ML Associate - Mock Test 2

Domain: Databricks Machine Learning (13 Questions)

Question 1

A data science team is comparing models from a recent Databricks AutoML run. They want to find the best-performing model based on the F1 score, but also want to ensure the model is interpretable. Which generated model should they investigate first?

- A. The scikit-learn (XGBoost) model, as it has the highest F1 score and is a "glass box."
- B. The Decision Tree model, as it is highly interpretable even if its F1 score is slightly lower.
- C. The LightGBM model, as it is always the most accurate.
- D. The scikit-learn (Logistic Regression) model, as it is a "black box" model.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: AutoML generates notebooks for all trial runs. While XGBoost/LightGBM often have the best performance, Decision Trees are inherently interpretable ("glass box"). The notebook for the Decision Tree model will include a visualization of the tree, making it the best choice to satisfy the interpretability requirement.

Question 2

A team is using Unity Catalog (UC) for model governance. They have a model named `prod_catalog.ml_models.customer_churn`. They want to mark version 5 as the "live" version that applications should use, and version 4 as the "challenger" for A/B testing. What is the correct way to do this in UC?

- A. Transition v5 to the "Production" stage and v4 to the "Staging" stage.
- B. This is not possible; you can only have one "Production" version.
- C. Set the alias "live" to point to version 5, and the alias "challenger" to point to version 4.
- D. Edit the model's description to include `live: v5` and `challenger: v4`.

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: In Unity Catalog, stages (like "Staging" and "Production") are replaced by aliases. An alias is a mutable pointer to a specific model version. This is the standard way to manage environment-specific model versions in UC, allowing for flexible setups like A/B testing.

Question 3

A data scientist has logged a model using `mlflow.sklearn.log_model()`. They now want to load this model back into a notebook for batch prediction. The model is part of run ID `xyz123` and was saved with the artifact path `model`. Which MLflow command should they use?

- A. `model = mlflow.pyfunc.load_model(model_uri="runs:/xyz123/model")`
- B. `model = mlflow.sklearn.load_model(model_uri="models:/model/xyz123")`
- C. `model = mlflow.get_run("xyz123").load_artifact("model")`
- D. `model = mlflow.load_model(run_id="xyz123", path="model")`

✓ **Answer and Rationale** ›

Correct Answer: A

Rationale: The `mlflow.pyfunc.load_model()` command is the standard, flavor-agnostic way to load any MLflow model for inference. The `runs:/<run_id>/<artifact_path>` URI is the correct syntax to reference a model artifact that is part of a specific run (and not yet registered in the Model Registry).

Question 4

An ML engineer is writing a CI/CD script to automatically register a model in Unity Catalog. Which MLflow command correctly registers the model from a run (`<run_id>`) to the UC model registry?

- A. `mlflow.register_model("runs:/<run_id>/model", "prod_catalog.ml_models.my_model")`
- B. `mlflow.uc.register_model("runs:/<run_id>/model", "my_model")`
- C. `mlflow.create_model_version("my_model", run_id="<run_id>")`
- D. `mlflow.log_model(model, "prod_catalog.ml_models.my_model")`

✓ **Answer and Rationale** ›

Correct Answer: A

Rationale: The `mlflow.register_model()` function is used for this. It takes the `model_uri` (e.g., `runs:/<run_id>/model`) as the source and the three-level UC name (`catalog.schema.model`) as the destination name.

Question 5

A data scientist is building a training set using the Databricks Feature Store client. They have their raw data in a DataFrame `labels_df` with `user_id` and `timestamp` columns. They want to join features from a feature table `fs_table` . What is the correct syntax?

- A. `fs.create_training_set(df=labels_df, feature_lookups=[FeatureLookup(table_name='fs_table', lookup_key='user_id')], label='target')`
- B. `fs.join(labels_df, fs_table, on='user_id')`
- C. `fs.get_features(labels_df, 'fs_table')`
- D. `TrainingSet(labels_df, [FeatureLookup('fs_table')])`

✓ **Answer and Rationale** ›

Correct Answer: A

Rationale: The correct method is `fs.create_training_set()` . It requires the `labels_df` , a `label` column, and a list of `FeatureLookup` objects. The `FeatureLookup` object specifies the feature table to join and the key(s) to join on. This method also correctly handles point-in-time joins if a timestamp is provided.

