# Databricks.Databricks-Machine-Learning-Professional.v2026-04-30.q96

| | |
|---|---|
| **Exam Code:** | Databricks-Machine-Learning-Professional |
| **Exam Name:** | Databricks Certified Machine Learning Professional |
| **Certification Provider:** | Databricks |
| **Free Question Number:** | 96 |
| **Version:** | v2026-04-30 |
| **# of views:** | 103 |
| **# of Questions views:** | 977 |
| https://www.freecram.net/torrent/Databricks.Databricks-Machine-Learning-Professional.v2026-04-30.q96.html | |

**NEW QUESTION: 1**
Which of the following is a simple, low-cost method of monitoring numeric feature drift?
**A.** Kolmogorov-Smirnov (KS) test
**B.** Chi-squared test
**C.** None of these can be used to monitor feature drift
**D.** Jensen-Shannon test
**E.** Summary statistics trends
**Answer: (SHOW ANSWER)**

**NEW QUESTION: 2**
What is the main purpose of the Databricks Feature Store?
**A.** Storing raw data
**B.** Centralizing reusable ML features
**C.** Replacing Spark ML pipelines
**D.** Storing trained models
**Answer: (SHOW ANSWER)**
Feature Store allows teams to:
share features
avoid training/serving skew
maintain feature lineage.

**NEW QUESTION: 3**
A Machine Learning Engineer has deployed a fraud detection model that processes 10,000 transactions per hour. The model was trained on data from Q1 2024, but it's now Q4 2024. The ML team notices three concerning trends: (1) the model's precision has dropped from 92% to

78% over the past month, (2) the average transaction amount in recent data has increased from $150 to $220 and (3) the relationship between transaction frequency and fraud likelihood has weakened significantly due to new payment methods being introduced. The engineer needs to implement a monitoring solution that can detect the root cause of the performance degradation, identify why the precision dropped, and be able to do this at the scale needed. Which monitoring pipeline component will do this?

**A.** Model Health Monitoring Pipeline because it ensures the infrastructure can handle the 10,000 transactions per hour load.

**B.** Drift Detection Pipeline because it can identify both the data drift (transaction amount changes) and concept drift (weakened relationship between features and target).

**C.** Logging and Monitoring Pipeline because it captures all prediction requests and can identify patterns in the fraud predictions.

**D.** Model Performance Monitoring Pipeline because it directly tracks precision metrics and can alert when thresholds are breached.

**Answer: (SHOW ANSWER)**

A drift detection pipeline is designed to diagnose the root causes of model performance degradation at scale. It can detect data drift, such as changes in the distribution of transaction amounts, and concept drift, where the relationship between input features and the fraud label changes due to new payment methods. This directly explains why precision dropped and provides actionable insight beyond simply observing metric degradation.

**NEW QUESTION: 4**

Which of the following machine learning model deployment paradigms is the most common for machine learning projects?

**A.** Streaming

**B.** None of these deployments

**C.** Batch

**D.** On-device

**E.** Real-time

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 5**

Which of the following statements about built-in library-specific MLflow Model flavors is true?

**A.** Built-in library-specific flavors are required for model signature use

**B.** Built-in library-specific flavors allow models to be exported as library objects

**C.** Built-in library-specific flavors allow models to be used with any library/span>

**D.** Built-in library-specific flavors can only be used for logging models

**Answer: (SHOW ANSWER)**

Built-in library-specific MLflow model flavors (e.g., mlflow.sklearn, mlflow.xgboost) allow models to be exported and later loaded as native library objects, enabling seamless reuse with the original libraries for inference or further training.

**NEW QUESTION: 6**
Which of the following can be used to compare the relative prevalence of specific values in a single categorical variable between two datasets or time periods?
**A.** Contingency tables
**B.** Number of unique values
**C.** Mode
**D.** Jensen-Shannon distance
**Answer: (SHOW ANSWER)**
Contingency tables (also known as cross-tabulations) display the frequency distribution of categorical variables, allowing comparison of how specific category values occur across two datasets or time periods. This makes them effective for identifying categorical feature drift or shifts in value prevalence.

**NEW QUESTION: 7**
In order to connect an MLflow Model Registry Webhook to a Databricks Job, the Job ID must be provided to the code block used to create the webhook. Which approach can be used to obtain a Databricks Job ID?
**A.** The Job ID field returned in a query using the Jobs API
**B.** The Job ID field in the Jobs page in the UI
**C.** The Job ID field in the Job details section of the specific Job page in the UI
**D.** All of these approaches
**Answer: (SHOW ANSWER)**
A Databricks Job ID can be obtained in multiple ways - it is displayed directly in the Jobs page, in the Job details section of a specific Job, and can also be retrieved programmatically through the Databricks Jobs API. Any of these methods can be used to supply the Job ID when configuring an MLflow Model Registry Webhook.

**NEW QUESTION: 8**
A Machine Learning Engineer needs to build a time series model. In Databricks, they have created isolated environments in different workspaces for development, staging, and production. To manage this model, they are planning on using a "deploy code" strategy. They are concerned that the model trained in development will not remain consistent across environments, due to differences in the dependencies installed. What can they do to ensure that the model is trained with the same packages?
**A.** Use the same Databricks Runtime (DBR) version across environments and define dependencies in a lock file that is installed as part of the job.
**B.** Use custom init scripts that install the necessary packages as part of the job.
**C.** Use MLflow to track the model parameters and metrics in development, then export the same training conditions in staging and production.

**D.** Use Databricks Repos to store the model code and install the required libraries in the notebook.

**Answer: (SHOW ANSWER)**

Using the same Databricks Runtime across environments ensures a consistent base environment, and installing dependencies from a lock file guarantees identical package versions during training. This combination follows best practices for deploy-code workflows by making the training environment reproducible and preventing dependency drift between development, staging, and production.

**NEW QUESTION: 9**

Which of the following MLflow Model Registry use cases requires the use of an HTTP Webhook?

**A.** None of these use cases require the use of an HTTP Webhook

**B.** Starting a testing job when a new model is registered

**C.** Updating data in a source table for a Databricks SQL dashboard when a model version transitions to the Production stage

**D.** Sending a message to a Slack channel when a model version transitions stages

**E.** Sending an email alert when an automated testing Job fails

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 10**

Which MLflow component is used to log parameters, metrics, and artifacts during model training?

**A.** Model Registry

**B.** MLflow Tracking

**C.** MLflow Projects

**D.** MLflow Models

**Answer: (SHOW ANSWER)**

MLflow Tracking records experiment runs including:
parameters
metrics
artifacts (models, plots, etc.)

**NEW QUESTION: 11**

A Machine Learning Engineer wants to monitor the quality and stability of their machine learning model's predictions over time. They have a Delta table, retail_inference_log, which records each model prediction along with input features, a timestamp, and (when available) the true label. They need to detect data drift and monitor model performance trends using Databricks Lakehouse Monitoring, ensuring that alerts are triggered if the distribution of predictions or input features changes significantly. Which approach will set up monitoring for this use case?

**A.** Create a monitor with the Inference profile on the retail_inference_log table, specifying the timestamp column and the columns for model inputs, predictions, and labels. Configure the monitor to compute drift and performance metrics over time windows.

**B.** Create a monitor with the Inference profile on the retail_inference_log table, and specify a recent batch of production data as the baseline table for drift detection. Use this recent production data to compare against new data for drift and performance monitoring.
**C.** Create a monitor with the Snapshot profile on the retail_inference_log table, so that metrics are calculated over the entire table each time the monitor runs and therefore is able to compare new values with previous ones to compute data drift.
**D.** Create a monitor with the Time Series profile on the retail_inference_log table, specifying the timestamp column and including model input, prediction columns and the true label column. This will track drift in features and predictions over time, and model performance could also be tracked using a custom metric.
**Answer: (SHOW ANSWER)**
The Inference profile is specifically designed for monitoring production inference logs. By configuring it on the inference table with the timestamp, input feature columns, prediction column, and label column, Databricks Lakehouse Monitoring can automatically compute prediction drift, input feature drift, and model performance metrics over rolling time windows, and trigger alerts when significant distribution changes or performance degradation are detected.

**NEW QUESTION: 12**
Feature drift occurs when there is a change in which element?
**A.** The distribution of the predicted target given by the model
**B.** The distribution of an input variable
**C.** The distribution of a target variable
**D.** The relationship between input variables and target variables
**Answer: (SHOW ANSWER)**
Feature drift refers to a change in the distribution of one or more input features over time, even if the target variable and model relationship remain stable. This can degrade model performance because the model was trained on a different feature distribution than it encounters during inference.

**NEW QUESTION: 13**
Which of the following is a simple statistic to monitor for categorical feature drift?
**A.** Mode
**B.** Mode, number of unique values, and percentage of missing values
**C.** Number of unique values
**D.** Percentage of missing values
**E.** None of these
**Answer: (SHOW ANSWER)**

**NEW QUESTION: 14**
A Data Scientist has created a sales forecasting model, named sales-forecasting. The model is deployed to a model serving endpoint with a schema. They need to invoke this model using

Python and would prefer to use a SDK function to make the request. Which method suits these requirements?

**A.** Import the MLflow Deployments class and use the predict method and provide the endpoint and input parameters.

**B.** Use the built in ai_query function and provide the endpoints and request parameters.

**C.** Import the MLflow Deployments class and use the ai_query method and provide the endpoints and request parameters.

**D.** Make an API request to the MODEL_VERSION_URI and provide the dataframe_split as the request parameter.

**Answer: (SHOW ANSWER)**

The MLflow Deployments SDK provides a predict method specifically designed to invoke Databricks model serving endpoints from Python. This method abstracts the HTTP request details, allows specifying the endpoint name directly, and cleanly passes input parameters that conform to the model's serving schema, making it the preferred SDK-based approach.

