ML Associate - Mock Test 1

Domain: Databricks Machine Learning (13 Questions)

Question 1

A data science team is using Databricks AutoML to quickly generate baseline models. A senior ML engineer wants to review the exact preprocessing steps and feature engineering logic that AutoML applied. How can they achieve this?

- A. Access the "Model Explainability" tab in the AutoML experiment, which lists the source code.
- B. This is not possible; AutoML is a "black box" solution to protect proprietary algorithms.
- C. Download the generated Python notebook for a specific trial run from the AutoML experiment page.
- D. Query the MLflow Tracking server using mlflow.search_runs() to retrieve the source code as an artifact.

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: Databricks AutoML follows a "glass box" approach. For every trial run, it generates an editable Python notebook that contains the full source code for data preprocessing, model training, and evaluation. This allows for full transparency and customization.

Question 2

An ML Engineer has a new model version that has passed all automated tests in a CI/CD pipeline. They want to promote this model to serve 10% of production traffic for A/B testing before a full rollout. Which MLflow Model Registry stage should they transition the model to?

- A. None
- B. Staging
- C. Archived
- D. Production

✓ **Answer and Rationale** ›

Correct Answer: D

Rationale: The Production stage is used for all models serving live traffic. Databricks Model Serving allows you to split traffic between different versions within the Production stage (e.g., 90% to v1, 10% to v2). The Staging stage is for final pre-production validation, not for serving live traffic.

Question 3

A data scientist has run hundreds of experiments for a single project, all logged with MLflow. They now want to find the specific run that achieved the highest `val_recall` metric, but only for runs where the `learning_rate` parameter was less than `0.01`. Which tool should they use?

- A. `mlflow.search_runs()` with a `filter_string` and `order_by`.
- B. Databricks AutoML, as it automatically ranks models.
- C. The "Compare Runs" button in the MLflow UI, as it allows custom SQL queries.
- D. The MLflow Model Registry UI, which has built-in filters for metrics.

✓ **Answer and Rationale** >

Correct Answer: A

Rationale: The `mlflow.search_runs()` API command is designed for this exact purpose. It allows you to programmatically query the MLflow Tracking server with a SQL-like `filter_string` (e.g., `"params.learning_rate < 0.01"`) and sort the results using `order_by` (e.g., `["metrics.val_recall DESC"]`).

Question 4

A team is migrating their ML platform to Unity Catalog. They want to register a new version of their customer churn model. What is the correct way to name and register this model using the `mlflow.register_model()` command?

- A. `mlflow.register_model(model_uri, "customer_churn_prod")`
- B. `mlflow.register_model(model_uri, "prod.models.customer_churn")`
- C. `mlflow.register_model(model_uri, "prod_catalog.ml_models.customer_churn")`
- D. `mlflow.register_model(model_uri, "models:/customer_churn/production")`

✓ **Answer and Rationale** >

Correct Answer: C

Rationale: Unity Catalog uses a three-level namespace for all assets: `catalog_name.schema_name.object_name`. When registering models in UC, you must provide this full three-part name. The other options represent workspace-scoped model names.

Question 5

An ML engineer is cleaning up their project and needs to delete an old Feature Store table named `prod_catalog.features.user_profile_legacy`. They try running `fs.delete_table("prod_catalog.features.user_profile_legacy")` but receive an error. What is the correct way to delete this feature table?

- A. Use the Databricks CLI command `databricks fs -rm "prod_catalog.features.user_profile_legacy"`.
- B. Use the Catalog Explorer UI to manually delete the table.
- C. Run a SQL command `DROP TABLE prod_catalog.features.user_profile_legacy`.
- D. They must first archive the table in the Feature Store UI before deleting it with the API.

✓ **Answer and Rationale** >

Correct Answer: B

Rationale: This is a specific "gotcha" from your study materials. Feature tables created in Unity Catalog are managed as Delta tables within UC. The Feature Store Python API (fs) cannot delete them. They must be deleted from the Catalog Explorer (or via a `DROP TABLE` SQL command), just like any other table in UC.