Question 6

A team is using Databricks AutoML for a classification problem. The "best" model generated is an XGBoost model, but the team is concerned about overfitting. How can they modify the AutoML run to generate less complex models?

- A. This is not possible; AutoML always picks the most complex model.
- B. In the AutoML UI, specify "Decision Tree" and "Logistic Regression" in the Frameworks setting.
- C. Manually edit the generated notebook's SHAP plots.
- D. Set the `timeout_minutes` to a smaller value.

✓ Answer and Rationale ›

Correct Answer: B

Rationale: The AutoML UI (and API) allows you to restrict the frameworks (algorithms) it will try. By default, it tries a mix (XGBoost, LightGBM, scikit-learn, etc.). To get simpler, more interpretable, or less complex models, the team can explicitly tell AutoML to only try frameworks like "Decision Tree" or "Logistic Regression".

Question 7

A team has two feature tables in Unity Catalog: `user_features` (keyed on `user_id`) and `product_features` (keyed on `product_id`). They want to create a training set for a recommendation model that has both `user_id` and `product_id` in its raw data. How should they configure the `fs.create_training_set()` call?

- A. It is not possible; `create_training_set` only supports one `FeatureLookup`.
- B. Use two separate `FeatureLookup` objects in the `feature_lookups` list, one for each table.
- C. They must first manually join `user_features` and `product_features` into a single table.
- D. Use a `FeatureLookup` with a composite key: `lookup_key=['user_id', 'product_id']`.

✓ Answer and Rationale ›

Correct Answer: B

Rationale: The `feature_lookups` parameter in `fs.create_training_set()` accepts a *list* of `FeatureLookup` objects. This is the correct way to join features from multiple different tables. One `FeatureLookup` would be `FeatureLookup(table_name='user_features', lookup_key='user_id')` and the second would be `FeatureLookup(table_name='product_features', lookup_key='product_id')`.

Question 8

A data scientist is training a PyTorch model in a Databricks notebook. They want to automatically log all metrics, parameters, and the final model with MLflow. What single line of code should they add before their training loop?

- A. `mlflow.autolog(framework="pytorch")`
- B. `mlflow.start_run(autolog=True)`
- C. `mlflow.pytorch.autolog()`
- D. `mlflow.enable_autologging(frameworks="all")`

✓ Answer and Rationale ›

Correct Answer: C

Rationale: MLflow autologging is flavor-specific. Just as `mlflow.sklearn.autolog()` is used for scikit-learn, `mlflow.pytorch.autolog()` is the correct command to enable automatic logging of parameters, metrics (from the training loop), and model artifacts for PyTorch models.

Question 9

A data scientist has finished their analysis in a notebook within Databricks Repos. They are on a feature branch named `feature-xyz`. They open the Git dialog in the notebook UI. What is the correct sequence of operations to save their work to the remote Git repository?

- A. Push -> Commit
- B. Create Branch -> Commit -> Push
- C. Add a commit message -> Commit -> Push
- D. Pull -> Commit -> Push

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: The Databricks Repos UI simplifies the Git workflow. When you have changes, you open the Git dialog, type a commit message, click the "Commit" button (which stages and commits the changes), and then click the "Push" button to send those commits to the remote repository.

Question 10

An ML engineer has reviewed a notebook generated by AutoML. They like the preprocessing steps but want to use a different model (e.g., a custom `statsmodels` algorithm) instead of the one AutoML chose. What is the recommended workflow?

- A. This is not possible; the notebooks are read-only.
- B. Clone the generated notebook, delete the model training cell, and add new cells with the custom model code.
- C. Re-run AutoML and specify `statsmodels` in the Frameworks setting.
- D. Export the notebook and run it outside of Databricks.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: The "glass box" notebooks generated by AutoML are fully editable. The intended workflow for customization is to clone the notebook, which gives you a copy you can edit. The engineer can then modify the model training section while preserving all the preprocessing and data-loading code that AutoML generated.