**NEW QUESTION: 15**

A team is building a machine learning pipeline, which includes feature engineering, model training, model evaluation, and model deployment. Over the last year, the project has grown a lot, with multiple contributors working on it. Currently, they are manually testing their pipelines and functions. However, they are looking to build out a MLOps process and want to organize their unit tests. Which approach to organizing functions and unit tests should they take?

**A.** Organize functions and unit tests together in multiple notebooks, grouping them by feature or component.

**B.** Write unit tests only after deployment to avoid slowing down development.

**C.** Place all functions and their corresponding unit tests in a single notebook or script.

**D.** Separate functions into dedicated modules and place all unit tests in a separate test suite or notebook.

**Answer: D (LEAVE A REPLY)**

Separating production code into dedicated modules and organizing unit tests in a separate test suite follows standard software engineering and MLOps best practices. This structure improves maintainability, enables automated testing in CI/CD pipelines, supports collaboration among multiple contributors, and scales effectively as the project grows.

**NEW QUESTION: 16**

A machine learning engineer is attempting to create a webhook that will trigger a Databricks Job job_id when a model version for model model transitions into any MLflow Model Registry stage. They have the following incomplete code block:



Which lines of code can be used to fill in the blank so that the code block accomplishes the task?

**A.** "MODEL_VERSION_TRANSITIONED_TO_STAGING", "MODEL_VERSION_TRANSITIONED_TO_PRODUCTION"

**B.** "MODEL_VERSION_TRANSITIONED_TO_PRODUCTION"

**C.** "MODEL_VERSION_TRANSITIONED_STAGE"

**D.** "MODEL_VERSION_CREATED"

**E.** "MODEL_VERSION_TRANSITIONED_TO_STAGING"

**Answer: (SHOW ANSWER)**



**NEW QUESTION: 17**

Concept drift is when there is a change in which element?

**A.** The relationship between input variables and target variables

**B.** The distribution of an input variable

**C.** The distribution of the predicted target given by the model

**D.** The distribution of a target variable

**Answer: (SHOW ANSWER)**

Concept drift occurs when the underlying relationship between input features and the target variable changes over time. This means that even if the feature and label distributions remain stable, the way features influence the target (the predictive relationship) has shifted, leading to reduced model accuracy unless retraining occurs.

**NEW QUESTION: 18**
A Machine Learning Engineer has a production AI model that predicts fraudulent transactions in real time. This model is integrated into critical business workflows and any downtime could result in significant financial loss and regulatory penalties. The engineer needs to deploy a new, improved version of the model to production to replace the existing model. The bank's fairness policy requires that all transactions be exposed to the same model at any given time. Additionally, the deployment must meet the following requirements:
- Zero downtime: The fraud detection service must remain continuously
available to users and downstream systems.
- Immediate rollback: If the new model causes issues, you must be able
to revert to the previous version instantly.
Which deployment strategy meets these requirements?
**A.** Canary
**B.** Rolling
**C.** Shadow
**D.** Blue-Green
**Answer: (SHOW ANSWER)**
Blue-green deployment maintains two fully operational production environments and switches all traffic from the old model to the new model at once. This guarantees zero downtime and ensures that all transactions are handled by the same model version at any given time. If issues arise, traffic can be immediately switched back to the previous environment, enabling instant rollback while meeting strict fairness and availability requirements.

**NEW QUESTION: 19**
A Machine Learning Engineer is setting up a cluster for a deep learning training run, but has a number of settings options to choose from. Their cluster will be reused by other engineers on their team and their datasets vary in size from hundreds of MBs to hundreds of GBs. They need to choose a configuration that allows for performant, stable deep learning training without excessive costs. Which configuration will do this?
**A.** Using ML Runtime, use a moderate sized GPU VM for the driver and a variable number of memory optimized CPU VMs for the workers with autoscale enabled
**B.** Using ML Runtime, use a large single node, GPU-enabled VM with auto termination set to 20 minutes to reduce costs
**C.** Using ML Runtime, use a moderately sized CPU VM for the driver and a variable number of GPU enabled VMs for the workers with autoscale enabled

**D.** Using Databricks Runtime, use a moderately sized CPU VM for the driver and a variable number of GPU enabled VMs for the workers with autoscale enabled

**Answer: (SHOW ANSWER)**

Using the ML Runtime ensures deep learning frameworks and GPU drivers are preconfigured and optimized. A moderately sized CPU driver is sufficient for coordination, while GPU-enabled worker nodes handle the computationally intensive training workload. Enabling autoscaling allows the cluster to efficiently adapt to datasets ranging from hundreds of megabytes to hundreds of gigabytes, providing strong performance without overprovisioning and controlling costs in a shared team environment.

**NEW QUESTION: 20**

A machine learning engineer has registered a Spark ML model in the MLflow Model Registry using the Spark ML model flavor with UI model_uri. Which operation can be used to load the model as a Spark ML object for batch deployment?



**A.** mlflow.spark.read_model(model_uri)

**B.** mlflow.pyfunc.load_model(model_uri)

**C.** mlflow.spark.load_model(model_uri)

**D.** mlflow.sklearn.load_model(model_uri)

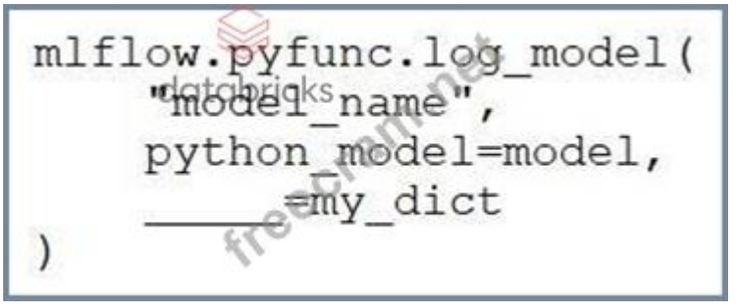
**Answer: C (LEAVE A REPLY)**

To load a model that was logged using the Spark ML flavor, the correct operation is mlflow.spark.load_model(model_uri). This restores the model as a Spark ML object, preserving its original structure and functionality for batch inference or further processing in Spark environments.

**NEW QUESTION: 21**

A machine learning engineer needs a python_model to access a collection of files using its load_context operation. The collection of files being accessed by python_model.load_context needs to be saved when the model is being logged. A dictionary of the names and paths of these files is properly stored in my_dict.

The machine learning engineer has written the following incomplete block of code:




Which lines of code can be used to fill in the blank to successfully complete the code block to accomplish the task?

**A.** artifacts

**B.** None of these lines of code can be used to successfully complete the code block

**C.** context

**D.** model_context

**Answer: (SHOW ANSWER)**

When logging a custom Python model with mlflow.pyfunc.log_model(), the artifacts parameter is used to specify a dictionary (my_dict) of file names and their paths that the model may need to access during loading or inference. These files are stored as model artifacts and become available to the model through the load_context method.

**NEW QUESTION: 22**

Which MLflow function registers a model to the registry?

**A.** mlflow.log_param()

**B.** mlflow.register_model()

**C.** mlflow.deploy_model()

**D.** mlflow.create_model()

**Answer: (SHOW ANSWER)**

mlflow.register_model() promotes a model into the Model Registry.

**NEW QUESTION: 23**

Which of the following is a benefit of logging a model signature with an MLflow model?

**A.** The schema of input data will be converted to match the signature

**B.** The schema of input data can be validated when serving models

**C.** The model will have a unique identifier in the MLflow experiment

**D.** The model can be deployed using real-time serving tools

**E.** The model will be secured by the user that developed it

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 24**

A data scientist has developed a scikit-learn random forest model model, but they have not yet logged model with MLflow. They have created a model signature model_signature and a few example input records input_records. Which block of code can be used to log all of this information?

**A.**

```
mlflow.sklearn.log_model(
    model,
    "model",
    signature=model_signature,
    input_example=input_records
)
```



**B.** mlflow.sklearn.log_model(model, "model")

mlflow.log_signature(model_signature)
mlflow.log_input(input_records)

**C.** There is no way to log a model signature.

```
mlflow.sklearn.log_model(
    model,
    "model",
    signature=input_records,
    input_example=model_signature
)
```

**D.**

**Answer: (SHOW ANSWER)**

The code block correctly uses the mlflow.sklearn.log_model() function with both signature and input_example parameters.
signature=model_signature defines the expected model input and output schema, ensuring reproducibility and validation during inference.
input_example=input_records provides a sample input that helps users understand the model's expected input format.

**NEW QUESTION: 25**

A Machine Learning Engineer needs to build a credit risk model using Databricks. Due to strict data governance, production data cannot be accessed from development or staging environments. To manage MLOps, the engineer uses a "deploy code" strategy with separate development, staging, and production environments mapped to different catalogs in Unity Catalog. The CI/CD pipeline automates environment transitions. What is the primary architectural component promoted from staging to production to generate the final, production-ready model in this scenario?

**A.** A containerized image of the complete inference pipeline, including feature transformations.

**B.** The CI/CD-approved model training source code, which will be executed in the production environment on the full production dataset.

**C.** The model version entry in the Unity Catalog that has been transitioned from the "Staging" to the
"Production" alias.

**D.** The model artifact trained and validated in the staging environment on a subset of the data.

**Answer: (SHOW ANSWER)**

In a deploy-code strategy, models are not promoted as trained artifacts across environments. Instead, the approved training code is promoted through CI/CD and re-executed in the production environment against production data to produce the final model. This approach satisfies strict data governance requirements by ensuring production data is only accessed in production while maintaining reproducibility and traceability.

