the data scientist to make updates and improvements to the feature engineering part of the preprocessing pipeline without affecting the main codebase that runs daily. By creating a new branch, they can work on their changes in isolation. Once the changes are ready and tested, they can be merged back into the main branch through a pull request, ensuring a smooth integration process and allowing for code review and collaboration with other team members.
Reference:
Databricks documentation on Git integration: Databricks Repos

Question: 42

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

A. Run each notebook interactively
B. Review the matrix view in the Job's runs
C. Migrate the Job to a Delta Live Tables pipeline
D. Change each Task's setting to use a dedicated cluster

Answer: B

Explanation:

To identify which task is causing the failure in the job, the team should review the matrix view in the Job's runs. The matrix view provides a clear and detailed overview of each task's status, allowing the team to quickly identify which task failed. This approach is more efficient than running each notebook interactively, as it provides immediate insights into the job's execution flow and any issues that occurred during the run.
Reference:
Databricks documentation on Jobs: Jobs in Databricks

Question: 43

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

A. Single Node cluster
B. Standard cluster
C. SQL Warehouse
D. None of these compute tools support this task

Answer: B

Explanation:

For a data scientist using Spark SQL to import data and then performing machine learning tasks using Spark ML, the best-suited compute tool is a Standard cluster. A Standard cluster in Databricks

provides the necessary resources and scalability to handle large datasets and perform distributed computing tasks efficiently, making it ideal for running Spark SQL and Spark ML operations.
Reference:
Databricks documentation on clusters: Clusters in Databricks

Question: 44

A machine learning engineer is trying to perform batch model inference. They want to get predictions using the linear regression model saved at the path model_uri for the DataFrame batch_df.

batch_df has the following schema:

customer_id STRING

The machine learning engineer runs the following code block to perform inference on batch_df using the linear regression model at model_uri:

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

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

A. When the Feature Store feature set was logged with the model at model_uri
B. When all of the features used by the model at model_uri are in a Spark DataFrame in the PySpark
C. When the model at model_uri only uses customer_id as a feature
D. This code block will not perform the desired inference in any situation.
E. When all of the features used by the model at model_uri are in a single Feature Store table

Answer: A

Explanation:

The code block provided by the machine learning engineer will perform the desired inference when the Feature Store feature set was logged with the model at model_uri. This ensures that all necessary feature transformations and metadata are available for the model to make predictions. The Feature Store in Databricks allows for seamless integration of features and models, ensuring that the required features are correctly used during inference.
Reference:
Databricks documentation on Feature Store: Feature Store in Databricks

Question: 45

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

A. F1
B. R-squared
C. MAE
D. MSE

Answer: A

Explanation:

The code block provided by the machine learning engineer will perform the desired inference when the Feature Store feature set was logged with the model at model_uri. This ensures that all necessary feature transformations and metadata are available for the model to make predictions.
The Feature Store in Databricks allows for seamless integration of features and models, ensuring that the required features are correctly used during inference.
Reference:
Databricks documentation on Feature Store: Feature Store in Databricks

Question: 46

A data scientist wants to use Spark ML to impute missing values in their PySpark DataFrame features_df. They want to replace missing values in all numeric columns in features_df with each respective numeric column's median value.

They have developed the following code block to accomplish this task:

```
imputer = Imputer(
    strategy="median",
    inputCols=input_columns,
    outputCols=output_columns
)
imputed_features_df = imputer.transform(features_df)
```

The code block is not accomplishing the task.

Which reasons describes why the code block is not accomplishing the imputation task?

A. It does not impute both the training and test data sets.
B. The inputCols and outputCols need to be exactly the same.
C. The fit method needs to be called instead of transform.
D. It does not fit the imputer on the data to create an ImputerModel.