Question 11

A team is deploying a model to a Databricks Model Serving endpoint. The model requires several complex, custom Python libraries that are not in the standard conda environment. How can the team ensure these libraries are available in the serving environment?

- A. Add the libraries to the "Libraries" tab of the cluster the notebook was trained on.
- B. Log a `requirements.txt` file as an artifact with the model in MLflow.
- C. Manually SSH into the serving endpoint and `pip install` the libraries.
- D. This is not possible; endpoints only support standard libraries.

✓ Answer and Rationale ›

Correct Answer: B

Rationale: When you log a model with MLflow (e.g., `mlflow.pyfunc.log_model`), you can specify a `pip_requirements` or `conda_env` argument. This saves the dependencies with the model. When Databricks Model Serving deploys the model, it reads this file and installs the specified libraries in the serving container.

Question 12

What is the primary difference between the MLflow Model Registry in the workspace and the Model Registry in Unity Catalog?

- A. The workspace registry is for scikit-learn models, while the UC registry is for Spark ML models.
- B. The workspace registry uses stages (Staging, Production), while the UC registry uses aliases (live, challenger).
- C. The workspace registry is open-source, while the UC registry is proprietary.
- D. The workspace registry cannot be accessed via the UI.

✓ Answer and Rationale ›

Correct Answer: B

Rationale: A key functional difference is the governance model. The classic workspace registry uses "stages" (Staging, Production, Archived) to manage model promotion. The modern Unity Catalog registry deprecates stages in favor of "aliases," which are mutable pointers (e.g., `live`) that can be pointed to any version.

Question 13

An MLOps engineer is using a Databricks Job to orchestrate a nightly retraining pipeline. The first task (data ingestion) fails. What is the default behavior of the second task (model training) which depends on the first?

- A. The second task will run, but it will use cached data from the previous day.
- B. The second task will not be run, and the job will be marked as "Failed."
- C. The second task will run after a 5-minute delay to allow the first task to be fixed.
- D. The job will send an email, and the second task will wait until a user manually approves it.

✓ Answer and Rationale ›

Correct Answer: B

Rationale: In Databricks Jobs (Workflows), tasks have dependencies. By default, if a task fails, any downstream tasks that depend on it will be skipped, and the overall job run will be marked as "Failed." This prevents the training task from running without its required input data.

Domain: ML Workflows (13 Questions)

Question 14

A data scientist is preparing features for a linear regression model. The `user_age` feature has a range of 18-90, while the `monthly_spend` feature has a range of $5-$10,000. Why is it important to scale these features?

- A. It is not important; linear regression is immune to feature scale.
- B. To prevent the `monthly_spend` feature from dominating the `user_age` feature in the model's coefficient calculation.
- C. To convert the features into a single vector using `VectorAssembler`.
- D. To impute missing values using the mean.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: Models that use gradient descent (like Linear/Logistic Regression) or distance calculations (like K-Means/SVM) are sensitive to feature scales. Without scaling, a feature with a large range (like `monthly_spend`) will have a disproportionately large impact on the model's cost function and coefficients compared to a feature with a small range.

Question 15

A model is being trained to predict customer churn (a binary classification task). The business priority is to have a good balance between correctly identifying churners (Recall) and not incorrectly flagging non-churners (Precision). Which metric best represents this balance?

- A. F1-Score
- B. Area Under ROC Curve (AUC-ROC)
- C. Accuracy
- D. Root Mean Squared Error (RMSE)

✓ **Answer and Rationale** ›

Correct Answer: A

Rationale: The F1-Score is the harmonic mean of Precision and Recall. It is a single metric that seeks a balance between the two, making it ideal for scenarios where both false positives and false negatives are costly.

Question 16

A team is training a model on a highly imbalanced dataset (98% Class A, 2% Class B). They find their accuracy is 98%, but their model is not useful. Which metric would have immediately revealed the model's poor performance on the minority class (Class B)?

- A. Area Under the Precision-Recall Curve (AUC-PR)
- B. R-squared

- C. Mean Absolute Error (MAE)
- D. Accuracy

✓ Answer and Rationale ›

Correct Answer: A

Rationale: In an imbalanced dataset, accuracy is misleading. AUC-ROC can also be overly optimistic. The Precision-Recall Curve (and its area, AUC-PR) is the most informative metric for imbalanced classification because it focuses on the performance of the rare, positive class (Class B) and is not inflated by the large number of true negatives (Class A).