**NEW QUESTION: 26**

A Machine Learning Engineer has created a custom PyFunc model wrapper for a fraud detection system that needs to be registered in Unity Catalog under the schema risk_models in the production catalog. The model requires specific dependencies and must be accessible for governance and version control. The engineer is working on a dedicated cluster with Unity Catalog enabled, but the model registration is failing. Why is the model failing to be registered in this case?

**A.** The model signature must be provided, as Unity Catalog requires all models to have signatures.

**B.** The cluster must be configured with shared access mode instead of dedicated mode.

**C.** The PyFunc model must inherit from mlflow.sklearn.Model instead of mlflow.pyfunc.PythonModel.

**D.** The model artifacts must be stored in DBFS rather than Unity Catalog volumes.

**Answer: B (LEAVE A REPLY)**

Unity Catalog model registration requires the cluster to run in shared access mode. Dedicated (single-user) clusters do not support registering models into Unity Catalog because governance, lineage, and access control enforcement rely on shared-mode execution. As a result, even if Unity Catalog is enabled, model registration will fail when attempted from a dedicated cluster.

**NEW QUESTION: 27**

A machine learning engineer wants to move their model version model_version for the MLflow Model Registry model model from the Staging stage to the Production stage using MLflow Client client. Which of the following code blocks can they use to accomplish the task?

**A.**

```
client.transition_model__stage(
    name=model,
    version=model_version,
    from="Staging",
    to="Production"
)
```



**B.**

```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    from="Staging",
    to="Production"
)
```

**C.**

```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    stage="Production"
)
```

**D.**

```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    stage="Production"
)
```

```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    stage="Staging"
)
```

**E.**

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 28**

Which MLflow function logs files like plots or datasets?

**A.** mlflow.log_param()

**B.** mlflow.log_artifact()

**C.** mlflow.log_metric()

**D.** mlflow.save_model()

**Answer: (SHOW ANSWER)**

Artifacts include:

charts

model files

datasets.

**NEW QUESTION: 29**

What tool helps avoid feature mismatch between training and inference?

**A.** MLflow Projects

**B.** Feature Store

**C.** Model Registry

**D.** Spark SQL

**Answer: (SHOW ANSWER)**
Feature Store ensures:
same feature computation logic
consistency between training and serving.

**NEW QUESTION: 30**
A machine learning engineer is migrating a machine learning pipeline to use Databricks Machine Learning. They have programmatically identified the best run from an MLflow Experiment and stored its URI in the model_uri variable and its Run ID in the run_id variable. They have also determined that the model was logged with the name "model". Now, the machine learning engineer wants to register that model in the MLflow Model Registry with the name "best_model".
Which line of code can they use to register the model to the MLflow Model Registry?

**A.** mlflow.register_model(model_uri, "best_model")
**B.** mlflow.register_model(run_id, "best_model")
**C.** mlflow.register_model(f"runs:/{run_id}/best_model", "model")
**D.** mlflow.register_model(f"runs:/{run_id}/model")
**E.** mlflow.register_model(model_uri, "model")
**Answer: A (LEAVE A REPLY)**

**NEW QUESTION: 31**
Which component manages model versions?
**A.** MLflow Tracking
**B.** Model Registry
**C.** Feature Store
**D.** Spark ML
**Answer: (SHOW ANSWER)**
Model Registry handles:
versioning
stage transitions
governance.

**NEW QUESTION: 32**

A machine learning engineer has created a webhook with the following code block:

```
job_json = {
    "model_name": model,
    "events": ["MODEL_VERSION_TRANSITIONED_TO_STAGING"],
    "description": "Job webhook trigger",
    "status": "Active",
    "job_spec": {
        "job_id": job_id,
        "workspace_url": url,
        "access_token": token
    }
}

response = http_request(
    host_creds=host_creds,
    endpoint=endpoint,
    method="POST",
    json=job_json
)
```

Which of the following code blocks will trigger this webhook to run the associate job?

**A.**

```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    from="None",
    to="Staging"
)
```

**B.**

```
client.transition_model_stage(
    name=new_model,
    version=model_version,
    stage="Staging"
)
```

**C.**

```
client.transition_model_version_stage(
    name=new_model,
    version=model_version,
    stage="Staging"
)
```

**D.**
```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    stage="Staging"
)
```


**E.**
```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    stage="Staging"
)
```


**Answer: (SHOW ANSWER)**

**NEW QUESTION: 33**
Which of the following is a drawback associated with using Jensen-Shannon (JS) distance for numeric feature drift detection?
**A.** All of these reasons
**B.** JS requires a manual threshold or cutoff determinations
**C.** JS is not robust when working with large datasets
**D.** None of these reasons
**Answer: (SHOW ANSWER)**
A key drawback of using Jensen-Shannon (JS) distance for numeric feature drift detection is that it requires manual threshold tuning to determine what level of divergence indicates significant drift. This makes automation and consistent interpretation challenging, especially across features or datasets with differing distributions.

**NEW QUESTION: 34**
A Machine Learning Engineer has deployed a fraud detection model in Databricks Model Serving to detect fraudulent transactions. The engineer wants to compare the model's predictions with the actual fraud classifications from the Fraud Ops team to monitor model performance. The Fraud Ops team uses a unique transaction_id to investigate fraudulent activity and persist their findings to a fraud_findings table. The engineer enabled inference tables on the endpoint, but they are not sure how to map the models' predictions to the Fraud Ops team's classifications. How can the engineer uniquely join the models' prediction to the fraud_findings table with the fewest code changes?
**A.** Store databricks_request_id returned from each model serving request and persist it to the fraud_findings table. Join the inference table with the fraud_findings table using databricks_request_id as the join key.
**B.** Join the inference table with the fraud_findings table using timestamp_ms as the join key.
**C.** Populate the client_request_id field with the transaction_id in the model serving request body.

Join the inference table with the fraud_findings table using client_request_id (which contains the transaction_id) as the join key.

**D.** Modify the model to include an additional input: transaction_id. Log, register and deploy the new model. In the model serving request body, add transaction_id as an additional input feature. Join the inference table with the fraud_findings table using transaction_id as the join key.

**Answer: (SHOW ANSWER)**

Databricks Model Serving inference tables automatically log the client_request_id field for each request. By populating this field with the existing transaction_id in the request body, the engineer can directly and uniquely join inference predictions with the fraud_findings table using the same identifier, achieving accurate performance monitoring with minimal code changes and no model retraining or redeployment.

**NEW QUESTION: 35**

A machine learning engineer wants to move their model version model_version for the MLflow Model Registry model model from the Staging stage to the Production stage using MLflow Client client. Which code block can they use to accomplish the task?

**A.**

```
client.transition_model__stage(
    name=model,
    version=model_version,
    from="Staging",
    to="Production"
)
```

**B.**

```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    from="Staging",
    to="Production"
)
```

**C.**

```
client.transition_model_stage(
    name=model,
    version=model_
    version, stage="Production"
)
```

**D.**

```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    stage="Production"
)
```



**Answer: (SHOW ANSWER)**

The correct method for transitioning a model version to a new stage using the MLflow Client is client.transition_model_version_stage(name, version, stage). This updates the stage of a specific model version in the Model Registry, such as moving it from "Staging" to "Production". The stage parameter specifies the target stage only -- there is no from argument.

**NEW QUESTION: 36**

A machine learning engineering team has written predictions computed in a batch job to a Delta table for querying. However, the team has noticed that the querying is running slowly. The team has already tuned the size of the data files. Upon investigating, the team has concluded that the rows meeting the query condition are sparsely located throughout each of the data files. Based on the scenario, which optimization technique could speed up the query by colocating similar records while considering values in multiple columns?

**A.** Write as a Parquet file

**B.** Bin-packing

**C.** Z-Ordering

**D.** Data skipping

**E.** Tuning the file size

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 37**

A data scientist wants to remove the star_rating column from the Delta table at the location path. To do this, they need to load in data and drop the star_rating column. Which of the following code blocks accomplishes this task?

**A.** spark.read.format("delta").table(path).drop("star_rating")

**B.** Delta tables cannot be modified

**C.** spark.read.table(path).drop("star_rating")

**D.** spark.read.format("delta").load(path).drop("star_rating")

**E.** spark.sql("SELECT * EXCEPT star_rating FROM path")

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 38**

After a data scientist noticed that a column was missing from a production feature set stored as a Delta table, the machine learning engineering team has been tasked with determining when the

column was dropped from the feature set. Which SQL command can be used to accomplish this task?

**A.** VERSION

**B.** DESCRIBE

**C.** HISTORY

**D.** DESCRIBE HISTORY

**E.** TIMESTAMP

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 39**

A Data Scientist is preparing a Spark ML pipeline on a customer dataset with numeric features age, annual_income, and transaction_count, each varying widely. Because the chosen algorithm requires inputs normalized to the [0,1] range, they need to apply the appropriate Spark ML transformer to these features. Which Spark ML transformer should the Data Scientist use to scale all features to the [0,1] range?

**A.** MaxAbsScaler

**B.** RobustScaler

**C.** MinMaxScaler

**D.** StandardScaler

**Answer: (SHOW ANSWER)**

MinMaxScaler rescales each numeric feature to a fixed range, typically [0, 1], by subtracting the minimum value and dividing by the feature's range. This makes it the appropriate Spark ML transformer when an algorithm explicitly requires inputs normalized to the [0,1] interval.