Question 6

A company wants to automate its financial forecasting. The data science team has a large dataset of historical daily sales figures. Which Databricks ML capability should they use to automatically preprocess the data, select the best model, and tune its hyperparameters for this specific problem?

- A. Databricks Jobs
- B. Databricks AutoML, selecting the "Forecasting" problem type.
- C. MLflow Model Registry, as it supports time-series models.
- D. Databricks Feature Store, using point-in-time lookups.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: Databricks AutoML is designed to automate the end-to-end process of model selection and tuning. Crucially, it has a specific problem type for "Forecasting" that automatically handles time-series-specific preprocessing and modeling.

Question 7

A team is creating a new Feature Store table in Unity Catalog to store user transaction data. They want to be able to look up features for a specific user at a specific point in time. What is a required step when creating this table?

- A. The table must be stored in the default schema.
- B. The table must have a primary key defined.
- C. The table must be created using the Feature Store UI, not the API.
- D. The table must be partitioned by date.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: To be used as a Feature Store table (especially for point-in-time lookups), Unity Catalog requires the table to have a primary key. This key is used to identify the entity (e.g., `user_id`) for feature lookups.

Question 8

A data scientist is training a scikit-learn model in a Databricks notebook. They want to ensure that all model parameters, metrics, and the model artifact are automatically logged to MLflow without writing explicit `mlflow.log_param()` or `mlflow.log_metric()` commands. What should they do?

- A. Use `mlflow.start_run(autolog=True)`.
- B. This is the default behavior of all Databricks ML runtimes.

- C. Call `mlflow.sklearn.autolog()`.
- D. Wrap the training code in a `with mlflow.autolog():` context manager.

✓ Answer and Rationale >

Correct Answer: C

Rationale: MLflow's autologging is flavor-specific. To automatically log metrics, parameters, and artifacts for scikit-learn models, you must call `mlflow.sklearn.autolog()` before your training code (e.g., `.fit()`) is executed.

Question 9

A data scientist has cloned a project from a remote Git repository into Databricks Repos. They have made several changes to a notebook and now want to commit and push their changes back to their feature branch. Where in the Databricks UI can they do this?

- A. From the Databricks Jobs UI, by creating a new "Git Push" task.
- B. From the MLflow Model Registry UI, by linking the repo to a model version.
- C. From the Cluster configuration page, under the "Libraries" tab.
- D. From the Git menu directly inside the notebook editor.

✓ Answer and Rationale >

Correct Answer: D

Rationale: Databricks Repos provides a Git-integrated UI. Directly within the notebook editor, there is a button (usually showing the current branch name) that opens a Git dialog. From there, you can add a commit message, commit the changes, and push them to the remote repository.

Question 10

A business stakeholder is skeptical about a new model that AutoML produced, fearing it's a "black box." How can the data science team best explain which features are most important for the model's predictions?

- A. Show them the `databricks.yaml` file, which defines the model's inputs.
- B. Refer them to the SHAP plots in the "Explainability" tab of the generated AutoML notebook.
- C. Run `model.feature_importances_` on the model, as SHAP is not supported.
- D. Explain that AutoML models are too complex to be interpreted.

✓ Answer and Rationale >

Correct Answer: B

Rationale: A key feature of the AutoML "glass box" approach is the generated notebook. This notebook includes an "Explainability" section with SHAP (SHapley Additive exPlanations) plots, which are a standard and powerful way to visualize global and local feature importance.

Question 11

A data scientist is building a model to predict if a user will click an ad. They have a feature table of user ad-click history and a separate table of ad impressions. To prevent data leakage, they must ensure that features (like `clicks_in_last_30_days`) are only calculated based on data available before the ad was shown. What Databricks capability is designed for this?

- A. Databricks AutoML, which automatically detects and prevents leakage.
- B. MLflow, by logging a timestamp for each parameter.
- C. Databricks Feature Store's point-in-time lookup capabilities.
- D. Databricks Model Serving, which validates timestamps at inference time.