Question 17

A data scientist is tuning a model using Hyperopt. What is the purpose of the `fmin()` function in the Hyperopt library?

- A. It is the function that defines the model to be trained.
- B. It is the main function used to run the hyperparameter search, which minimizes an objective function.
- C. It defines the search space (e.g., `hp.choice`, `hp.uniform`).
- D. It is a utility for parallelizing the search using `SparkTrials`.

✓ Answer and Rationale ›

Correct Answer: B

Rationale: The `fmin()` function is the "find minimum" function and is the core of Hyperopt. It orchestrates the tuning process by: 1) taking the objective function to minimize, 2) the search space, 3) the search algorithm (e.g., TPE), and 4) the Trials object. Its goal is to find the set of hyperparameters that minimizes the objective (e.g., loss, or `1 - accuracy`).

Question 18

In a CI/CD pipeline for MLOps, what is the primary purpose of the "Continuous Integration" (CI) stage?

- A. To automatically deploy the model to a production REST endpoint.
- B. To automatically retrain the model on new data.
- C. To automatically run tests (e.g., unit tests, data validation, model validation) every time new code is committed.
- D. To manually review the performance of the model in a dashboard.

✓ Answer and Rationale ›

Correct Answer: C

Rationale: Continuous Integration (CI) is the practice of frequently merging code changes into a central repository, after which automated builds and tests are run. In MLOps, this includes running unit tests for feature engineering code, data quality tests, and tests to validate the model training process.

Question 19

A data scientist is evaluating a regression model for predicting house prices. They want a metric that is in the same units as the label (e.g., in dollars) and penalizes large errors more heavily. Which metric should they use?

- A. R-squared (R^2)
- B. Mean Absolute Error (MAE)
- C. Root Mean Squared Error (RMSE)
- D. F1-Score

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: Both MAE and RMSE are in the same units as the label (dollars). However, RMSE is calculated by squaring the errors before averaging and then taking the square root. This squaring process gives a much higher weight to large errors (outliers). Therefore, RMSE is the metric that penalizes large errors more heavily.

Question 20

A data scientist has a categorical feature `city` with thousands of unique values (high cardinality). They are concerned that one-hot encoding will create too many dimensions. What is an alternative encoding technique that can represent these categories in a dense, lower-dimensional vector?

- A. String Indexing
- B. Target Encoding
- C. Min-Max Scaling
- D. Vector Assembler

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: Target Encoding (or Mean Encoding) is a technique where each category is replaced with the mean of the target variable for that category. This (or variations like it) captures information about the category's relationship with the target in a single numerical feature, avoiding the high dimensionality of one-hot encoding.

Question 21

A data engineering team is using Delta Live Tables (DLT) to build the data pipeline for an ML model. How can they automatically stop the pipeline from processing data that does not meet certain quality standards (e.g., `user_id IS NOT NULL`)?

- A. By setting a `data_quality_check` parameter in the cluster configuration.
- B. By using a "data quality expectation" in the DLT pipeline definition.
- C. By writing a separate notebook to query the table and check for nulls.
- D. By configuring an alert in Databricks SQL.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: A key feature of DLT is the ability to define data quality constraints, or "expectations." An expectation (e.g., `@dlt.expect_or_fail("user_id_not_null", "user_id IS NOT NULL")`) tells the DLT pipeline to fail (or drop) records that do not meet this constraint, thus automatically enforcing data quality.

Question 22

An MLOps team is adopting Databricks Asset Bundles (DABs) for CI/CD. What is the name of the YAML file that defines all the project's assets, such as notebooks, cluster configurations, and job tasks?

- A. `databricks.yaml`
- B. `bundle.yml`
- C. `mlproject`
- D. `config.yaml`

✓ **Answer andR ationale** ›

Correct Answer: A

Rationale: The `databricks.yaml` file is the core configuration file for a Databricks Asset Bundle. It is used to declare all the resources (jobs, notebooks, pipelines, cluster specs, etc.) that make up the project, allowing them to be deployed as a single unit ("bundle").