Answer: D

Explanation:

In the provided code block, the Imputer object is created but not fitted on the data to generate an ImputerModel. The transform method is being called directly on the Imputer object, which does not yet contain the fitted median values needed for imputation. The correct approach is to fit the imputer on the dataset first.
Corrected code:
imputer = Imputer(strategy="median", inputCols=input_columns, outputCols=output_columns)
imputer_model = imputer.fit(features_df) # Fit the imputer to the data imputed_features_df = imputer_model.transform(features_df) # Transform the data using the fitted imputer
Reference:
PySpark ML Documentation

Question: 47

A data scientist wants to use Spark ML to one-hot encode the categorical features in their PySpark DataFrame features_df. A list of the names of the string columns is assigned to the input_columns variable.

They have developed this code block to accomplish this task:

```
ohe = OneHotEncoder(
    inputCols=input_columns,
    outputCols=output_columns
)
ohe_model = ohe.fit(features_df)
ohe_features_df = ohe_model.transform(features_df)
```

The code block is returning an error.

Which of the following adjustments does the data scientist need to make to accomplish this task?

A. They need to specify the method parameter to the OneHotEncoder.
B. They need to remove the line with the fit operation.
C. They need to use StringIndexer prior to one-hot encodinq the features.
D. They need to use VectorAssembler prior to one-hot encoding the features.

Answer: C

Explanation:

The OneHotEncoder in Spark ML requires numerical indices as inputs rather than string labels. Therefore, you need to first convert the string columns to numerical indices using StringIndexer. After

that, you can apply OneHotEncoder to these indices.
Corrected code:
from pyspark.ml.feature import StringIndexer, OneHotEncoder # Convert string column to index
indexers = [StringIndexer(inputCol=col, outputCol=col+"_index") for col in input_columns]
indexer_model = Pipeline(stages=indexers).fit(features_df) indexed_features_df =
indexer_model.transform(features_df) # One-hot encode the indexed columns ohe =
OneHotEncoder(inputCols=[col+"_index" for col in input_columns], outputCols=output_columns)
ohe_model = ohe.fit(indexed_features_df) ohe_features_df =
ohe_model.transform(indexed_features_df)
Reference:
PySpark ML Documentation

Question: 48

A data scientist is wanting to explore the Spark DataFrame spark_df. The data scientist wants visual histograms displaying the distribution of numeric features to be included in the exploration.
Which of the following lines of code can the data scientist run to accomplish the task?

A. spark_df.describe()
B. dbutils.data(spark_df).summarize()
C. This task cannot be accomplished in a single line of code.
D. spark_df.summary()
E. dbutils.data.summarize (spark_df)

Answer: E

Explanation:

To display visual histograms and summaries of the numeric features in a Spark DataFrame, the Databricks utility function dbutils.data.summarize can be used. This function provides a comprehensive summary, including visual histograms.
Correct code:
dbutils.data.summarize(spark_df)
Other options like spark_df.describe() and spark_df.summary() provide textual statistical summaries but do not include visual histograms.
Reference:
Databricks Utilities Documentation

Question: 49

A data scientist wants to efficiently tune the hyperparameters of a scikit-learn model in parallel. They elect to use the Hyperopt library to facilitate this process.
Which of the following Hyperopt tools provides the ability to optimize hyperparameters in parallel?

A. fmin
B. SparkTrials
C. quniform
D. search_space

E. objective_function

Answer: B

Explanation:

The SparkTrials class in the Hyperopt library allows for parallel hyperparameter optimization on a Spark cluster. This enables efficient tuning of hyperparameters by distributing the optimization process across multiple nodes in a cluster.
from hyperopt import fmin, tpe, hp, SparkTrials search_space = { 'x': hp.uniform('x', 0, 1), 'y': hp.uniform('y', 0, 1) } def objective(params): return params['x'] ** 2 + params['y'] ** 2 spark_trials = SparkTrials(parallelism=4) best = fmin(fn=objective, space=search_space, algo=tpe.suggest, max_evals=100, trials=spark_trials)
Reference:
Hyperopt Documentation

Question: 50

A data scientist uses 3-fold cross-validation and the following hyperparameter grid when optimizing model hyperparameters via grid search for a classification problem:
- Hyperparameter 1: [2, 5, 10]
- Hyperparameter 2: [50, 100]

Which of the following represents the number of machine learning models that can be trained in parallel during this process?