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: This is the primary purpose of the Feature Store's point-in-time correctness. When you create a training set using `fs.create_training_set()`, you provide the impression data with timestamps. The Feature Store performs a "point-in-time join" that correctly looks up feature values as they were at the time of the impression, preventing "future" data from leaking into the training set.

Question 12

A company has multiple data science teams. They are struggling with version control for their models, and it is difficult to know which model version is currently approved for production use. Which component of Databricks ML is designed to solve this problem?

- A. Databricks Repos
- B. MLflow Model Registry
- C. Databricks Jobs
- D. Databricks SQL

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: The MLflow Model Registry is the central, collaborative hub for managing the ML model lifecycle. It allows you to version models, add descriptions, and, most importantly, assign stages (like Staging and Production) to clearly track which model is approved for deployment.

Question 13

A team has a complete ML project that involves a data ingestion notebook, a feature engineering notebook, and a model training notebook. They need to run this entire sequence every night. What is the best way to orchestrate and schedule this multi-notebook workflow in Databricks?

- A. Create a Databricks Job with multiple tasks.
- B. Use `mlflow.run()` from a master notebook.
- C. Create a separate cluster for each notebook.
- D. Configure a real-time endpoint using Databricks Model Serving.

✓ Answer and Rationale >

Correct Answer: A

Rationale: Databricks Jobs (also known as Workflows) is the native Databricks orchestrator. It is designed to create and schedule multi-task workflows, where each task can be a different notebook, Python script, or JAR. This allows you to define dependencies (e.g., run training after feature engineering) and set a schedule (e.g., nightly).

Domain: ML Workflows (13 Questions)

Question 14

A data scientist is performing feature engineering on a Spark DataFrame. They need to impute missing values for a column `user_age`. The summary statistics for the column show a mean of 45, a median of 35, and a standard deviation of 20, with a few extreme outliers. Which imputation strategy is most robust?

- A. Impute with the mean (45), as it is the mathematical average.
- B. Impute with the median (35), as it is less sensitive to outliers.
- C. Impute with a constant value of 0.
- D. Drop all rows with missing values.

✓ Answer and Rationale >

Correct Answer: B

Rationale: The median is a robust statistic, meaning it is not heavily influenced by extreme outliers. The mean, in contrast, can be skewed by outliers, making it a poor representation of the central tendency in this case. Imputing with the median (35) is the most appropriate choice.

Question 15

A hospital is developing a classification model to detect a rare but serious disease. The business priority is to find as many true positive cases as possible, even if it means some healthy patients are incorrectly flagged (false positives). Which evaluation metric should be prioritized for tuning?

- A. Recall
- B. Precision
- C. Accuracy
- D. R-squared

✓ Answer and Rationale >

Correct Answer: A

Rationale: This scenario describes a high cost for False Negatives (missing a sick patient). Recall (defined as True Positives / (True Positives + False Negatives)) measures the model's ability to find all positive cases. Prioritizing recall means you are optimizing to minimize false negatives.

Question 16

A data scientist trains a classification model on a highly imbalanced dataset where 99% of the samples belong to the "Negative" class and 1% belong to the "Positive" class. The model achieves 99% accuracy on the test set. Why is accuracy a poor metric in this scenario?

- A. Accuracy is only intended for regression problems.
- B. The model is likely just predicting the "Negative" class for every sample.
- C. The accuracy is too high, which indicates overfitting.
- D. The test set is too small to calculate accuracy.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: This is the classic "accuracy paradox." In a 99% imbalanced dataset, a trivial model that always predicts the majority class ("Negative") will achieve 99% accuracy. This metric is misleading as it fails to capture the model's (in)ability to identify the rare positive class.

Question 17

A data scientist has a large hyperparameter search space with a mix of continuous and discrete values (e.g., `learning_rate` and `n_estimators`). They have a limited compute budget and time. Which tuning method is most efficient?

- A. Grid Search, because it guarantees finding the optimal combination.
- B. Manual Tuning, because the data scientist can use their intuition.
- C. Random Search, because it is the simplest to parallelize.
- D. Hyperopt, because it uses Bayesian optimization (TPE) to explore the space more intelligently.