Question 23

A production model is being monitored. The team observes that the relationship between the input features and the target variable has changed. For example, a feature that was previously a strong positive predictor is now a negative predictor. What is this phenomenon called?

- A. Data Drift
- B. Concept Drift
- C. Upstream Data Change
- D. Model Staleness

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: Concept Drift is defined as a change in the statistical *relationship* between the input features and the target label. This is distinct from Data Drift, which is a change in the distribution of the input features themselves.

Question 24

A data scientist wants to visualize the distribution of a numerical column in a very large Spark DataFrame. Running `df.toPandas().hist()` causes an Out-Of-Memory (OOM) error. What is the most appropriate way to generate this plot in a Databricks notebook?

- A. Use the built-in `display(df)` command and select the histogram plot type.
- B. Increase the driver node memory and try again.

- C. Use `df.sample(0.01).toPandas().hist()`.
- D. Use the `dbutils.display.histogram()` command.

✓ **Answer and Rationale** ›

Correct Answer: A

Rationale: The `display(df)` command in a Databricks notebook is designed for large-scale data visualization. When you call it on a Spark DataFrame, it does not pull all the data to the driver. Instead, it computes the necessary statistics/samples in a distributed way and renders the plot. The user can then select the histogram plot type from the UI.

Question 25

When setting up Databricks Lakehouse Monitoring, what is the purpose of the "baseline table"?

- A. It is the table that stores the monitoring metrics (e.g., drift, performance).
- B. It is the inference table that logs all new predictions.
- C. It is a table of "ground truth" data (e.g., the training or validation set) that the new inference data is compared against.
- D. It is a table that defines the thresholds for alerting.

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: Lakehouse Monitoring works by comparing two tables: the "inference table" (live data) and the "baseline table." The baseline table (often the training or validation data) acts as the "ground truth" distribution. The monitor compares the statistics of the inference table against the baseline to detect data and concept drift.

Question 26

A model predicts product categories. After deployment, a new product category, "Electronics," is introduced. The model, which was not trained on this category, starts receiving it as an input. What specific type of drift is this?

- A. Concept Drift
- B. Label Drift
- C. Data Drift (specifically, new category drift)
- D. Prediction Drift

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: This is a form of Data Drift. The statistical distribution of the `product_category` input feature has changed because a new, unseen value ("Electronics") has appeared. This is a common sub-type of data drift that monitoring systems are designed to catch.

Domain: Spark ML (15 Questions)

Question 27

A data scientist is building a Spark ML Pipeline. They create a `StringIndexer` and a `OneHotEncoder`. What is the base class for both of these objects?

- A. `Transformer`
- B. `Estimator`
- C. `PipelineStage`
- D. `Model`

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: Both `StringIndexer` and `OneHotEncoder` are **Estimators**. They must *learn* from the data (e.g., learn the string-to-index mapping or the number of categories) by calling their `.fit()` method. This `fit()` call returns a **Transformer** (e.g., `StringIndexerModel`, `OneHotEncoderModel`) which can then `.transform()` the data.

Question 28

A data scientist has a `Pipeline` object (not a `PipelineModel`). What will happen if they call the `.transform()` method on this object?

- A. It will transform the data by applying all stages sequentially.
- B. It will return an error, as a `Pipeline` is an Estimator and does not have a `.transform()` method.
- C. It will automatically call `.fit()` first and then `.transform()`.
- D. It will return an empty DataFrame.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: This is a key API distinction. A `Pipeline` is an **Estimator**. Estimators have a `.fit()` method, not a `.transform()` method. You must first call `.fit()` on the `Pipeline` to produce a `PipelineModel`, which is a **Transformer**. You can then call `.transform()` on the `PipelineModel`.

Question 29

A data scientist has a column `age` and wants to create a new column `age_group` that sorts the ages into discrete bins (e.g., 0-18, 19-35, 36-50, 50+). Which Spark ML Transformer or Estimator should they use?

