Question 1

Question Type: MultipleChoice

Which of the following tools can be used to parallelize the hyperparameter tuning process for single-node machine learning models using a Spark cluster?

Options:

A- MLflow Experiment Tracking
B- Spark ML
C- Autoscaling clusters
D- Autoscaling clusters
E- Delta Lake

Answer:

B

Explanation:

Spark ML (part of Apache Spark's MLlib) is designed to handle machine learning tasks across multiple nodes in a cluster, effectively parallelizing tasks like hyperparameter tuning. It supports various machine learning algorithms that can be optimized over a Spark cluster, making it suitable for parallelizing hyperparameter tuning for single-node machine learning models when they are adapted to run on Spark.

Reference

Apache Spark MLlib Guide: https://spark.apache.org/docs/latest/ml-guide.html

Spark ML is a library within Apache Spark designed for scalable machine learning. It provides tools to handle large-scale machine learning tasks, including parallelizing the hyperparameter tuning process for single-node machine learning models using a Spark cluster. Here's a detailed explanation of how Spark ML can be used:

Hyperparameter Tuning with CrossValidator: Spark ML includes the CrossValidator and TrainValidationSplit classes, which are used for hyperparameter tuning. These classes can evaluate multiple sets of hyperparameters in parallel using a Spark cluster.

from pyspark.ml.tuning import CrossValidator, ParamGridBuilder

from pyspark.ml.evaluation import BinaryClassificationEvaluator

# Define the model

```
model = ...

# Create a parameter grid

paramGrid = ParamGridBuilder() \

.addGrid(model.hyperparam1, [value1, value2]) \

.addGrid(model.hyperparam2, [value3, value4]) \

.build()

# Define the evaluator

evaluator = BinaryClassificationEvaluator()

# Define the CrossValidator

crossval = CrossValidator(estimator=model,

estimatorParamMaps=paramGrid,

evaluator=evaluator,

numFolds=3)
```

Parallel Execution: Spark distributes the tasks of training models with different hyperparameters across the cluster's nodes. Each node processes a subset of the parameter grid, which allows multiple models to be trained simultaneously.

Scalability: Spark ML leverages the distributed computing capabilities of Spark. This allows for efficient processing of large datasets and training of models across many nodes, which speeds up the hyperparameter tuning process significantly compared to single-node computations.

Reference

Apache Spark MLlib Documentation

Hyperparameter Tuning in Spark ML

Question 2

Question Type: MultipleChoice

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

Which of the following approaches describes how the machine learning engineer can begin using

Databricks Runtime for Machine Learning?

Options:

A- They can add a line enabling Databricks Runtime ML in their init script when creating their clusters.
B- They can check the Databricks Runtime ML box when creating their clusters.
C- They can select a Databricks Runtime ML version from the Databricks Runtime Version dropdown when creating their clusters.
D- They can set the runtime-version variable in their Spark session to ''ml''.

Answer:

C

Explanation:

The Databricks Runtime for Machine Learning includes pre-installed packages and libraries essential for machine learning and deep learning, including MLflow. To use it, the machine learning engineer can simply select an appropriate Databricks Runtime ML version from the 'Databricks Runtime Version' dropdown menu while creating their cluster. This selection ensures that all necessary machine learning libraries, including MLflow, are pre-installed and ready for use, avoiding the need to manually install them each time.

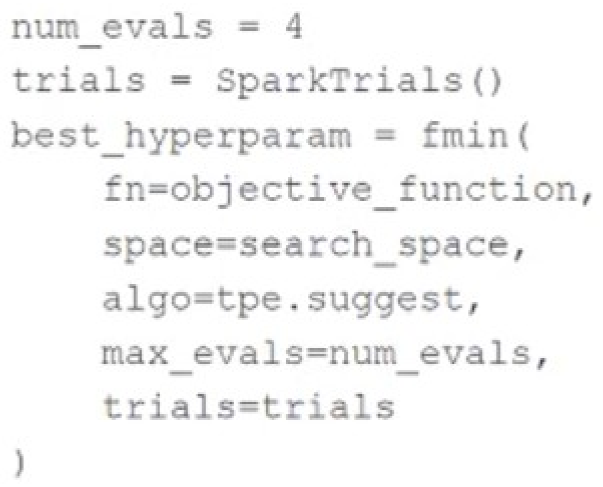
Reference

Databricks documentation on creating clusters: https://docs.databricks.com/clusters/create.html

Question 3

Question Type: MultipleChoice

A data scientist is using the following code block to tune hyperparameters for a machine learning model:



```
num_evals = 4
trials = SparkTrials()
best_hyperparam = fmin(
     fn=objective_function,
     space=search_space,
     algo=tpe.suggest,
     max_evals=num_evals,
     trials=trials
)
```

Which change can they make the above code block to improve the likelihood of a more accurate model?