✓ **Answer and Rationale** ›

Correct Answer: D

Rationale: Hyperopt (which uses Tree-structured Parzen Estimators, a Bayesian method) is more efficient than Grid Search or Random Search. It learns from past trials, focusing the search on more promising areas of the hyperparameter space. This "intelligent" search is ideal for limited compute budgets.

Question 18

In a mature MLOps lifecycle, what is the primary purpose of the Staging environment?

- A. To allow data scientists to explore new data and experiment with new algorithms.
- B. To serve live production traffic to end-users.
- C. To run automated integration tests and validate the model pipeline in a production-like environment before full deployment.
- D. To archive old model versions and data artifacts.

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: The Staging (or "pre-prod") environment is the quality-assurance step. It's meant to be a mirror of the production environment, where the final model candidate and pipeline are tested (e.g., integration tests, user acceptance tests) before being promoted to serve live traffic. Development/experimentation (A) happens in a Dev environment.

Question 19

A bank is building a model to approve loan applications. The business priority is to minimize the number of high-risk applicants who are incorrectly approved (false positives), as this costs the bank a significant amount of money. Which evaluation metric should be prioritized?

- A. Recall
- B. Precision
- C. Accuracy
- D. RMSE

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: This scenario describes a high cost for False Positives (approving a bad loan). Precision (defined as True Positives / (True Positives + False Positives)) measures the accuracy of the positive predictions. Prioritizing precision means you are optimizing to ensure that every "approved" prediction is correct, minimizing false positives.

Question 20

When imputing missing values, a data scientist is concerned that the fact that a value was missing is itself a useful signal for the model. What is the standard practice to preserve this information?

- A. Impute with a special value like -999.
- B. Add a binary indicator variable (column) that flags whether the original value was missing.
- C. Use a StringIndexer to convert the null values to a unique integer.
- D. This information cannot be preserved; it is lost during imputation.

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: Adding a binary indicator variable (e.g., `was_age_missing`) is the standard method. This allows the model to learn a weight for the "missingness" of the feature, separate from the imputed value (e.g., the median). This preserves the "missingness" as a potential predictive signal.

Question 21

A data engineering team needs to build a highly reliable data pipeline for an ML model. The pipeline must ingest streaming data, handle late-arriving data, and provide robust error handling and data quality checks automatically. Which tool is the best choice for this?

- A. Delta Live Tables (DLT)
- B. A standard Databricks Job running a Python notebook.
- C. Databricks Model Serving
- D. The Databricks Feature Store Client

✓ **Answer and Rationale** >

Correct Answer: A

Rationale: Delta Live Tables (DLT) is a declarative framework designed for this exact purpose. It simplifies the creation of reliable ETL pipelines, automatically handling orchestration, error handling, data quality monitoring (with "expectations"), and complex dependencies for both batch and streaming data.

Question 22

An MLOps engineer is trying to package all the artifacts for a project—including notebooks, environment definitions, and cluster configurations—so it can be automatically deployed to staging and production environments. What is the Databricks-native way to define this "Infrastructure as Code"?

- A. A Databricks Asset Bundle (DAB), configured with a `databricks.yaml` file.
- B. An MLflow `mlproject` file.
- C. A Databricks Repos `git.yaml` file.
- D. A Delta Lake manifest file.

✓ **Answer and Rationale** >

Correct Answer: A

Rationale: Databricks Asset Bundles (DABs) are the modern, standard way to define and manage ML projects as Infrastructure as Code on Databricks. The project is defined in a `databricks.yaml` file, which specifies notebooks, cluster definitions, job configurations, and other assets for easy CI/CD.

Question 23

A production model is being monitored using Databricks Lakehouse Monitoring. A drift threshold is breached, and an alert is triggered. What is the expected behavior of this alert?