A. 3
B. 5
C. 6
D. 18

Answer: D

Explanation:

To determine the number of machine learning models that can be trained in parallel, we need to calculate the total number of combinations of hyperparameters. The given hyperparameter grid includes:
Hyperparameter 1: [2, 5, 10] (3 values)
Hyperparameter 2: [50, 100] (2 values)
The total number of combinations is the product of the number of values for each hyperparameter:
3 (values of Hyperparameter 1)×2 (values of Hyperparameter 2)=63 (values of Hyperparameter 1)×2 (values of Hyperparameter 2)=6
With 3-fold cross-validation, each combination of hyperparameters will be evaluated 3 times. Thus, the total number of models trained will be:
6 (combinations)×3 (folds)=186 (combinations)×3 (folds)=18
However, the number of models that can be trained in parallel is equal to the number of hyperparameter combinations, not the total number of models considering cross-validation.
Therefore, 6 models can be trained in parallel.
Reference:

Databricks documentation on hyperparameter tuning: Hyperparameter Tuning

Question: 51

An organization is developing a feature repository and is electing to one-hot encode all categorical feature variables. A data scientist suggests that the categorical feature variables should not be one-hot encoded within the feature repository.
Which of the following explanations justifies this suggestion?

A. One-hot encoding is a potentially problematic categorical variable strategy for some machine learning algorithms.
B. One-hot encoding is dependent on the target variable's values which differ for each apaplication.
C. One-hot encoding is computationally intensive and should only be performed on small samples of training sets for individual machine learning problems.
D. One-hot encoding is not a common strategy for representing categorical feature variables numerically.

Answer: A

Explanation:

The suggestion not to one-hot encode categorical feature variables within the feature repository is justified because one-hot encoding can be problematic for some machine learning algorithms. Specifically, one-hot encoding increases the dimensionality of the data, which can be computationally expensive and may lead to issues such as multicollinearity and overfitting. Additionally, some algorithms, such as tree-based methods, can handle categorical variables directly without requiring one-hot encoding.
Reference:
Databricks documentation on feature engineering: Feature Engineering

Question: 52

A data scientist has created a linear regression model that uses log(price) as a label variable. Using this model, they have performed inference and the predictions and actual label values are in Spark DataFrame preds_df.

They are using the following code block to evaluate the model:

regression_evaluator.setMetricName("rmse").evaluate(preds_df)

Which of the following changes should the data scientist make to evaluate the RMSE in a way that is comparable with price?

A. They should exponentiate the computed RMSE value
B. They should take the log of the predictions before computing the RMSE
C. They should evaluate the MSE of the log predictions to compute the RMSE
D. They should exponentiate the predictions before computing the RMSE

Answer: D

Explanation:

When evaluating the RMSE for a model that predicts log-transformed prices, the predictions need to be transformed back to the original scale to obtain an RMSE that is comparable with the actual price values. This is done by exponentiating the predictions before computing the RMSE. The RMSE should be computed on the same scale as the original data to provide a meaningful measure of error.
Reference:
Databricks documentation on regression evaluation: Regression Evaluation

Question: 53

A data scientist is working with a feature set with the following schema:

```
customer_id STRING,
spend DOUBLE,
units INTEGER,
loyalty_tier STRING
```

The customer_id column is the primary key in the feature set. Each of the columns in the feature set has missing values. They want to replace the missing values by imputing a common value for each feature.

Which of the following lists all of the columns in the feature set that need to be imputed using the most common value of the column?

