repos/git-operations-with-repos.html, Section: "Create a new branch".
2. Databricks Official Documentation, "CI/CD techniques with Git and Databricks Repos": This guide on best practices for development workflows emphasizes using separate Git branches for development work ("feature branches") to isolate changes from the main production branch. It states, "A common workflow is to create a new feature branch for your work... You can make your changes, and then commit and push them to the Git provider." This aligns perfectly with the scenario of improving a feature without affecting the daily production job.
Source: Databricks Documentation, docs/en/repos/ci-cd-repos.html, Section: "Development workflow".

Question: 12

A machine learning engineering team has a Job with three successive tasks. Each task runs a single notebook. The team has been alerted that the Job has failed in its latest run. Which of the following approaches can the team use to identify which task is the cause of the failure?

A. Run each notebook interactively

B. Review the matrix view in the Job's runs

C. Migrate the Job to a Delta Live Tables pipeline

D. Change each Task's setting to use a dedicated cluster

Answer:

B

Explanation:

The Databricks Jobs UI provides a detailed history for each job run. The matrix view (also known as the Gantt view) within a specific job run visually displays the status of each task in the job. This view clearly indicates which tasks succeeded, which are running, and, most importantly, which one failed, along with its duration and start/end times. This is the primary and most efficient method for pinpointing the exact point of failure in a multi-task job without needing to re-run code or change the job's configuration.

Why Incorrect Options are Wrong:

A. Running notebooks interactively is a manual debugging step performed after identifying the failed task, not the method to identify it.
C. Migrating to Delta Live Tables is a major architectural change and is not a tool for diagnosing a standard Databricks Job failure.
D. Using dedicated clusters is a configuration change to prevent future failures, not a method to diagnose a past failed run.

References:

1. Databricks Official Documentation, "View job runs": This document describes how to monitor job runs. It states, "To view the run history for a job, click the job name in the Jobs list... You can view the matrix or Gantt chart of job runs and a list of job runs." The matrix view visually distinguishes between successful and failed tasks. (See section: "View the runs for a job").
2. Databricks Official Documentation, "Troubleshoot and fix job failures": This guide explicitly recommends using the job run history to diagnose failures. It states, "If a job fails, you can investigate the cause of the failure by viewing the job's run history... The job run details page shows which tasks ran successfully and which failed." (See section: "View job run details to determine the cause of a job failure").

Question: 13

A data scientist is using Spark SQL to import their data into a machine learning pipeline. Once the data is imported, the data scientist performs machine learning tasks using Spark ML. Which of the following compute tools is best suited for this use case?

A. Single Node cluster

B. Standard cluster

C. SQL Warehouse

D. None of these compute tools support this task

Answer:

B

Explanation:

The described workload involves both data manipulation using Spark SQL and model training using Spark ML. This combination is a typical data science task that requires a general-purpose, interactive environment capable of running both SQL and arbitrary code (like Python for Spark ML). A Standard cluster (also known as an all-purpose cluster) is specifically designed for these interactive data science and data engineering workloads. It allows data scientists to attach notebooks and execute a mix of commands, including Spark SQL for data preparation and Spark ML for distributed model training, making it the best-suited compute tool for this scenario.

Why Incorrect Options are Wrong:

A. Single Node cluster: This cluster has no worker nodes and is not designed for distributed computing, which is the primary advantage of using Spark ML for scalable machine learning tasks.
C. SQL Warehouse: This compute resource is optimized specifically for running SQL queries for business intelligence (BI) and analytics. It cannot be used to execute general-purpose code or machine learning libraries like Spark ML.
D. None of these compute tools support this task: This is incorrect because a Standard cluster is the designated and appropriate compute resource for this exact combination of tasks on the Databricks platform.

References:

1. Databricks Official Documentation, "What is Databricks compute?": This document distinguishes between different compute types. It states, "For data science and data engineering workloads, you can use either all-purpose compute or job compute." It further clarifies that SQL warehouses are for "running SQL queries with Databricks SQL." This directly supports using a

Standard (all-purpose) cluster for a data science workload involving Spark ML and explicitly excludes SQL Warehouses.
Source: Databricks Documentation Compute Get started What is Databricks compute?
2. Databricks Official Documentation, "Cluster modes": This page details the different cluster modes. It describes the "Standard" mode as the recommended option for single users, capable of running workloads in various languages (Python, R, Scala, SQL). It contrasts this with the "Single Node" mode, which has no workers and is not suitable for large-scale Spark jobs.
Source: Databricks Documentation Compute Configuration Cluster modes.
3. Databricks Official Documentation, "What is a SQL warehouse?": This source defines the purpose of a SQL Warehouse. It states, "A SQL warehouse is a compute resource that lets you run SQL commands on data objects within Databricks SQL." This confirms its specialization for SQL and BI tools, not for programmatic machine learning tasks found in notebooks.
Source: Databricks Documentation SQL Get started What is a SQL warehouse?