Options:

A- Increase num_evals to 100
B- Change fmin() to fmax()
C- Change sparkTrials() to Trials()
D- Change tpe.suggest to random.suggest

Answer:

A

Explanation:

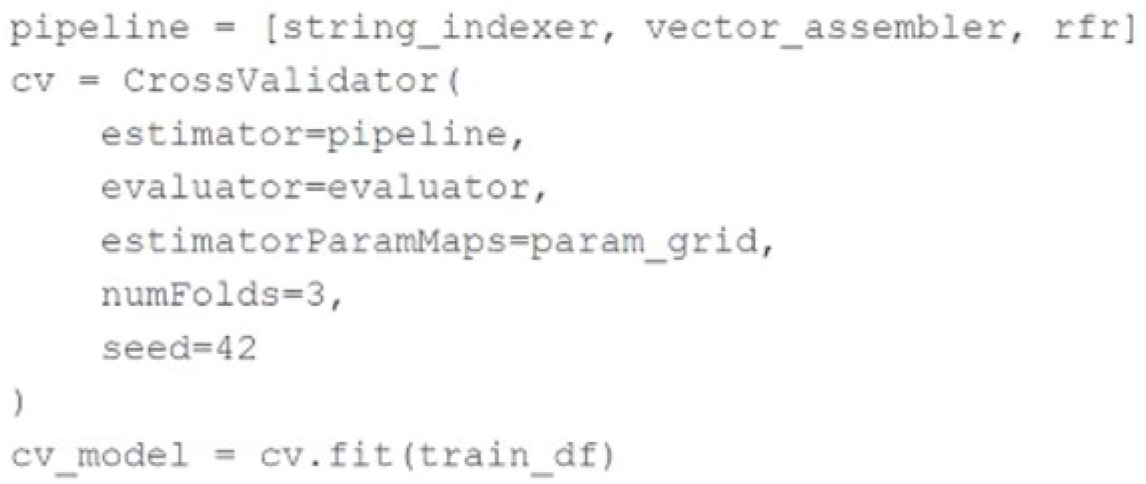
To improve the likelihood of a more accurate model, the data scientist can increase num_evals to 100. Increasing the number of evaluations allows the hyperparameter tuning process to explore a larger search space and evaluate more combinations of hyperparameters, which increases the chance of finding a more optimal set of hyperparameters for the model.

Databricks documentation on hyperparameter tuning: Hyperparameter Tuning

Question 4

Question Type: MultipleChoice

A data scientist has developed a random forest regressor rfr and included it as the final stage in a Spark MLPipeline pipeline. They then set up a cross-validation process with pipeline as the estimator in the following code block:



```
pipeline = [string_indexer, vector_assembler, rfr]
cv = CrossValidator(
     estimator=pipeline,
     evaluator=evaluator,
     estimatorParamMaps=param_grid,
     numFolds=3,
     seed=42
)
cv_model = cv.fit(train_df)
```

Which of the following is a negative consequence of including pipeline as the estimator in the cross-validation process rather than rfr as the estimator?

Options:

A- The process will have a longer runtime because all stages of pipeline need to be refit or retransformed with each mode

B- The process will leak data from the training set to the test set during the evaluation phase

C- The process will be unable to parallelize tuning due to the distributed nature of pipeline

D- The process will leak data prep information from the validation sets to the training sets for each model

Answer:

A

Explanation:

Including the entire pipeline as the estimator in the cross-validation process means that all stages of the pipeline, including data preprocessing steps like string indexing and vector assembling, will be refit or retransformed for each fold of the cross-validation. This results in a longer runtime because each fold requires re-execution of these preprocessing steps, which can be computationally expensive.

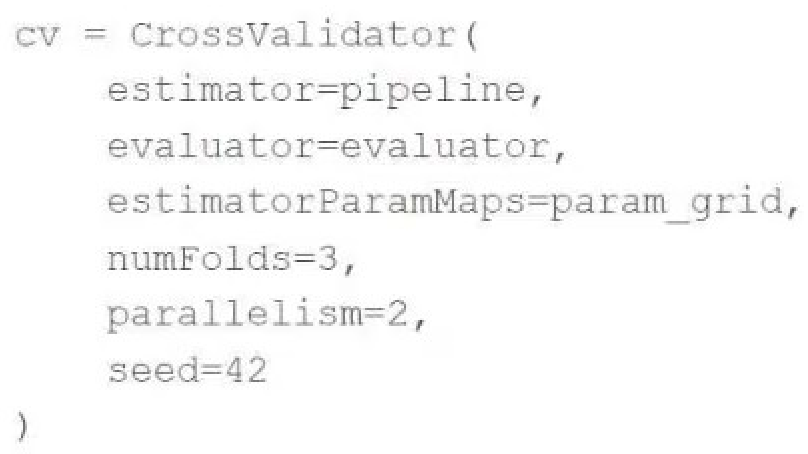
If only the random forest regressor (rfr) were included as the estimator, the preprocessing steps would be performed once, and only the model fitting would be repeated for each fold, significantly reducing the computational overhead.