**NEW QUESTION: 40**

A Data Scientist is building a propensity model for an e-commerce start-up. The company maintains 7GB of historical data and receives about 5MB of new transaction data daily. The goal is to generate daily purchase predictions for all users by 7:00 AM each morning. As the start-up is in its early stages, the data scientist must prioritize a highly cost-efficient approach. Which approach should the Data Scientist take?



**A.** Use a multi-node compute and leverage distributed frameworks like SparkML to train the model.

The model can then be scheduled as a nightly batch job that runs on a multi-node compute.

**B.** Use a single-node memory compute to build a model with libraries like scikit-learn. The model can then be scheduled as a nightly batch job that runs on a single-node compute.

**C.** Use a single-node compute to build a model with libraries like scikit-learn. The model can then be deployed as an always-on REST API so that users can query the API to get the predictions whenever they want.

**D.** Use a single-node compute to build a model with libraries like scikit-learn. The model can then be integrated into an always-on streaming pipeline, to ensure immediate processing of incoming data in order to meet the SLA.

**Answer: (SHOW ANSWER)**

With only 7GB of historical data and a small daily increment (about 5MB), a single-node memory-optimized cluster can comfortably train and score using scikit-learn without the overhead and cost of distributed compute. Scheduling a nightly batch job is the most cost-efficient way to meet a fixed daily SLA (7:00 AM) because the compute can be started only for the job run and then terminated, avoiding the expense of always-on serving or streaming infrastructure.

**NEW QUESTION: 41**
A machine learning engineer is in the process of implementing a concept drift monitoring solution. They are planning to use the following steps:
1. Deploy a model to production and compute predicted values
2. Obtain the observed (actual) label values
3. _____
4. Run a statistical test to determine if there are changes over time
Which of the following should be completed as Step #3?
**A.** Compute the evaluation metric using the observed and predicted values
**B.** Measure the latency of the prediction time
**C.** None of these should be completed as Step #3
**D.** Obtain the observed values (actual) feature values
**E.** Retrain the model
**Answer: (SHOW ANSWER)**

**NEW QUESTION: 42**
A machine learning engineer has registered a sklearn model in the MLflow Model Registry using the sklearn model flavor with UI model_uri. Which operation can be used to load the model as an sklearn object for batch deployment?
**A.** mlflow.sklearn.load_model(model_uri)
**B.** mlflow.pyfunc.load_model(model_uri)
**C.** mlflow.pyfunc.read_model(model_uri)
**D.** mlflow.spark.load_model(model_uri)
**E.** mlflow.sklearn.read_model(model_uri)
**Answer: (SHOW ANSWER)**

**NEW QUESTION: 43**
Which approach is best for scoring large historical datasets?
**A.** Real-time REST API
**B.** Batch inference with Spark
**C.** Manual predictions
**D.** Local Python script
**Answer: (SHOW ANSWER)**
Spark allows distributed batch scoring across massive datasets.

**NEW QUESTION: 44**

A Machine Learning Engineer is tasked with building an automated daily pipeline that updates a customer_features table in Unity Catalog. They have implemented a function, compute_customer_features, that returns a DataFrame with a unique customer_id as the primary key and want to ensure the latest feature values are merged into the table each day. Which code snippet implements this requirement?

```
from databricks.feature_engineering import FeatureEngineeringClient

fe = FeatureEngineeringClient()
customer_features_df = compute_customer_features(data)

fe.create_table(
    name='ml.recommender_system.customer_features',
    primary_keys='customer_id',
    df=customer_features_df
)
```

**A.**

**B.**

```
from databricks.feature_engineering import FeatureEngineeringClient

fe = FeatureEngineeringClient()
customer_features_df = compute_customer_features(data)

fe.write_table(
    df=customer_features_df,
    name='ml.recommender_system.customer_features',
    mode='overwrite'
)
```

**C.**

```
from databricks.feature_engineering import FeatureEngineeringClient

fe = FeatureEngineeringClient()

fe.write_table(
    name='ml.recommender_system.customer_features',
    df=customer_features_df
    # Default mode only
)
```

```
from databricks.feature_engineering import FeatureEngineeringClient

fe = FeatureEngineeringClient()
customer_features_df = compute_customer_features(data)

fe.write_table(
    df=customer_features_df,
    name='ml.recommender_system.customer_features',
    mode='merge'
)
```

**D.**

**Answer: (SHOW ANSWER)**

Using write_table with mode set to merge updates existing rows and inserts new ones based on the table's primary key. This ensures that the latest feature values for each customer_id are merged into the existing feature table each day without overwriting the entire table, which is the correct and scalable approach for maintaining up-to-date customer features in Unity Catalog.

**NEW QUESTION: 45**

A data scientist wants to track the runs of their random forest model. The data scientist is changing the number of trees and the maximum depth of the trees in the forest across each run. They write the following code block:

```
with mlflow.start_run(experiment_id=exp_id, run_name=run_name) as run:
    rf = RandomForestRegressor(**params)
    rf.fit(X, y)
    predictions = rf.predict(X_test)
    mlflow.log_params(params)
    return run.info.run_id
```

Which Python object type does params need to be an instance of?

**A.** array

**B.** PySpark DataFrarne

**C.** dict

**D.** list

**Answer: (SHOW ANSWER)**

The params variable must be a dictionary (dict) because mlflow.log_params() expects a dictionary where each key-value pair represents a parameter name and its corresponding value. Additionally, the model instantiation RandomForestRegressor(**params) also requires params to be a dictionary to unpack the parameters correctly.

**NEW QUESTION: 46**

A Machine Learning Engineer is setting up Databricks Lakehouse Monitoring to issue alerts based on IoT sensors that monitor various water conditions at a high tech hydroponic farm. Specifically:

- pH sensors

- Records acidity/alkalinity on a logarithmic scale from 0 to 14
- Pump status monitor
- Records the current status of the pump as "on", "off", "maintenance", "fault", or "clean"
- Electrical Conductivity (EC) sensors
- Records nutrient level of the water as "very rich", "rich", "medium", "poor", "very poor"

What are the appropriate drift metrics for each of the given sensor types?

**A.** pH: Kolmogorov-Smirnov (KS), pump status: Chi-squared, EC sensor: Jensen-Shannon distance.

**B.** pH: Chi-squared, pump status: Kolmogorov-Smirnov (KS), EC sensor: Wasserstein distance.

**C.** pH: Jenson-Shannon distance, pump status: Kolmogorov-Smirnov (KS), EC sensor: Chi-squared.

**D.** pH: Jensen-Shannon distance, pump status: Wasserstein distance, EC sensor: Kolmogorov-Smirnov (KS).

**Answer: (SHOW ANSWER)**

pH readings are continuous numerical values, so the Kolmogorov-Smirnov test is appropriate for detecting distribution shifts. Pump status is a nominal categorical variable, making the chi-squared test suitable for identifying changes in category frequencies. Electrical Conductivity levels are categorical with ordered labels, and Jensen-Shannon distance effectively measures distributional changes in such categorical probability distributions.



**NEW QUESTION: 47**

Which MLflow command logs a trained model?

**A.** mlflow.run()

**B.** mlflow.start_run()

**C.** mlflow.log_model()

**D.** mlflow.register_model()

**Answer: (SHOW ANSWER)**

mlflow.log_model() saves the model artifact within the run.

**NEW QUESTION: 48**
A Data Scientist is tasked with developing models to forecast product demand. The company offers 5000 different product types, and the Data Scientist must generate weekly forecasts for each type. They have access to two years of historical purchase data and are given ample project budget.
For their next project, they want to build 5000 separate Random Forest models, one for each product type. They aim to train all the models as quickly as possible with minimal setup.
Which approach meets these requirements?
**A.** Use the DeepSpeed library to distribute the data by product across different nodes to enable the parallel training of multiple models.
**B.** Create a Databricks Workflow with 5000 tasks. Each task is configured to accept a product ID as a parameter which will then train a model based on the specified product ID.
**C.** Leverage the pandas function API (Grouped map) to group the data by product type and apply a custom model training function to each group.
**D.** Use the RandomForest method from MLlib. This will leverage Spark's parallel processing capability to train 5000 different models.
**Answer: (SHOW ANSWER)**
The pandas function API with grouped map allows data to be grouped by product type and applies a custom training function independently to each group. This approach enables massive parallelism across the cluster with minimal orchestration or setup, making it well suited for rapidly training thousands of independent models in parallel.

**NEW QUESTION: 49**
A Machine Learning Engineer is working with a Spark DataFrame containing 100 million rows of retail transactions across thousands of stores. Each store requires its own demand forecasting model using the same scikit-learn pipeline. They want to train and apply these models in parallel for each store without collecting the data to the driver. Which approach will do this?
**A.** Use rdd.mapPartitions() to iterate through each store and apply the model logic in Python.
**B.** Use groupBy("store_id").applyInPandas() to train and apply the model per group using a Pandas UDF.
**C.** Use pyspark.pandas to call .groupby("store_id").apply() and train the model on each group.
**D.** Convert the DataFrame to a Pandas DataFrame using .toPandas() and loop through each store locally.
**Answer: (SHOW ANSWER)**
groupBy().applyInPandas() enables grouped Pandas UDFs that receive each store's data as a Pandas DataFrame on the executors. This allows training and applying a separate scikit-learn model per store in parallel without collecting data to the driver, making it the correct and scalable approach for large Spark DataFrames.

**NEW QUESTION: 50**

Label drift occurs where there is a change in which element?

**A.** The relationship between input variables and target variables

**B.** The distribution of the predicted target given by the model

**C.** The distribution of a target variable

**D.** The distribution of an input variable

**Answer: (SHOW ANSWER)**

Label drift refers to a change in the distribution of the target variable over time. This means the frequencies or proportions of classes or target values shift, which can impact model performance even if the input feature distributions remain unchanged.