- A. It will automatically trigger a new Databricks Job to retrain the model.
- B. It will automatically roll back the model to the previous version in the Model Registry.
- C. It will send a notification to a configured channel like Email, Slack, or Teams.
- D. It will automatically fix the model errors by correcting the input data.

✓ **Answer and Rationale** >

Correct Answer: C

Rationale: Lakehouse Monitoring is a tool for detection and alerting. It does not perform automated remediation (like retraining or rollback). When an alert condition (like a drift threshold) is met, it sends a notification to a pre-configured channel (Email, Slack, Teams, PagerDuty, etc.) so a human can investigate.

Question 24

A data scientist wants to get a quick statistical overview of their Spark DataFrame, `df`. They want to see the count, mean, stddev, min, and max for numerical columns, as well as approximate quartiles (25%, 50%, 75%). Which command should they use?

- A. `df.summary()`
- B. `df.describe()`
- C. `dbutils.data.summarize(df)`
- D. `df.toPandas().describe()`

✓ **Answer and RVideo** ›

Correct Answer: A

Rationale: The `df.describe()` command provides the count, mean, stddev, min, and max. The `df.summary()` command provides all of those plus approximate quartiles (and count of NaNs). Since the question explicitly asks for quartiles, `df.summary()` is the correct and more comprehensive command.

Question 25

What is the primary role of an Inference Table in the context of model monitoring?

- A. It is a feature table used by the model during online inference.
- B. It is a table in the Model Registry that lists all model versions.
- C. It is a Delta table that logs model inputs and their corresponding predictions for monitoring.
- D. It is a temporary table used by Spark ML for distributed training.

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: An Inference Table is a Delta table specifically created to log the inputs (features) and outputs (predictions) of a model serving in production. This table then becomes the source of truth for Databricks Lakehouse Monitoring to calculate drift and performance metrics.

Question 26

A production model's performance is degrading. The monitoring dashboard shows that the statistical distribution of the `age` input feature in the live data is significantly different from the distribution it was trained on. What is this phenomenon called?

- A. Concept Drift
- B. Data Drift

- C. Model Drift
- D. Label Drift

✓ Answer and Rationale >

Correct Answer: B

Rationale: Data Drift (or feature drift) is defined as a change in the statistical properties of the input features over time. Concept Drift is a change in the relationship between the features and the target label. Since only the input feature `age` changed, this is Data Drift.

Domain: Spark ML (15 Questions)

Question 27

In the Spark ML API, what is the key difference between an Estimator and a Transformer?

- A. An Estimator has a `.transform()` method, while a Transformer has a `.fit()` method.
- B. An Estimator learns from data and has a `.fit()` method, while a Transformer applies transformations and has a `.transform()` method.
- C. Estimators are used for feature engineering, while Transformers are used for modeling.
- D. Estimators can only be used in a Pipeline, while Transformers can be used standalone.

✓ Answer and Rationale >

Correct Answer: B

Rationale: This is the fundamental API distinction in Spark ML.

- **Estimator:** An algorithm that learns from data (e.g., `LinearRegression`, `StringIndexer`). You call `.fit(data)` on it to produce a Transformer.
- **Transformer:** An algorithm that transforms data (e.g., `LinearRegressionModel`, `StringIndexerModel`). You call `.transform(data)` on it to produce a new DataFrame.

Question 28

A data scientist has defined a Pipeline with three stages: a `StringIndexer`, a `OneHotEncoder`, and a `LinearRegression` model. They then call the `.fit()` method on this Pipeline object using their training data. What is the class of the object that is returned?

- A. A `PipelineModel`
- B. A `Transformer`
- C. A `LinearRegressionModel`
- D. A Spark DataFrame

✓ Answer and Rationale >

Correct Answer: A

Rationale: A `Pipeline` is an Estimator. When you call `.fit()` on any Estimator (including a `Pipeline`), it returns a corresponding Transformer. The Transformer returned by a `Pipeline` is a `PipelineModel`. This `PipelineModel` object contains all the fitted stages (e.g., the `StringIndexerModel`, `OneHotEncoderModel`, and `LinearRegressionModel`) and can be used to make predictions.