Question: 14

A machine learning engineer is trying to perform batch model inference. They want to get predictions using the linear regression model saved at the path modeluri for the DataFrame batchdf. batchdf has the following schema: customerid STRING The machine learning engineer runs the following code block to perform inference on batchdf using the linear regression model at modeluri:

```
predictions = fs.score_batch(
    model_uri,
    batch_df
)
```

In which situation will the machine learning engineer's code block perform the desired inference?

A. When the Feature Store feature set was logged with the model at modeluri

B. When all of the features used by the model at modeluri are in a Spark DataFrame in the PySpark

C. When the model at modeluri only uses customerid as a feature

D. This code block will not perform the desired inference in any situation.

E. When all of the features used by the model at modeluri are in a single Feature Store table

Answer:

A

Explanation:

The code attempts to generate predictions on a DataFrame (batchdf) that contains only a primary key (customerid). For a typical machine learning model that requires multiple feature columns, this would fail. However, the Databricks Feature Store provides a specific mechanism to handle this scenario. By using featurestore.logmodel(), a model is packaged with metadata about the features it was trained on. When this "feature-aware" model is used for inference via mlflow.pyfunc.sparkudf, it automatically uses the provided keys (customerid) to look up the corresponding feature values from the Feature Store, joins them, and then computes the prediction. This is the only scenario among the options where the code will execute as intended.

Why Incorrect Options are Wrong:

B. The code as written does not reference or perform any joins with other Spark DataFrames, so their mere existence in the environment is irrelevant.
C. A linear regression model requires numerical features for computation. A string identifier like customerid is unsuitable as the sole feature and would likely cause a type error.
D. This is incorrect because the integration between MLflow and the Databricks Feature Store (as described in option A) provides a valid and common pattern for this code to work.
E. While the features must exist in the Feature Store, the critical condition is that the model was logged with the feature lookup metadata, making it "feature-aware".

References:

1. Databricks Official Documentation, "Train models and perform batch inference with Feature Store":
Section: "Perform batch inference"
Content: This section explicitly states: "To score a model on new data, use the featurestore.scorebatch method... This method looks up the features for the data in df from the feature tables specified in the featurelookups used when the model was logged...". The mlflow.pyfunc.sparkudf leverages this underlying capability for models logged with feature metadata. This directly supports that logging the feature set with the model is the required condition.

2. Databricks Official Documentation, "Feature Store Python API reference":
Section: databricks.featurestore.client.FeatureStoreClient.logmodel
Content: The documentation for this function explains that it "Packages the model with feature metadata." This metadata is precisely what enables the automatic feature lookup during inference, which is the core principle making the code in the question work.
3. Databricks Official Documentation, "What is a feature store?":
Section: "Model training and inference with Feature Store"
Content: The documentation illustrates the MLOps lifecycle, showing that for inference, the model can be provided with only primary keys. The model then "retrieves the precomputed features from the feature store" before making a prediction. This confirms the pattern described in the correct answer.

Question: 15

Which of the following evaluation metrics is not suitable to evaluate runs in AutoML experiments for regression problems?

A. F1

B. R-squared

C. MAE

D. MSE

Answer:

A

Explanation:

The F1 score is an evaluation metric used for classification problems, not regression. It is calculated as the harmonic mean of precision and recall, which are metrics based on the counts of true positives, false positives, and false negatives from a confusion matrix. These concepts are fundamentally tied to predicting discrete class labels. Regression problems, in contrast, predict continuous numerical values. Therefore, metrics like R-squared, Mean Absolute Error (MAE), and Mean Squared Error (MSE) are appropriate as they measure the magnitude of the error between the predicted and actual continuous values.

Why Incorrect Options are Wrong:

B. R-squared: This is a standard regression metric measuring the proportion of variance in the target variable that is predictable from the features.
C. MAE: Mean Absolute Error is a common regression metric that measures the average magnitude of the errors in a set of predictions.
D. MSE: Mean Squared Error is a widely used regression metric that measures the average of the squared differences between predicted and actual values.

References:

1. Databricks Official Documentation, "Regression and forecasting: model metrics": This document explicitly lists the metrics calculated for each run in a Databricks AutoML regression experiment. The primary metric is root mean squared error (RMSE), and other generated metrics include R-squared, MAE, and MSE. The F1 score is not mentioned for regression.
Source: Databricks Machine Learning Guide AutoML Reference Regression and forecasting: model metrics.
2. Databricks Official Documentation, "Classification: model metrics": This document lists the metrics for AutoML classification experiments, which include F1 score, accuracy, log loss, precision, and recall. This confirms that F1 is a classification metric within the Databricks

ecosystem.
Source: Databricks Machine Learning Guide AutoML Reference Classification: model metrics.
3. Stanford University, CS229 Machine Learning Course Notes: In the course materials covering evaluation metrics, a clear distinction is made. Metrics for classification include accuracy, precision, recall, and F1 score. Metrics for regression include Mean Squared Error (MSE) and Mean Absolute Error (MAE).
Source: Ng, A. (2023). CS229 Machine Learning Course Notes, Stanford University, "Part V: Learning Theory" and "Part VI: Evaluation Metrics".