**NEW QUESTION: 51**

A Machine Learning Engineer has automated a model retraining job in Databricks. Each scheduled run trains multiple candidate models with new sales data and logs all runs with MLflow. The goal is to select and register the best-performing model at the end of each cycle to ensure optimal forecast accuracy. Which approach will meet this goal?

**A.** Register the first model that has an evaluation metric better than the previously deployed model.

**B.** Register the best model based on the primary evaluation metric.

**C.** Register the model with the lowest training loss regardless of validation metrics.

**D.** Register all models from the run since all of them are trained on newer data and hence will be more accurate.

**Answer: (SHOW ANSWER)**

Selecting and registering the model that performs best on the primary evaluation metric ensures that only the highest-quality model is promoted at each retraining cycle. This approach aligns with MLOps best practices by basing promotion decisions on objective performance criteria rather than training order or assumptions about data freshness.

**NEW QUESTION: 52**

A Machine Learning Engineer uses Lakehouse Monitoring to track their credit scoring model's performance. The existing profile metrics table contains three aggregate metrics:

- adefault_risk_score
- payment_history_score
- credit_utilization_score

They need to:

1. Create a composite risk rating that combines these three scores using weights of 0.5, 0.3, and 0.2 respectively.
2. Monitor drift of this composite score against an established baseline.

Which approach should be used to implement both requirements within Lakehouse Monitoring?

**A.** Create two separate aggregate metrics: one for the composite score calculation and another for drift detection.

**B.** Use a scheduled notebook to calculate the composite score and manually insert both the score and drift values into the profile metrics table.
**C.** Create a derived metric for the composite_risk_rating calculation and create a drift metric on this derived metric.
**D.** Create an aggregate metric for composite_risk_rating and configure a separate drift metric to monitor changes.
**Answer: (SHOW ANSWER)**
Lakehouse Monitoring supports derived metrics that are computed from existing profile metrics using custom expressions. By defining a derived metric for the composite_risk_rating using the specified weights, the composite score becomes a first-class metric in the monitoring framework. A drift metric can then be directly configured on this derived metric to compare current values against the baseline, fulfilling both the composite calculation and drift monitoring requirements in a native, governed way.

**NEW QUESTION: 53**
A machine learning engineer would like to compute predictions on inference data as it becomes available through the pipeline in microbatches. The predictions should be stored in a table for query later. Which deployment strategy can the engineer use?
**A.** Real-time
**B.** Streaming
**C.** Batch
**D.** Edge/on-device
**Answer: (SHOW ANSWER)**
A streaming deployment strategy processes incoming data continuously or in microbatches, allowing predictions to be computed as new data arrives. The results can then be written to a table for later querying. This approach balances near-real-time inference with manageable resource use, making it ideal for scenarios where data flows steadily through a pipeline.

**NEW QUESTION: 54**
A data scientist wants to log outlier feature data from a CSV file at path outlier_path with an MLflow run for model model. Which code block will accomplish this task inside of an existing MLflow run block?
**A.** mlflow.log_artifact(outlier_path, "outlier-features.csv")
**B.** mlflow.log_data(outlier_path, "outlier-features.csv")
**C.**


**D.**

```
mlflow.log_model_and_data(
    model,
    outlier_path,
    "outlier-features.csv"
)
```

**Answer: (SHOW ANSWER)**

To log external files, such as a CSV containing outlier feature data, MLflow provides the mlflow.log_artifact() function. This function uploads the specified file or directory (outlier_path) as an artifact under the provided artifact path ("outlier-features.csv"). It is the correct way to associate data files with an MLflow run, whereas mlflow.log_model() is reserved for logging model objects, not arbitrary data.

**NEW QUESTION: 55**

Which of the following describes batch deployment for machine learning projects?

**A.** Predictions are computed and delivered as soon as feature values are available

**B.** None of these describe batch deployment for machine learning projects

**C.** Predictions are computed prior to delivery and stored for later querying

**D.** Predictions are computed immediately as data arrives and stored for later querying

**Answer: (SHOW ANSWER)**

In batch deployment, predictions are precomputed at scheduled intervals and stored for later querying. This approach is ideal when real-time inference is not required and when predictions can be generated in bulk ahead of time.

**NEW QUESTION: 56**

A machine learning engineer wants to move their model version model_version for the MLflow Model Registry model model from the Staging stage to the Production stage using MLflow Client client. At the same time, they would like to archive any model versions that are already in the Production stage. Which code block can they use to accomplish the task?

**A.**

```
client.transition_model_stage(
    name=model,
    version=model_version,
    stage="Archived"
)
client.transition_model_stage(
    name=model,
    version=model_version,
    stage="Production"
)
```

**B.**

```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    stage="Production",
    archive_existing_versions=True
)
```

**C.**

```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    stage="Archived"
)
client.transition_model_version_stage(
    name=model,
    version=model_version,
    stage="Production"
)
```

**D.**

```
client.transition_model_stage(
    name=model,
    version=model_version,
    stage="Production",
    archive_existing_versions=True
)
```

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 57**

A Data Scientist needs to perform inference on a continuously updated Delta table called sales_data using an MLflow-registered Spark ML pipeline model (catalog.prod.sales_forecaster). Predictions must be written to a Delta table forecast_results, which must be updated with low latency leveraging a cluster with three executors. They want to maximize the efficient use of their cluster when doing this. Which approach will suit their needs?

**A.**

```
model_uri = "models:/catalog.prod.sales_forecaster@latest"
predict_udf = mlflow.pyfunc.spark_udf(spark, model_uri)
scored_df = spark.table("sales_data").withColumn("prediction", predict_udf(*feature_cols))
scored_df.write.saveAsTable("forecast_results")
```

**B.**

```
model = mlflow.pyfunc.load_model("models:/catalog.prod.sales_forecaster/latest")
def predict_batch(batch_df, batch_id):
    pandas_df = batch_df.toPandas()
    pandas_df["prediction"] = model.predict(pandas_df[feature_cols])

(spark.readStream.table("sales_data")
      .writeStream
      .foreachBatch(predict_batch)
      .outputMode("append")
      .option("checkpointLocation", "/Volumes/catalog/prod/checkpoints/forecast")
      .toTable("forecast_results")
      .start())
```

**C.**

```
client = mlflow.deployments.get_deploy_client("databricks")
scored_df = client.predict("sales_forecaster-endpoint", spark.table("sales_data"))
scored_df.write.saveAsTable("forecast_results")
```

**D.**

```
model_udf = mlflow.pyfunc.spark_udf(spark, "models:/catalog.prod.sales_forecaster@latest")
scored_df = spark.readStream.table("sales_data")
            .withColumn("prediction", model_udf(struct(*feature_cols)))
(scored_df.writeStream
  .format("delta")
  .outputMode("append")
  .option("checkpointLocation", "/Volumes/catalog/prod/checkpoints/forecast")
  .toTable("forecast_results")
  .start())
```



**Answer: (SHOW ANSWER)**

This approach uses a Spark Structured Streaming read from the continuously updated Delta table and applies an MLflow-registered Spark UDF for inference. The model execution is distributed across the three executors, enabling parallel, low-latency scoring as new data arrives. Writing the results with writeStream efficiently updates the forecast_results Delta table incrementally, maximizing cluster utilization and aligning with best practices for continuous, scalable batch-stream inference in Databricks.

**NEW QUESTION: 58**

A Machine Learning Engineer is implementing integration tests for an ML pipeline in Databricks. The current integration test runs the complete workflow but takes four hours to execute due to large dataset processing and extensive model training. They need to select an approach that will be the most effective for optimizing integration test execution while maintaining test reliability. The approach should also be based on MLOps best practices. Which approach will do this?

**A.** Run integration tests only in the production environment using full datasets to ensure complete accuracy of the testing process.

**B.** Use small subsets of production-like data and reduce training iterations while maintaining the same pipeline structure and validation checkpoints in a staging environment that closely matches production.

**C.** Skip the model training phase entirely and only test data transformations and API endpoints using mock model predictions.

**D.** Replace integration tests with unit tests for each pipeline component to reduce execution time and focus on individual component validation in a staging environment that closely matches production.

**Answer: (SHOW ANSWER)**

Using smaller, production-like datasets and reduced training iterations preserves the full pipeline structure while significantly reducing execution time. This aligns with MLOps best practices by maintaining high-fidelity integration testing in a staging environment that mirrors production behavior, without the cost and delay of running full-scale training workloads.

**NEW QUESTION: 59**

In a Spark ML Pipeline, what is the role of a Transformer?

**A.** Fits data

**B.** Converts datasets into new datasets

**C.** Trains models

**D.** Evaluates models

**Answer: (SHOW ANSWER)**

A Transformer takes a dataset and transforms it into another dataset.

**NEW QUESTION: 60**

A machine learning engineer needs to select a deployment strategy for a new machine learning application. The feature values are not available until the time of delivery, and results are needed exceedingly fast for one record at a time. Which of the following deployment strategies can be used to meet these requirements?



**A.** Real-time

**B.** Batch

**C.** Streaming

**D.** Edge/on-device

**E.** None of these strategies will meet the requirements.

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 61**

A Machine Learning Engineer is considering moving their functions and unit tests from notebooks into separate Python files (e.g., modules and test scripts) to take advantage of the numerous benefits of this approach like automated execution, code reusability, and version control. Which challenge should the engineer consider with this approach?