- A. `VectorAssembler`
- B. `StandardScaler`
- C. `Bucketizer`
- D. `StringIndexer`

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: `Bucketizer` is a **Transformer** that converts a continuous numerical feature into discrete bins (buckets). The user defines the split points (e.g., `[0, 18, 35, 50, float('inf')]`), and `Bucketizer` transforms the `age` column into a new column with the corresponding bin index (e.g., 0.0, 1.0, 2.0).

Question 30

A data scientist has a `VectorAssembler` output column named `features` which contains categorical features (e.g., `zip_code_index`, `product_category_index`). They are building a `DecisionTreeClassifier`. Which Spark ML Estimator can automatically identify these features as categorical and improve the tree's performance?

- A. `VectorIndexer`
- B. `OneHotEncoder`
- C. `StandardScaler`
- D. `RFormula`

✓ **Answer and Rationale** ›

Correct Answer: A

Rationale: `VectorIndexer` is designed for this purpose. It "indexes" a feature vector, automatically identifying categorical features (based on the number of distinct values) and storing this metadata. Tree-based models in Spark ML can then use this metadata to correctly treat those features as categorical, rather than continuous.

Question 31

What are the primary types of Pandas UDFs available in PySpark?

- A. Series to Series, and Series to Scalar
- B. `map`, `flatMap`, and `filter`
- C. Series to Series, Iterator of Series to Iterator of Series, and Iterator of multiple Series to Iterator of Series
- D. `apply`, `transform`, and `aggregate`

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: Pandas UDFs (defined with `@pandas_udf`) operate on `pandas.Series` or `pandas.DataFrame` objects. The main types are: 1) Series to Series (e.g., `df.withColumn("col", udf("col"))`), 2) Iterator of Series to Iterator of Series (for operations with state), and 3) Iterator of multiple Series to Iterator of Series (e.g., `df.mapInPandas()`).

Question 32

A data scientist has a large Spark DataFrame `sdf`. They want to use the pandas API on Spark to perform a `groupby` operation. What is the correct way to do this?

- A. `pdf = sdf.toPandas()` and then `pdf.groupby('col').mean()`
- B. `import pyspark.pandas as ps` and then `ps.DataFrame(sdf).groupby('col').mean()`
- C. `import pyspark.pandas as ps` and then `pdf = sdf.to_pandas_on_spark()`
- D. `import pyspark.pandas as ps` and then `pdf = ps.from_spark(sdf)`

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: The Pandas API on Spark (imported as `ps`) is integrated with the Spark DataFrame API. The correct way to convert a Spark DataFrame (`sdf`) to a Pandas-on-Spark DataFrame (`pdf`) is to call the `.to_pandas_on_spark()` method. (Note: `ps.from_spark(sdf)` is also a valid method, but `.to_pandas_on_spark()` is the more common and idiomatic way). `toPandas()` (Option A) will OOM.

Question 33

Why is it critical to fit a `StandardScaler` as part of a `Pipeline` rather than scaling the training data before creating the `Pipeline`?

- A. To prevent data leakage from the test set into the scaler's statistics.
- B. To ensure the scaler can be saved as part of the `PipelineModel`.
- C. To allow the scaler to be fit on the training data and then applied (with the same mean/stddev) to the test data.
- D. All of the above.

✓ **Answer and Rationale** ›

Correct Answer: D

Rationale: All three points are correct and crucial. 1) When used with `CrossValidator`, the `Pipeline` ensures the scaler is fit *only* on the training fold, preventing the test fold's stats from leaking. 2) Saving the `PipelineModel` saves the fitted `StandardScalerModel`. 3) This ensures that the exact same transformation (using the training data's mean/std) is applied to new data at inference time.

Question 34

A data scientist needs to perform a stratified split on their DataFrame `df` based on the `label` column to ensure both training and test sets have the same class distribution. Which method should they use?

- A. `df.randomSplit([0.8, 0.2], seed=42)`
- B. `df.sampleBy('label', fractions={0: 0.8, 1: 0.8}, seed=42)`
- C. `df.stratifiedSplit('label', test_size=0.2)`
- D. This is not directly supported in Spark's DataFrame API for train/test splits.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: Spark ML's `randomSplit` does not support stratification. The correct way to perform a stratified sample is to use the `DataFrame.sampleBy()` method. You specify the column to stratify by (`label`) and a dictionary of the fractions to take for each class. (Note: This creates one sample; you would need a second step, like `df.exceptAll(train_df)` , to get the test set).