Databricks documentation on cross-validation: Cross Validation

Question 5

Question Type: MultipleChoice

A machine learning engineer is trying to scale a machine learning pipeline pipeline that contains multiple feature engineering stages and a modeling stage. As part of the cross-validation process, they are using the following code block:



```
cv = CrossValidator(
     estimator=pipeline,
     evaluator=evaluator,
     estimatorParamMaps=param_grid,
     numFolds=3,
     parallelism=2,
     seed=42
)
```

A colleague suggests that the code block can be changed to speed up the tuning process by passing the model object to the estimator parameter and then placing the updated cv object as the final stage of the pipeline in place of the original model.

Which of the following is a negative consequence of the approach suggested by the colleague?

Options:

A- The model will take longer to train for each unique combination of hvperparameter values
B- The feature engineering stages will be computed using validation data
C- The cross-validation process will no longer be
D- The cross-validation process will no longer be reproducible
E- The model will be refit one more per cross-validation fold

Answer:

B

Explanation:

If the model object is passed to the estimator parameter of CrossValidator and the cross-validation object itself is placed as a stage in the pipeline, the feature engineering stages within the pipeline would be applied separately to each training and validation fold during cross-validation. This leads to a significant issue: the feature engineering stages would be computed using validation data, thereby leaking information from the validation set into the training process. This would potentially invalidate the cross-validation results by giving an overly optimistic performance estimate. Reference:

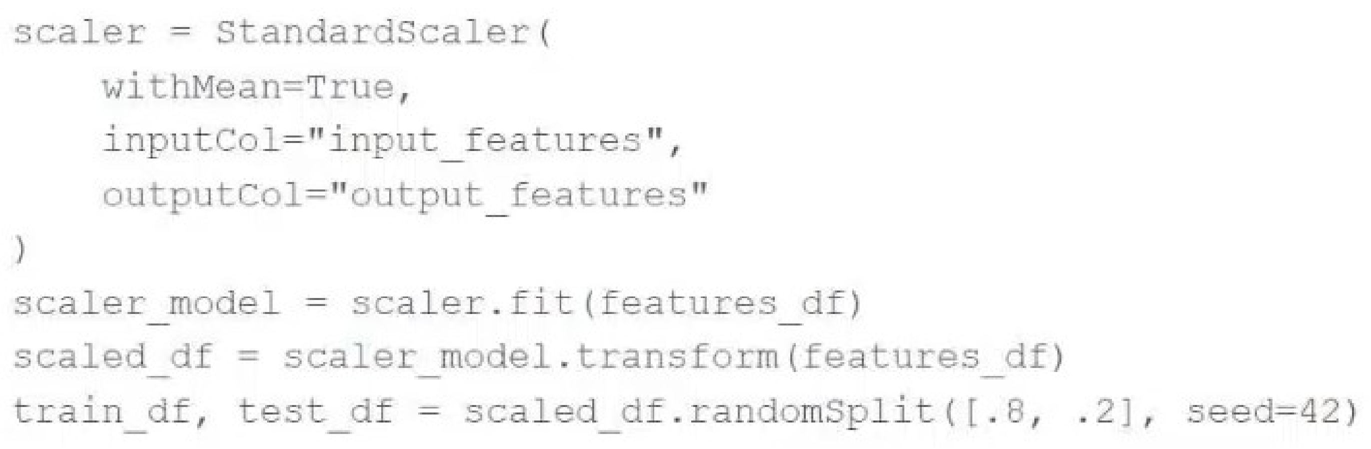
Cross-validation and Pipeline Integration in MLlib (Avoiding Data Leakage in Pipelines).

Question 6

Question Type: MultipleChoice

A data scientist is using Spark ML to engineer features for an exploratory machine learning project.

They decide they want to standardize their features using the following code block:



```
scaler = StandardScaler(
    withMean=True,
    inputCol="input_features",
    outputCol="output_features"
)
scaler_model = scaler.fit(features_df)
scaled_df = scaler_model.transform(features_df)
train_df, test_df = scaled_df.randomSplit([.8, .2], seed=42)
```

Upon code review, a colleague expressed concern with the features being standardized prior to splitting the data into a training set and a test set.

Which of the following changes can the data scientist make to address the concern?

Options:

A- Utilize the MinMaxScaler object to standardize the training data according to global minimum and maximum values

B- Utilize the MinMaxScaler object to standardize the test data according to global minimum and maximum values

C- Utilize a cross-validation process rather than a train-test split process to remove the need for standardizing data

D- Utilize the Pipeline API to standardize the training data according to the test data's summary statistics

E- Utilize the Pipeline API to standardize the test data according to the training data's summary statistics

Answer:

E

Explanation:

To address the concern about standardizing features prior to splitting the data, the correct approach is to use the Pipeline API to ensure that only the training data's summary statistics are used to standardize the test data. This is achieved by fitting the StandardScaler (or any scaler) on the training data and then transforming both the training and test data using the fitted scaler. This approach prevents information leakage from the test data into the model training process and ensures that the model is evaluated fairly. Reference:

Best Practices in Preprocessing in Spark ML (Handling Data Splits and Feature Standardization).