**A.** This change would prevent the use of any non-Python environments like Scala.

**B.** Separating code and tests often leads to decreased reliability and poor code quality.

**C.** Managing a more complex project structure can be harder to maintain and navigate.

**D.** It makes functions harder to import and reuse across different notebooks.

**Answer: (SHOW ANSWER)**

Moving code and unit tests into separate Python modules introduces a more structured project layout, which can increase complexity. Engineers must manage directories, dependencies, and imports carefully, making the project slightly harder to navigate and maintain compared to simple notebook-based workflows, especially for teams new to this structure.

**NEW QUESTION: 62**
A data scientist has developed a scikit-learn random forest model model, but they have not yet logged model with MLflow. They want to obtain the input schema and the output schema of the model so they can document what type of data is expected as input. Which of the following MLflow operations can be used to perform this task?
**A.** mlflow.models.Model.get_input_schema
**B.** mlflow.models.Model.signature
**C.** mlflow.models.schema.infer_schema
**D.** mlflow.models.signature.infer_signature
**E.** There is no way to obtain the input schema and the output schema of an unlogged model.
**Answer: (SHOW ANSWER)**

**NEW QUESTION: 63**
A machine learning engineer needs to select a deployment strategy for a new machine learning application. The machine learning application requires central prediction computation and exceedingly fast results, but only a handful of predictions need to be computed at a time. Which deployment strategy can be used to meet these requirements?
**A.** Streaming
**B.** Real-time
**C.** Batch
**D.** Edge/on-device
**Answer: (SHOW ANSWER)**
Real-time deployment is the appropriate strategy when predictions need to be computed centrally with very low latency, even if only a small number of predictions are required at a time. This ensures fast responses for applications requiring immediate inference.

**NEW QUESTION: 64**
In a continuous integration, continuous deployment (CI/CD) process for machine learning pipelines, which of the following events commonly triggers the execution of automated testing?
**A.** The launch of a new cost-efficient SQL endpoint
**B.** The launch of a new cost-efficient job cluster
**C.** The arrival of a new feature table in the Feature Store
**D.** The arrival of a new model version in the MLflow Model Registry
**E.** CI/CD pipelines are not needed for machine learning pipelines
**Answer: (SHOW ANSWER)**

**NEW QUESTION: 65**

Which statement describes streaming with Spark as a model deployment strategy?
**A.** The inference of all types of records in real-time
**B.** The inference of incrementally processed records as soon as a Spark job is run
**C.** The inference of incrementally processed records as soon as trigger is hit
**D.** The inference of batch processed records as soon as a Spark job is run
**E.** The inference of batch processed records as soon as a trigger is hit
**Answer: (SHOW ANSWER)**

**NEW QUESTION: 66**
A Data Scientist is training a binary classification model using LogisticRegression in SparkML on a large dataset stored in a Delta table. After fitting the pipeline, they want to evaluate the model's performance using an appropriate metric and scalable method across the distributed test data using the SparkML API. Which model evaluation strategy will suit their needs?
**A.** Use BinaryClassificationEvaluator with rawPredictionCo1 parameter, but first convert probability predictions to binary predictions using a UDF to ensure consistent evaluation across different Spark versions.
**B.** Convert the DataFrame to RDD and use MulticlassMetrics.confusionMatrix() to calculate precision and recall manually, as this avoids the overhead of DataFrame operations in distributed environments.
**C.** Use the BinaryClassificationEvaluator on the transformed test DataFrame and specify the appropriate metric.
**D.** Use MulticlassClassificationEvaluator with weightedPrecision metric since it provides more comprehensive evaluation than BinaryClassificationEvaluator for binary problems.
**Answer: C (LEAVE A REPLY)**
BinaryClassificationEvaluator is the Spark ML-native, distributed evaluation API designed specifically for binary classification models. It operates directly on the transformed DataFrame produced by the pipeline and efficiently computes scalable metrics such as areaUnderROC or areaUnderPR without requiring data conversion or custom logic, making it the correct and efficient choice for large, distributed datasets.

**NEW QUESTION: 67**
A Machine Learning Engineer is conducting hyperparameter tuning for multiple XGBoost models using Ray Tune on Databricks. They want to integrate MLflow tracking to monitor their experiments and need to ensure proper authentication. The engineer has Ray 2.41 installed and wants to use both Ray Tune and MLflow together in their distributed tuning workflow. They have to configure Databricks to run the hyperparameter optimization with MLflow integration. Which set of configuration steps will do this?
**A.** Install the MLflow Ray plugin using %pip install mlflow-ray and configure the workspace connection.
**B.** Configure DATABRICKS_HOST and DATABRICKS_TOKEN environment variables before calling setup_ray_cluster().

**C.** Enable MLflow autologging with mlflow.ray.autolog() and set the tracking server URI.
**D.** Set MLFLOW_TRACKING_URI and MLFLOW_EXPERIMENT_TD environment variables before initializing Ray.
**Answer: (SHOW ANSWER)**
When using Ray Tune with MLflow on Databricks, Ray workers must be able to authenticate back to the Databricks workspace to log runs to MLflow Tracking. Setting the DATABRICKS_HOST and DATABRICKS_TOKEN environment variables before initializing the Ray cluster ensures all Ray processes can securely communicate with Databricks and correctly log MLflow experiments during distributed hyperparameter tuning.

**NEW QUESTION: 68**
A Data Scientist has trained a regression model using Spark's MLlib to predict housing prices. The model needs to be evaluated to determine how well it predicts on a testing dataset. The Data Scientist wants to use a standard regression metric for the evaluation. They also want to evaluate the model with the least processing overhead required. Which approach will suit their needs?
**A.** Convert the testing dataset to a pandas DataFrame and use scikit-learn's root_mean_squared_error function to calculate the root mean squared error on the testing dataset.
**B.** Use the MLlib's RegressionEvaluator with the metricName set to rmse to calculate root mean squared error on the testing dataset.
**C.** Implement a Spark UDF to calculate the root mean squared error for each row in the testing dataset and aggregate these values to obtain the overall RMSE score.
**D.** Use the MLlib's RankingEvaluator with the metricName set to rmse to calculate root mean squared error on the testing dataset.
**Answer: (SHOW ANSWER)**
RegressionEvaluator is the Spark ML-native, distributed evaluation API for regression models. Setting metricName to rmse computes root mean squared error directly on the distributed test DataFrame with minimal overhead, avoiding unnecessary data movement or custom computation logic.

**NEW QUESTION: 69**
A Data Scientist needs to analyze drift detection results from Databricks Lakehouse Monitoring. The system has generated both profile metrics and drift metrics tables. The scientist needs to identify baseline drift in numerical features by comparing current data against a baseline from 6 months ago. Which combination of table columns and values indicates baseline drift in a numerical feature?
**A.** window_cmp pointing to baseline time window and tv_distance > 0.5 with any drift_type value.
**B.** drift_type = "BASELINE", chi_squared_test.p_value < 0.05, and js_distance > 0.2.
**C.** drift_type = "BASELINE", ks_test.p_value < 0.05, and wasserstein_distance > 0.1.
**D.** log_type = "BASELINE" in profile metrics table with population_stability_index > 0.2 in drift metrics table.

**Answer: (SHOW ANSWER)**

Baseline drift is identified in the drift metrics table by setting drift_type to BASELINE, and for numerical features the drift statistics are the KS test (ks_test with a p-value indicating a statistically significant distribution change) and a numeric distance metric such as wasserstein_distance.

**NEW QUESTION: 70**

A Machine Learning Engineer has a large dataset with a customer_region column and wants to train separate models for each region, then generate predictions. They need to parallelize this group-specific model training process using Databricks and the Pandas Function API. Which approach will implement this solution?



**A.** Use collect() to gather all data and process regions using standard pandas groupby.

**B.** Use groupBy("customer_region") and apply a training function with applyInPandas().

**C.** Use foreachBatch() to sequentially process each region's data.

**D.** Use mapInPandas() to apply the training function across all partitions without grouping.

**Answer: (SHOW ANSWER)**

The Pandas Function API supports parallel, group-specific processing by using groupBy on the grouping column and applyInPandas to execute a custom training function independently for each group. This enables separate models to be trained per region in parallel across the cluster, with each function invocation receiving only the data for its group.

**NEW QUESTION: 71**

A machine learning engineer has created a webhook with the following code block:

```
job_json = {
    "model name": model,
    "events": ["MODEL_VERSION_TRANSITIONED_TO_PRODUCTION"],
    "description": "Job webhook tigger",
    "status": "Active",
    "job_spec": {
        "job_id": job_id,
        "workspace_url": url,
        "access_token": token
    }
}

response = http_request(
    host_creds=host_creds,
    endpoint=endpoint,
    method="POST",
    json=job_json
)
```

Which code block will trigger this webhook to run the associate job?

A.
```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    stage="Production"
)
```

B.
```
client.transition_model_version_stage(
    name=new_model,
    version=model_version,
    stage="Production"
)
```

C.
```
client.transition_model_version_stage(
    name=model,
    version=model_version,
    from="None",
    to="Production"
)
```

D.
```
client.transition_model_stage(
    name=new_model,
    version=model_version,
    stage="Production"
)
```

**Answer: (SHOW ANSWER)**

The webhook in the first image is configured to trigger on the MODEL_VERSION_TRANSITIONED_TO_PRODUCTION event. The chosen code uses client.transition_model_version_stage(...) to move the model version to "Production", which matches the event type in the webhook and will correctly trigger the associated job.

**NEW QUESTION: 72**

A Machine Learning Engineer has a computer vision model in Databricks Model Serving that obscures sensitive data from images. Internal teams use the model throughout the work week when they request access to new files. Recently model users complained that the model takes much longer in the morning. This coincides with when people arrive at work and request files for

the day. When the engineer reviews the endpoint health metrics, they see P50 model latency peaks around 9AM at 20 seconds. Request rate also peaks at 9AM at 15 requests/second. The GPU utilization is over 60%, GPU memory usage over 50%, and provisioned concurrency at 4 throughout the day. What can the engineer do to reduce user wait time when request rate peaks at 9AM each morning?