Question 35

A data scientist has a feature vector with 500 features and is concerned about multicollinearity and overfitting. They want to reduce the dimensionality of their feature space. Which Spark ML Estimator should they use?

- A. `VectorAssembler`
- B. `PCA` (Principal Component Analysis)
- C. `Bucketizer`
- D. `StandardScaler`

✓ **Answer and Rationale** >

Correct Answer: B

Rationale: `PCA` is a dimensionality reduction technique. It (an **Estimator**) is fit on the feature vectors to find the principal components (linear combinations of the original features) that explain the most variance. It produces a `PCAModel` (a **Transformer**) that can transform the 500-dimension vectors into a much smaller vector (e.g., 50 dimensions).

Question 36

A `StringIndexer` is being fit to a `zip_code` column. During inference, a new `zip_code` appears that was not in the training data. What is the default behavior of the `StringIndexerModel` ?

- A. It will assign the new `zip_code` to the most frequent category.
- B. It will assign the new `zip_code` to a special "unseen" index.
- C. It will throw an error.
- D. It will drop the row containing the new `zip_code` .

✓ **Answer and Rationale** >

Correct Answer: C

Rationale: By default, the `handleInvalid` parameter of `StringIndexer` is set to `error` . This means that if it encounters a new, unseen string, it will throw an exception. To handle this, you must set `handleInvalid` to `skip` (drop the row) or `keep` (assign to a special index for unseen values).

Question 37

A data scientist has trained a binary classification model and wants to get the Area Under the ROC Curve (AUC-ROC) metric. Which Spark ML class should they use?

- A. `RegressionEvaluator(metricName="auc")`

- B. `BinaryClassificationEvaluator(metricName="areaUnderROC")`
- C. `MulticlassClassificationEvaluator(metricName="auc")`
- D. `ClusteringEvaluator()`

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: The `BinaryClassificationEvaluator` is the correct class. It is used to evaluate binary classification models, and its default metric is `areaUnderROC`. You can also explicitly set it with `metricName="areaUnderROC"` or `metricName="areaUnderPR"` (for the PR curve).

Question 38

When using `CrossValidator` in Spark ML, what is the purpose of the `EstimatorParamMaps`?

- A. It is a list of all the models trained during the cross-validation.
- B. It is the "grid" of hyperparameters to search.
- C. It is the Estimator (e.g., `LogisticRegression`) that will be tuned.
- D. It is the map of metrics (e.g., `auc`, `f1`) to be calculated.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: The `EstimatorParamMaps` (often generated by a `ParamGridBuilder`) is the set of hyperparameter combinations that `CrossValidator` will test. For example, it's a list where each element is a map like `{lr.regParam: 0.1, lr.elasticNetParam: 0.5}`, `{lr.regParam: 0.01, ...}`.

Question 39

A data scientist is creating features for a time-series model. They need to create a "3-day moving average" for a `sales` column. Which Spark SQL window function is most appropriate for this?

- A. `PARTITION BY`
- B. `LAG(sales, 3)`
- C. `AVG(sales) OVER (ORDER BY timestamp ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)`
- D. `NTILE(3) OVER (ORDER BY sales)`

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: This is the correct syntax for a window function. `ORDER BY timestamp` ensures the data is sorted chronologically. `ROWS BETWEEN 2 PRECEDING AND CURRENT ROW` defines the "3-day" window (the current day + the 2 previous days). `AVG(sales) OVER (...)` then calculates the average over that sliding window.

Question 40

A data scientist wants to use Hyperopt to tune a Spark ML `GBTClassifier` (a distributed model). Which `Trials` object should they pass to the `fmin()` function to ensure the tuning process is efficient?

- A. `SparkTrials(parallelism=sc.defaultParallelism)`
- B. `Trials()` (the default, single-node Trials object)
- C. `MongoTrials()`
- D. They should not use Hyperopt; `CrossValidator` is the only option.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: This is a common point of confusion. `SparkTrials` is used to parallelize the tuning of *single-node models* (like scikit-learn). When the model itself is already distributed (like a Spark ML model), you should use the standard, single-node `Trials()` object. The driver will manage the Hyperopt search, and each trial will run a full, distributed Spark ML job.