Question 29

A data scientist has a DataFrame with three numerical feature columns (`age`, `income`, `house_price`) and a label column (`churn`). Spark ML models require all features to be in a single vector column. Which Transformer should they use?

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

✓ **Answer and Rationale** ›

Correct Answer: A

Rationale: `VectorAssembler` is a Transformer whose sole purpose is to "assemble" multiple numerical columns into a single vector-valued column, which is the required input format for most Spark ML algorithms.

Question 30

A data scientist needs to encode a categorical string column `location` for a Spark ML model. What is the correct sequence of Estimators or Transformers to use?

- A. `OneHotEncoder` -> `StringIndexer`
- B. `StringIndexer` -> `OneHotEncoder`
- C. `VectorAssembler` -> `StringIndexer`
- D. `OneHotEncoder` -> `VectorAssembler`

✓ **Answer and Rationale** ›

Correct Answer: B

Rationale: This is the standard two-step pattern for categorical encoding in Spark ML.

1. `StringIndexer` (an Estimator) is fit to the data to convert string labels ("USA", "CAN", "MEX") into numerical indices (0.0, 1.0, 2.0).
2. `OneHotEncoder` (an Estimator) is then fit to the indexed column to convert those indices into sparse binary vectors (e.g., `(2, [0], [1.0])`).

Question 31

Pandas UDFs are known for being highly efficient for data transfer between the Spark JVM and Python processes. What technology enables this high-speed, in-memory data exchange?

- A. Apache Arrow
- B. Apache Parquet
- C. Delta Lake
- D. JSON

✓ Answer and Rationale >

Correct Answer: A

Rationale: Apache Arrow is an in-memory columnar data format. Spark uses Arrow to efficiently move data between the Java Virtual Machine (JVM) and the Python process with minimal (or zero) serialization/deserialization cost, which is what makes Pandas UDFs (and Pandas API on Spark) so performant.

Question 32

A data scientist is more comfortable with the pandas API than the Spark API. They need to perform data manipulation on a 5 TB dataset that cannot fit on a single machine. What Databricks capability allows them to use familiar pandas syntax on a distributed Spark cluster?

- A. Databricks AutoML
- B. Pandas UDFs
- C. Databricks Repos
- D. Pandas API on Spark

✓ Answer and Rationale >

Correct Answer: D

Rationale: The Pandas API on Spark (formerly known as Koalas) provides a pandas-like API that runs on top of Spark. It allows data scientists to use familiar pandas commands (e.g., `df['column']`, `df.groupby()`) on terabyte-scale data, which is executed by Spark in a distributed manner.

Question 33

Why is it a best practice to save the entire `PipelineModel` after training, rather than just the trained `LinearRegressionModel`?

- A. To ensure the exact same feature engineering and preprocessing steps are applied during inference.
- B. To save disk space, as the `PipelineModel` is more compressed.
- C. To allow for easier hyperparameter tuning at a later time.
- D. To log the model to the MLflow Model Registry, which only accepts `PipelineModel` objects.

✓ Answer and Rationale >

Correct Answer: A

Rationale: The `PipelineModel` contains all the fitted stages, including the `StringIndexerModel`, `OneHotEncoderModel`, `VectorAssembler`, etc. Saving the entire pipeline ensures that when you load it for inference, the new raw data goes through the exact same transformations (e.g., the same string-to-index mappings) that the training data did. This prevents training-serving skew.

Question 34

What is the correct Spark syntax to split a DataFrame `df` into a training set (80%) and a test set (20%) in a reproducible way?

- A. `train, test = df.randomSplit([0.8, 0.2], seed=42)`
- B. `train, test = df.split(ratio=0.8, seed=42)`
- C. `train = df.sample(0.8, seed=42)` and `test = df.sample(0.2, seed=42)`
- D. `train, test = train_test_split(df, test_size=0.2, random_state=42)`

✓ **Answer and Rationale** ›

Correct Answer: A

Rationale: The correct method on a Spark DataFrame is `.randomSplit()`. It takes a list of weights (`[0.8, 0.2]`) and a `seed` parameter to ensure the split is reproducible every time the code is run. Option D is the scikit-learn syntax.