**A.** The endpoint is constrained by its ability to handle simultaneous requests. Enable "scale_to_zero" on the endpoint to spin up additional endpoints quickly to handle the requests.

**B.** The endpoint is constrained by its ability to handle simultaneous requests. Scale the endpoint horizontally by editing the endpoint workload_size from "Small" to "Medium" to increase the model concurrency.

**C.** The endpoint is constrained by its ability to handle simultaneous requests. Define a rate_1imit on the endpoint to spread out the requests over the course of the day.

**D.** The endpoint is constrained by the GPU. Scale the endpoint vertically by changing the endpoint workload_type from "GPU_SMALL" to "GPU_MEDIUM".

**Answer: (SHOW ANSWER)**

The symptoms indicate a concurrency bottleneck during the 9AM traffic spike: request rate increases sharply, latency jumps, and the endpoint is already running at a fixed provisioned concurrency of 4. Increasing workload_size scales the endpoint horizontally to handle more concurrent requests, reducing queueing and bringing down user-perceived wait time during peak demand.

**NEW QUESTION: 73**

A machine learning engineer wants to log and deploy a model as an MLflow pyfunc model. They have custom preprocessing that needs to be completed on feature variables prior to fitting the model or computing predictions using that model. They decide to wrap this preprocessing in a custom model class ModelWithPreprocess, where the preprocessing is performed when calling fit and when calling predict. They then log the fitted model of the ModelWithPreprocess class as a pyfunc model. Which statement is a benefit of this approach when loading the logged pyfunc model for downstream deployment?

**A.** The pvfunc model can be used to deploy models in a parallelizable fashion

**B.** There is no longer a need for pipeline-like machine learning objects

**C.** The same preprocessing logic will automatically be applied when calling predict

**D.** The same preprocessing logic will automatically be applied when calling fit

**E.** This approach has no impact when loading the logged Pvfunc model for downstream deployment

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 74**

A Machine Learning Engineer is building an application that requires low latency data lookups in response to a user's question following a RAG based search. They want to ensure their users can receive as recent data as possible for urgent requests, so data should not be more than a few

minutes late. The underlying data is a large table that may contain hundreds of gigabytes of data. Which data serving approach will suit their use case?

**A.** Online tables with snapshot sync mode

**B.** A fast database hosted in MLflow model serving

**C.** Online tables with continuous sync mode

**D.** Online tables with triggered mode and a time series key

**Answer: (SHOW ANSWER)**

Online tables with continuous sync mode are designed for low-latency serving while keeping data fresh within minutes. Continuous sync incrementally propagates updates from the large underlying table to the online store, ensuring near-real-time availability for RAG-based lookups without requiring full refreshes, which is ideal for urgent, freshness-sensitive queries on large datasets.

**NEW QUESTION: 75**

Why is Apache Spark useful for machine learning training?

**A.** GPU rendering

**B.** Distributed data processing

**C.** Web serving

**D.** Data visualization

**Answer: (SHOW ANSWER)**

Spark processes large distributed datasets efficiently.

**NEW QUESTION: 76**

Which tool can assist in real-time deployments by packaging software with its own application, tools, and libraries?

**A.** Click

**B.** Structured Streaming

**C.** Docker

**D.** Flask

**Answer: (SHOW ANSWER)**

Docker is a containerization tool that packages software along with its dependencies, tools, and libraries into a single container. This ensures consistency across environments and is highly useful for real-time deployments, enabling scalable and portable model serving.

**NEW QUESTION: 77**
A Data Scientist is building a predictive maintenance model for a fleet of vehicles. They have two tables in their feature store:
1. A sensor_readings feature table with IoT data (e.g., engine_temp, oil_pressure) streamed continuously. This is a time-series table with vehicle_id as a primary key and ts as a timestamp key.
2. A maintenance_logs ground truth table that records when a vehicle component failed. This table includes vehicle_id and the exact failure_ts timestamp.
The goal is to create a training dataset by joining sensor_readings to maintenance_logs to train a model that predicts failures. They want to join the feature data with the ground truth data to ensure point-in-time correctness and prevent data leakage during model training.
Which approach will do this?
**A.** Perform an inner join on vehicle_id and select the sensor reading with the timestamp that is closest to the failure_ts, whether it occurred before or after the failure.
**B.** Join the tables on vehicle_id and then manually filter to only include sensor readings that were recorded in the hour immediately preceding the failure_ts.
**C.** Perform an "AS OF" join, using the failure_ts from the maintenance logs to look up the latest sensor reading values that were recorded at or before that specific timestamp.
**D.** For each failure, calculate the average of all sensor readings for that vehicle on the day of the failure and join that to the maintenance_logs table.
**Answer: (SHOW ANSWER)**
An AS OF join ensures point-in-time correctness by retrieving the most recent sensor readings that occurred at or before the failure timestamp for each vehicle. This guarantees that only information available prior to the failure event is used for training, preventing data leakage and aligning with best practices for time-series feature joins in predictive maintenance models.

**NEW QUESTION: 78**
A Data Scientist at an online gaming company is creating a model to predict player churn. The company currently collects terabytes of player activity logs daily, which are stored in Databricks and processed for daily reporting. The Data Scientist has completed feature engineering and the resulting data is saved as a Delta Table with a size of 500GB. They need to next build the model for the most performant and cost-effective performance for Databricks. Which approach will do this?
**A.** Load the feature data as a Spark DataFrame and train the model using Spark's DeepspeedTorchDistributor on a multi-node Databricks cluster.

**B.** Load the feature data as a pandas DataFrame and train the model using scikit-learn's RandomForestClassifier on a multi-node Databricks cluster.
**C.** Load the feature data as a pandas DataFrame and train the model using scikit-learn's RandomForestClassifier on a single-node Databricks cluster.
**D.** Load the feature data as a Spark DataFrame and train the model using SparkML's RandomForestClassifier on a multi-node Databricks cluster.
**Answer: (SHOW ANSWER)**
A 500GB Delta Table is far beyond what is practical to load into a single pandas DataFrame, and scaling pandas-based scikit-learn training across nodes is not the right fit for this workload. Using a Spark DataFrame with Spark ML's RandomForestClassifier leverages distributed data processing and distributed model training on a multi-node cluster, which is the most performant and cost-effective approach for large tabular datasets in Databricks.

**NEW QUESTION: 79**
A data scientist has developed a scikit-learn model sklearn_model and they want to log the model using MLflow.
They write the following incomplete code block:

```
with mlflow.start_run(experiment_id=exp_id, run_name=run_name) as run:
    # Log model
    _____
```

Which lines of code can be used to fill in the blank so the code block can successfully complete the task?
**A.** mlflow.sklearn.load_model("model")
**B.** mlflow.sklearn.track_model(sklearn_model, "model")
**C.** mlflow.spark.log_model(sklearn_model, "model")
**D.** mlflow.sklearn.log_model(sklearn_model, "model")
**E.** mlflow.spark.track_model(sklearn_model, "model")
**Answer: (SHOW ANSWER)**

**NEW QUESTION: 80**
A machine learning engineer has developed the following custom model class with preprocessing logic to combine two columns:

```
class RFWithPreprocess(mlflow.pyfunc.PythonModel):

    det __init__(self, params):
        self.params = params
        self.rf_model = None

    def preprocess_input(self, model_input):
        input = model_input.copy()
        input["price"] = input["spend"] / input["units"]
        return input

    def fit(self, X_train, y_train):
        from sklearn.ensemble import RandomForestRegressor
        input = self.preprocess_input(X_train)
        rf_model = RandomForestRegressor(**self.params)
        rf_model.fit(input, y_train)
        self.rf_model = rf_model

    def predict(self, model_input):
        input = model_input.copy()
        return self.rf_model.predict(input)
```



However, instances of this class are unable to compute predictions.

Which set of changes will update the class so predictions can be computed while continuing to apply the preprocessing logic?

**A.** Replace model_input.copy() with self.preprocess_input(model_input.copy()) in the preprocess_input method

**B.** Remove the self.rf_model = rf_model line from the fit method

**C.** Replace model_input.copy() with self.preprocess_input(model_input.copy()) in the predict method

**D.** Replace self.rf_model.predict(input) with self.predict(input) in the predict method

**Answer: (SHOW ANSWER)**

The issue is that the predict() method does not apply the same preprocessing as the fit() method. During training, the model uses preprocessed data (via self.preprocess_input()), but during prediction, it directly uses raw input. This mismatch causes prediction errors because the model expects preprocessed input.

By updating the predict() method to call self.preprocess_input(model_input.copy()), both training and prediction use consistent feature transformations, allowing predictions to be computed successfully.

**NEW QUESTION: 81**

A Machine Learning Engineer is building a Databricks ML pipeline to predict customer churn. The pipeline needs to include automated feature engineering, model training, evaluation, and

deployment to a REST API endpoint using MLflow. What is the primary goal of an integration test for this pipeline?

**A.** To verify that the pipeline can process a variety of data formats, even if some stages are skipped or bypassed

**B.** To ensure that the pipeline's performance meets latency and scalability requirements under simulated production loads

**C.** To monitor the model's performance after it has been deployed into the production environment

**D.** To ensure the pipeline runs with all components working together and data flowing correctly between stages

**Answer: (SHOW ANSWER)**

The primary purpose of an integration test is to validate that all components of the ML pipeline work together as expected. This includes confirming that data flows correctly through feature engineering, training, evaluation, and deployment steps, ensuring the end-to-end pipeline functions properly as a cohesive system.