Question 41

What is a key difference between `RandomForestClassifier` and `GBTClassifier` (Gradient-Boosted Trees) in Spark ML?

- A. `RandomForest` is an Estimator, while `GBTs` are Transformers.
- B. `RandomForest` trains many trees in parallel, while `GBTs` train trees sequentially, with each tree correcting the errors of the previous one.
- C. `RandomForest` can only be used for classification, while `GBTs` can be used for regression.
- D. `RandomForest` does not require a `VectorAssembler`.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: This is the fundamental algorithmic difference. Random Forest is a "bagging" method where many independent trees are trained in parallel (on different data samples) and vote on the result. GBTs are a "boosting" method where trees are trained sequentially, and each new tree is trained to correct the residual errors of the previous tree.

Domain: Scaling ML Models (4 Questions)

Question 42

A data scientist has a trained scikit-learn model logged in MLflow. They need to apply this model to a 10 TB Spark DataFrame for batch inference. What is the most scalable way to do this?

- A. Call `df.toPandas()` and then apply the model in a loop.
- B. Use `mlflow.pyfunc.spark_udf()` to wrap the model as a Spark UDF and apply it to the DataFrame.
- C. Deploy the model to a Model Serving endpoint and send each row as a REST request.
- D. Manually re-implement the scikit-learn model's logic using Spark SQL.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: `mlflow.pyfunc.spark_udf()` is designed for this exact use case. It takes a model URI, wraps the single-node scikit-learn model in a Pandas UDF, and broadcasts it to the Spark executors. This allows you to apply the model in a distributed fashion to a Spark DataFrame, scaling inference to terabytes of data.

Question 43

A financial services company needs to check every website login attempt for fraud, with a latency requirement of < 50ms. Which deployment pattern is required?

- A. Batch Inference, run as a Databricks Job every 1 minute.
- B. Real-time Inference (Online Serving), using a REST API endpoint.
- C. A Delta Live Tables pipeline with a streaming source.
- D. An MLflow `mlproject` file run from the web server.

✓ **Answer and RWarning** ›

Correct Answer: B

Rationale: The key requirements are "every... attempt" and "latency < 50ms." This is a classic real-time (or online) inference use case. It requires a model to be pre-loaded and waiting for individual requests via a low-latency REST API, such as that provided by Databricks Model Serving.

Question 44

A Databricks Model Serving endpoint is running a large (e.g., 10GB) deep learning model. It is experiencing high latency. The endpoint is configured to "scale to zero." What is the most likely cause of intermittent high latency, especially for the first request after a period of no traffic?

- A. The model is overfitting.
- B. This is "cold start" latency, as the endpoint is scaling up from zero and loading the large model into memory.
- C. The REST API request is formatted incorrectly.
- D. Data drift has occurred, and the model is taking longer to process the new data.

✓ **Answer andR ationale** ›

Correct Answer: B

Rationale: "Scale to zero" is a cost-saving feature, but its trade-off is "cold start" latency. When the endpoint scales to zero, all compute is shut down. The first request that arrives must wait for the system to provision a container, download the 10GB model, and load it into memory before it can be served. This one-time delay is the cold start.

Question 45

A team is running a large batch inference job that joins a 1TB `inference_data` table with a 50GB `feature_store` table. The job is very slow. What Delta Lake optimization would likely provide the most significant speedup for this join operation?

- A. Running `VACUUM` on the tables to remove old files.
- B. Enabling `autoOptimize` on the tables.
- C. Using Z-Ordering on the `inference_data` and `feature_store` tables by their join keys (e.g., `user_id`).
- D. Converting the tables from Parquet to Delta.

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: Z-Ordering is a data layout optimization that co-locates related data in the same files. By Z-Ordering both tables on their join key (`user_id`), you enable "data skipping." This means Spark's query optimizer knows that it only needs to read a small subset of files from each table to perform the join, drastically reducing I/O and shuffle.