Question 35

A data scientist is training a Support Vector Machine (SVM) model, which is sensitive to the scale of input features. Which Spark ML Transformer should they use to standardize the features?

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

✓ **Answer and Rationale** ›

Correct Answer: C

Rationale: Algorithms like SVMs (and Logistic Regression, K-Means) are sensitive to feature scales. The `StandardScaler` (or `MinMaxScaler`) is the Estimator used to scale the feature vector so that all features have a similar range, which improves model convergence and performance.

Question 36

What is a common drawback of using `OneHotEncoder` on a categorical feature with very high cardinality (e.g., a `zip_code` column with 50,000 unique values)?

- A. It can lead to a very sparse and high-dimensional feature vector.
- B. It does not work on string data and requires a `StringIndexer` first.
- C. It is an Estimator, which makes the pipeline slower.
- D. It can only be used for regression problems.

✓ **Answer and Rationale** ›

Correct Answer: A

Rationale: One-hot encoding creates a new binary feature for each unique category. A column with 50,000 unique values would be transformed into a vector with 50,000 dimensions (most of which are zero for any given row). This is known as the "curse of dimensionality" and can lead to performance issues and sparse data.

Question 37

When evaluating a regression model, a data scientist wants a metric that represents the proportion of the variance in the dependent variable that is predictable from the independent variables. Which metric should they use?

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

✓ **Answer and Rationale** ›

Correct Answer: A

Rationale: This is the text-book definition of R-squared (R^2). It is a measure of "goodness of fit" that indicates how much of the variance in the label is explained by the model. RMSE and MAE measure the error in the model's predictions, not the variance explained.

Question 38

A data scientist is performing hyperparameter tuning. They have a very large dataset and want to use `CrossValidator`, but the computational cost is too high. What is a less expensive, Spark-native alternative for validation that splits the data only once?

- A. `TrainValidationSplit`
- B. `Hyperopt` with `SparkTrials`
- C. Manually calling `.randomSplit()`
- D. `mlflow.autolog()`

✓ **Answer and Rationale** ›

Correct Answer: A

Rationale: `CrossValidator` is expensive because it splits the data *k* times and trains *k* models for each hyperparameter combination. `TrainValidationSplit` is a less expensive alternative. It performs only a single split (e.g., 80/20) and evaluates each hyperparameter combination just once, making it much faster.

Question 39

A data scientist is building a time-series forecasting model. They get poor results when using a standard `.randomSplit()` for validation. Why is this happening?

- A. The model is underfitting due to a lack of data.
- B. `.randomSplit()` shuffles the data, which breaks the temporal order and causes data leakage.

- C. The model requires feature scaling, which was not applied.
- D. The `seed` parameter was not set, making the split non-reproducible.

✓ Answer and Rationale >

Correct Answer: B

Rationale: Time-series data is ordered. A random split shuffles the data, meaning the model might be trained on data from Wednesday to predict data from Monday (data leakage). This breaks the temporal dependency and leads to models that perform well in validation but poorly in practice. Time-series data must be split chronologically (e.g., train on 2020-2022, test on 2023).

Question 40

A data scientist is using Hyperopt. They need to tune a scikit-learn `RandomForestClassifier` (a single-node model) but want to parallelize the tuning process across their Databricks cluster. How should they configure their Hyperopt `fmin()` call?

- A. Use `Trials()`, as scikit-learn is not a distributed library.
- B. Use `SparkTrials()`, which is designed to distribute the parallel training of single-node models.
- C. This is not possible; Hyperopt can only tune Spark ML models.
- D. Use `Trials()`, but set `parallelism=cluster.num_workers`.

✓ Answer and Rationale >

Correct Answer: B

Rationale: This is a key concept from your study guide. `SparkTrials` is used to parallelize the tuning of *single-node models* (like scikit-learn). Hyperopt distributes each trial (training a full scikit-learn model) to a different worker node. `Trials()` is used when the model itself is distributed (like a Spark ML model).