**NEW QUESTION: 82**

A data scientist has computed updated feature values for all primary key values stored in the Feature Store table features. In addition, feature values for some new primary key values have also been computed. The updated feature values are stored in the DataFrame features_df. They want to replace all data in features with the newly computed data. Which of the following code blocks can they use to perform this task using the Feature Store Client fs?

**A.**
```
fs.create_table(
    name="features",
    df=features_df,
    mode="overwrite"
)
```

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

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

**D.**

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

**E.**

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

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 83**
How can you save a trained Spark ML PipelineModel?
**A.** pipeline.save()
**B.** pipelineModel.write().save()
**C.** pipeline.persist()
**D.** pipeline.store()
**Answer: B (LEAVE A REPLY)**
Example:
pipelineModel.write().overwrite().save("/model")

**NEW QUESTION: 84**
A machine learning engineer wants to programmatically create a new Databricks Job whose schedule depends on the result of some automated tests in a machine learning pipeline. Which Databricks tool can be used to programmatically create the Job?
**A.** AutoML APIs
**B.** MLflow Client
**C.** Jobs cannot be created programmatically
**D.** Databricks REST APIs
**E.** MLflow APIs
**Answer: (SHOW ANSWER)**

**NEW QUESTION: 85**
Which of the following describes the concept of MLflow Model flavors?
**A.** A convention that MLflow Experiments can use to organize their Runs by project
**B.** A convention that deployment tools can use to wrap preprocessing logic into a Model
**C.** A convention that deployment tools can use to understand the model
**D.** A convention that MLflow Model Registry can use to version models
**E.** A convention that MLflow Model Registry can use to organize its Models by project

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 86**
Which of the following is a probable response to identifying drift in a machine learning application?
**A.** None of these responses
**B.** All of these responses
**C.** Rebuilding the machine learning application with a new label variable
**D.** Retraining and deploying a model on more recent data
**E.** Sunsetting the machine learning application
**Answer: (SHOW ANSWER)**

**NEW QUESTION: 87**
Which statement is a reason for using Jensen-Shannon (JS) distance over a Kolmogorov-Smirnov (KS) test for numeric feature drift detection?
**A.** JS is more robust when working with large datasets
**B.** None of these reasons
**C.** All of these reasons
**D.** JS is not normalized or smoothed
**E.** JS does not require any manual threshold or cutoff determinations
**Answer: (SHOW ANSWER)**

**NEW QUESTION: 88**
A Machine Learning Engineer is building a fraud detection model that needs to use both pre-computed features from a feature table and real-time calculated features based on user location data sent with each inference request. The engineer has created a Python UDF called calculate_distance in Unity Catalog at main.fraud_detection.calculate_distance that computes the distance between a transaction location and the user's current location. The feature table main.fraud_detection.user_features contains historical user spending patterns with primary key user_id.
The engineer has written the following code to implement this scenario:

```
from databricks.feature_engineering import FeatureEngineeringClient, FeatureFunction, FeatureLookup
import mlflow

fe = FeatureEngineeringClient()
features = [
    FeatureFunction(
        udf_name="main.fraud_detection.calculate_distance",
        input_bindings={"user_lat": "current_latitude", "user_lon": "current_longitude", "trans_lat":
"transaction_latitude", "trans_lon": "transaction_longitude"},
        output_name="distance_to_transaction"
    ),
    FeatureLookup(
        table_name="main.fraud_detection.user_features",
        feature_names=["avg_spending", "transaction_count"],
        lookup_key="user_id"
    )
]

training_set = fe.create_training_set(df=base_df, feature_lookups=features, label="is_fraud")
model = train_model(training_set.load_df())

mlflow.set_registry_uri("databricks-uc")
fe.log_model(model=model, artifact_path="fraud_model", flavor=mlflow.sklearn, training_set=training_set,
registered_model_name="main.fraud_detection.fraud_model")
```



Which benefit of this implementation approach makes it suited to the real-time fraud detection use case?

**A.** The Unity Catalog registry automatically creates REST API endpoints for UDF functions used in feature computation.

**B.** The FeatureLookup function avoids the need for joining the full table main.fraud_detection.user_features with training_set increasing efficiency.

**C.** The model automatically performs feature lookup and computation during inference without additional serving code.

**D.** The FeatureFunction caches computed distance values in the online store to improve inference latency for location pairs.

**Answer: (SHOW ANSWER)**

By defining both FeatureLookup and FeatureFunction objects in the training set and logging the model with the FeatureEngineeringClient, the feature logic is packaged with the model. During inference, Databricks automatically performs feature lookups from the feature table and computes the on-demand distance feature using request-time inputs, without requiring any additional custom serving or feature-joining code. This makes the approach well suited for real-time fraud detection.

**NEW QUESTION: 89**

Which of the following Databricks-managed MLflow capabilities is a centralized model store?

**A.** Feature Store

**B.** Models

**C.** Experiments

**D.** Model Serving

**E.** Model Registry

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 90**

A data scientist is utilizing MLflow to track their machine learning experiments. After completing a run with run ID run_id for the experiment with experiment ID exp_id, the data scientist wants to programmatically return the logged metrics for run_id. They have an active MLflow Client client and an active Spark session spark. Which lines of code can be used to return the logged metrics for run_id?

**A.** client.qet_run(exp_id.run_id).data.metrics

**B.**
```
mlflow.search_runs(
    exp_id,
    orderBy = [metrics.rmse DESC"]
)[0]
```

**C.** client.get_run(run_id).data.metrics

**D.** spark.read.format("mlflow-run").load(run_id)

**Answer: C (LEAVE A REPLY)**

The correct way to retrieve logged metrics for a specific run using the MLflow Client is client.get_run(run_id).data.metrics. This returns a dictionary of all metrics logged for that run. The method in the image (mlflow.search_runs(...)) is for querying across multiple runs, not for accessing a specific run's metrics.

**NEW QUESTION: 91**

A machine learning engineer is converting a Hyperopt-based hyperparameter tuning process from manual MLflow logging to MLflow Autologging. They notice that not all details and objects are automatically logged, and they will need to manually log some things. Which of the following will need to be manually logged when performing nested runs with Hyperopt and MLflow Autologging?

**A.** Trial models

**B.** Trial status

**C.** Best trial evaluation metric

**D.** Hyperparameter values

**E.** Evaluation metrics

**Answer: (SHOW ANSWER)**

When using MLflow Autologging with Hyperopt and nested runs, the best trial evaluation metric is not automatically logged and must be logged manually. Autologging captures trial-level details like hyperparameters and evaluation metrics, but summarizing and logging the overall best trial's result is a manual responsibility of the engineer.

**NEW QUESTION: 92**
A machine learning engineer wants to log feature importance data from a CSV file at path importance_path with an MLflow run for model model. Which code block will accomplish this task inside of an existing MLflow run block?

**A.**

```
mlflow.log_model(
    model,
    importance_path,
    "feature-importance.csv"
)
```

**B.** mlflow.log_data(importance_path, "feature-importance.csv")

**C.**

```
mlflow.log_model_and_data(
    model,
    importance_path,
    "feature-importance.csv"
)
```

**D.** None of these code blocks tan accomplish the task.

**E.** mlflow.log_artifact(importance_path, "feature-importance.csv")

**Answer: (SHOW ANSWER)**

**NEW QUESTION: 93**
Why is Delta Lake time travel useful in ML pipelines?

**A.** Faster model inference

**B.** Reproducible training datasets

**C.** Smaller datasets

**D.** Model tuning

**Answer: (SHOW ANSWER)**

Time travel allows training on exact historical datasets.

**NEW QUESTION: 94**
Why are Delta tables often used to store machine learning features?
**A.** They support schema enforcement and time travel
**B.** They allow faster GPU training
**C.** They reduce model size
**D.** They replace Spark DataFrames
**Answer: (SHOW ANSWER)**
Delta Lake provides:
ACID transactions
time travel
schema enforcement
These are essential for reproducible ML pipelines.

**NEW QUESTION: 95**
A machine learning engineer has developed a model and registered it using the FeatureStoreClient fs. The model has model URI model_uri. The engineer now needs to perform batch inference on the training set logged with the model, but a few of the feature values in the column spend have since been updated and arc present in the customer-level Spark DataFrame spark_df. The customer_id column is the primary key of spark_df and the training set used when training and logging the model. Which code block can be used to compute predictions for the training set while overwriting its old spend values with the new spend values from spark_df?
**A.** fs.score_batch(model_uri, spark_df)
**B.** fs.score_model(model_uri, spark_df)
**C.** df = fs.get_updated_feature(spark_df, model=uri)
fs.score_batch(model_uri, df)
**D.** df = fs.get_updated_features(spark_df)
fs.score_batch(model_uri, df)
**Answer: (SHOW ANSWER)**
To perform batch inference while incorporating updated feature values (like spend) from a DataFrame (spark_df), the correct approach is to use fs.get_updated_features(spark_df) to refresh the relevant features based on the primary key (customer_id), then score the model using fs.score_batch(...). This ensures predictions are made with the latest data.

**NEW QUESTION: 96**
Which MLflow operation can be used to log a plot that was generated during an MLflow run?
**A.** None of these operations can accomplish the task.
**B.** mlflow.log_metric
**C.** mlflow.log_visualization
**D.** mlflow.log_figure

**Answer: (SHOW ANSWER)**

The mlflow.log_figure() operation is used to log plots or figures generated during an MLflow run. It takes a Matplotlib, Plotly, or similar figure object and saves it as an artifact in the MLflow tracking server, enabling visualization and later retrieval from the MLflow UI.