A. customer_id, loyalty_tier
B. loyalty_tier
C. units
D. spend
E. customer_id

Answer: B

Explanation:

For the feature set schema provided, the columns that need to be imputed using the most common value (mode) are typically the categorical columns. In this case, loyalty_tier is the only categorical column that should be imputed using the most common value. customer_id is a unique identifier and should not be imputed, while spend and units are numerical columns that should typically be imputed using the mean or median values, not the mode.
Reference:
Databricks documentation on missing value imputation: Handling Missing Data
If you need any further clarification or additional questions answered, please let me know!

Question: 54

A data scientist has a Spark DataFrame spark_df. They want to create a new Spark DataFrame that contains only the rows from spark_df where the value in column discount is less than or equal 0.
Which of the following code blocks will accomplish this task?

A. spark_df.loc[:,spark_df["discount"] <= 0]
B. spark_df[spark_df["discount"] <= 0]
C. spark_df.filter (col("discount") <= 0)
D. spark_df.loc(spark_df["discount"] <= 0, :]

Answer: C

Explanation:

To filter rows in a Spark DataFrame based on a condition, the filter method is used. In this case, the condition is that the value in the "discount" column should be less than or equal to 0. The correct syntax uses the filter method along with the col function from pyspark.sql.functions.
Correct code:
from pyspark.sql.functions import col filtered_df = spark_df.filter(col("discount") <= 0)
Option A and D use Pandas syntax, which is not applicable in PySpark. Option B is closer but misses the use of the col function.
Reference:
PySpark SQL Documentation

Question: 55

A data scientist is performing hyperparameter tuning using an iterative optimization algorithm. Each evaluation of unique hyperparameter values is being trained on a single compute node. They are performing eight total evaluations across eight total compute nodes. While the accuracy of the model does vary over the eight evaluations, they notice there is no trend of improvement in the accuracy. The data scientist believes this is due to the parallelization of the tuning process.
Which change could the data scientist make to improve their model accuracy over the course of their tuning process?

A. Change the number of compute nodes to be half or less than half of the number of evaluations.
B. Change the number of compute nodes and the number of evaluations to be much larger but equal.
C. Change the iterative optimization algorithm used to facilitate the tuning process.
D. Change the number of compute nodes to be double or more than double the number of evaluations.

Answer: C

Explanation:

The lack of improvement in model accuracy across evaluations suggests that the optimization algorithm might not be effectively exploring the hyperparameter space. Iterative optimization algorithms like Tree-structured Parzen Estimators (TPE) or Bayesian Optimization can adapt based on

previous evaluations, guiding the search towards more promising regions of the hyperparameter space.
Changing the optimization algorithm can lead to better utilization of the information gathered during each evaluation, potentially improving the overall accuracy.
Reference:
Hyperparameter Optimization with Hyperopt

Question: 56

A data scientist learned during their training to always use 5-fold cross-validation in their model development workflow. A colleague suggests that there are cases where a train-validation split could be preferred over k-fold cross-validation when k > 2.
Which of the following describes a potential benefit of using a train-validation split over k-fold cross-validation in this scenario?

A. A holdout set is not necessary when using a train-validation split
B. Reproducibility is achievable when using a train-validation split
C. Fewer hyperparameter values need to be tested when using a train-validation split
D. Bias is avoidable when using a train-validation split
E. Fewer models need to be trained when using a train-validation split

Answer: E

Explanation:

A train-validation split is often preferred over k-fold cross-validation (with k > 2) when computational efficiency is a concern. With a train-validation split, only two models (one on the training set and one on the validation set) are trained, whereas k-fold cross-validation requires training k models (one for each fold).
This reduction in the number of models trained can save significant computational resources and time, especially when dealing with large datasets or complex models.
Reference:
Model Evaluation with Train-Test Split

Question: 57

Which of the following hyperparameter optimization methods automatically makes informed selections of hyperparameter values based on previous trials for each iterative model evaluation?

A. Random Search
B. Halving Random Search
C. Tree of Parzen Estimators
D. Grid Search

Answer: C

Explanation:

Tree of Parzen Estimators (TPE) is a sequential model-based optimization algorithm that selects