Question 41

A data scientist is using a `StringIndexer` and `OneHotEncoder` on their categorical features before feeding them into a `RandomForestClassifier`. A colleague notes that for tree-based models, the `OneHotEncoder` step is often unnecessary. Why?

- A. Tree models can natively handle string-based categorical features.
- B. Tree models can interpret the integer-based output of `StringIndexer` directly as categorical splits.
- C. `OneHotEncoder` makes the feature vectors too dense for tree models.
- D. `RandomForestClassifier` has its own built-in `OneHotEncoder`.

✓ Answer and Rationale >

Correct Answer: B

Rationale: Tree-based models (like Decision Trees and Random Forests) work by finding optimal split points. They can treat the integer output of `StringIndexer` (e.g., 0, 1, 2) as categorical groupings ("is feature in {0, 2}?"), so the `OneHotEncoder` step is redundant and adds unnecessary dimensionality.

Domain: Scaling ML Models (4 Questions)

Question 42

A company needs to generate personalized recommendations for all 5 million of its users once per day. The results are stored in a database for the website to read. Which deployment strategy is most appropriate and cost-effective?

- A. Real-time serving via Databricks Model Serving, as it has low latency.
- B. Batch inference, scheduled as a Databricks Job, because it can process large volumes of data efficiently in parallel.
- C. A Delta Live Tables pipeline, as it is best for streaming data.
- D. Deploying the model as a Pandas UDF.

✓ **Answer and Rationale** >

Correct Answer: B

Rationale: This is a classic batch inference use case. The key requirements are: 1) large volume of data (5 million users) and 2) results are not needed in real-time ("once per day"). A scheduled Databricks Job is the most cost-effective and efficient way to process this data in parallel.

Question 43

A credit card company needs to approve or deny transactions in less than 100 milliseconds. What deployment solution is required for this use case?

- A. A batch inference job that runs every minute.
- B. A real-time inference endpoint, such as one provided by Databricks Model Serving.
- C. A Databricks Job that saves results to a Delta table.
- D. An MLflow `mlproject` file executed by the transaction system.

✓ **Answer and Rationale** >

Correct Answer: B

Rationale: This is a classic real-time inference use case. The key requirement is low latency ("less than 100 milliseconds") for an immediate prediction. This requires a "live" model waiting for requests, which is provided by a REST API endpoint via Databricks Model Serving.

Question 44

A team has deployed a model to a Databricks Model Serving endpoint. The endpoint is only used during business hours (9 AM to 5 PM) and receives no traffic at night. What is the most effective way to minimize compute costs for this endpoint?

- A. Manually stop the endpoint every day at 5 PM and restart it at 9 AM.
- B. Configure the endpoint to "scale to zero" when not in use.

- C. Build a separate, smaller model for off-hours.
- D. Re-train the model on a smaller dataset to reduce its compute footprint.

✓ **Answer and Rationale** >

Correct Answer: B

Rationale: Databricks Model Serving endpoints have a "scale to zero" configuration. When enabled, the endpoint will automatically shut down its compute resources after a period of inactivity (e.g., at night) and then automatically "wake up" when it receives a new request (e.g., at 9 AM). This is the most effective and automated way to save costs.

Question 45

When running a large-scale batch inference job, the Spark `spark.read()` operation on the input Delta table is taking a long time. What Delta Lake optimization can significantly speed up data retrieval by physically reordering the data based on a specific column?

- A. Z-Ordering
- B. Auto Optimize
- C. Checkpointing
- D. Unity Catalog

✓ **Answer andR ationale** >

Correct Answer: A

Rationale: Z-Ordering is a Delta Lake optimization technique. It co-locates related data in the same set of files. If you frequently filter by a specific column (e.g., `user_id`), running `OPTIMIZE table_name ZORDER BY (user_id)` will physically reorder the data, allowing Spark to read only the relevant files (data skipping) and dramatically speeding up queries.