Databricks.Databricks-Machine-Learning-Associate.v2024-07-19.q27

Exam Code:	Databricks-Machine-Learning-Associate
Exam Name:	Databricks Certified Machine Learning Associate Exam
Certification Provider:	Databricks
Free Question Number:	27
Version:	v2024-07-19
# of views:	290
# of Questions views:	270
https://www.exam-tests.com/Databricks-Machine-Learning-Associate-exam/Databricks.Databricks-Machine-Learning-Associate.v2024-07-19.q27.html	

NEW QUESTION: 1
A machine learning engineer wants to parallelize the inference of group-specific models using the Pandas Function API. They have developed the apply_model function that will look up and load the correct model for each group, and they want to apply it to each group of DataFrame df.
They have written the following incomplete code block:

```
prediction_df = (df
    .groupby("device_id")
    .______(apply_model, schema=apply_return_schema)
)
```


Which piece of code can be used to fill in the above blank to complete the task?
A. applyInPandas
B. groupedApplyInPandas
C. mapInPandas
D. predict
Answer: (SHOW ANSWER)
To parallelize the inference of group-specific models using the Pandas Function API in PySpark, you can use the applyInPandas function. This function allows you to apply a Python function on each group of a DataFrame and return a DataFrame, leveraging the power of pandas UDFs (user-defined functions) for better performance.
prediction_df = (df.groupby("device_id") .applyInPandas(apply_model, schema=apply_return_schema)) In this code:
groupby("device_id"): Groups the DataFrame by the "device_id" column.
applyInPandas(apply_model, schema=apply_return_schema): Applies the apply_model function to each group and specifies the schema of the return DataFrame.

Reference:
PySpark Pandas UDFs Documentation

NEW QUESTION: 2
A data scientist has developed a linear regression model using Spark ML and computed the predictions in a Spark DataFrame preds_df with the following schema:
prediction DOUBLE
actual DOUBLE
Which of the following code blocks can be used to compute the root mean-squared-error of the model according to the data in preds_df and assign it to the rmse variable?

A.


B.


C.


D.


Answer: B (LEAVE A REPLY)
The code block to compute the root mean-squared error (RMSE) for a linear regression model in Spark ML should use the RegressionEvaluator class with metricName set to "rmse". Given the schema of preds_df with columns prediction and actual, the correct evaluator setup will specify predictionCol="prediction" and labelCol="actual". Thus, the appropriate code block (Option C in your list) that uses RegressionEvaluator to compute the RMSE is the correct choice. This setup correctly measures the performance of the regression model using the predictions and actual outcomes from the DataFrame.

Reference:
Spark ML documentation (Using RegressionEvaluator to Compute RMSE).

NEW QUESTION: 3
What is the name of the method that transforms categorical features into a series of binary indicator feature variables?
A. Leave-one-out encoding
B. Target encoding
C. One-hot encoding
D. Categorical
E. String indexing
Answer: C (LEAVE A REPLY)
The method that transforms categorical features into a series of binary indicator variables is known as one-hot encoding. This technique converts each categorical value into a new binary column, which is essential for models that require numerical input. One-hot encoding is widely used because it helps to handle categorical data without introducing a false ordinal relationship among categories.
Reference:
Feature Engineering Techniques (One-Hot Encoding).

NEW QUESTION: 4
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?
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: (SHOW ANSWER)
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).

NEW QUESTION: 5
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: (SHOW ANSWER)
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

NEW QUESTION: 6
A data scientist has replaced missing values in their feature set with each respective feature variable's median value. A colleague suggests that the data scientist is throwing away valuable information by doing this.
Which of the following approaches can they take to include as much information as possible in the feature set?

A. Impute the missing values using each respective feature variable's mean value instead of the median value
B. Refrain from imputing the missing values in favor of letting the machine learning algorithm determine how to handle them
C. Remove all feature variables that originally contained missing values from the feature set
D. Create a binary feature variable for each feature that contained missing values indicating whether each row's value has been imputed
E. Create a constant feature variable for each feature that contained missing values indicating the percentage of rows from the feature that was originally missing
Answer: D (LEAVE A REPLY)
By creating a binary feature variable for each feature with missing values to indicate whether a value has been imputed, the data scientist can preserve information about the original state of the data. This approach maintains the integrity of the dataset by marking which values are original and which are synthetic (imputed). Here are the steps to implement this approach:
Identify Missing Values: Determine which features contain missing values.
Impute Missing Values: Continue with median imputation or choose another method (mean, mode, regression, etc.) to fill missing values.
Create Indicator Variables: For each feature that had missing values, add a new binary feature. This feature should be '1' if the original value was missing and imputed, and '0' otherwise.
Data Integration: Integrate these new binary features into the existing dataset. This maintains a record of where data imputation occurred, allowing models to potentially weight these observations differently.
Model Adjustment: Adjust machine learning models to account for these new features, which might involve considering interactions between these binary indicators and other features.
Reference
"Feature Engineering for Machine Learning" by Alice Zheng and Amanda Casari (O'Reilly Media, 2018), especially the sections on handling missing data.
Scikit-learn documentation on imputing missing values: https://scikit-learn.org/stable/modules/impute.html

NEW QUESTION: 7
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:

```
.mputer = Imputer(
    strategy="median",
    inputCols=input_columns,
    outputCols=output_columns

.mputed_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 (LEAVE A REPLY)

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

NEW QUESTION: 8

A health organization is developing a classification model to determine whether or not a patient currently has a specific type of infection. The organization's leaders want to maximize the number of positive cases identified by the model.

Which of the following classification metrics should be used to evaluate the model?

A. RMSE

B. Precision

C. Area under the residual operating curve

D. Accuracy

E. Recall

Answer: E (LEAVE A REPLY)

When the goal is to maximize the identification of positive cases in a classification task, the metric of interest is Recall. Recall, also known as sensitivity, measures the proportion of actual positives that are correctly identified by the model (i.e., the true positive rate). It is crucial for scenarios where missing a positive case (false negative) has serious implications, such as in medical diagnostics. The other metrics like Precision, RMSE, and Accuracy serve different aspects of

performance measurement and are not specifically focused on maximizing the detection of positive cases alone.
Reference:
Classification Metrics in Machine Learning (Understanding Recall).

NEW QUESTION: 9
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 price is greater than 0.
Which of the following code blocks will accomplish this task?
A. spark_df[spark_df["price"] > 0]
B. spark_df.filter(col("price") > 0)
C. SELECT * FROM spark_df WHERE price > 0
D. spark_df.loc[spark_df["price"] > 0,:]
E. spark_df.loc[:,spark_df["price"] > 0]
Answer: B (LEAVE A REPLY)
To filter rows in a Spark DataFrame based on a condition, you use the filter method along with a column condition. The correct syntax in PySpark to accomplish this task is spark_df.filter(col("price") > 0), which filters the DataFrame to include only those rows where the value in the "price" column is greater than 0. The col function is used to specify column-based operations. The other options provided either do not use correct Spark DataFrame syntax or are intended for different types of data manipulation frameworks like pandas.
Reference:
PySpark DataFrame API documentation (Filtering DataFrames).

NEW QUESTION: 10
Which of the Spark operations can be used to randomly split a Spark DataFrame into a training DataFrame and a test DataFrame for downstream use?
A. TrainValidationSplit
B. DataFrame.where
C. CrossValidator
D. TrainValidationSplitModel
E. DataFrame.randomSplit
Answer: E (LEAVE A REPLY)
The correct method to randomly split a Spark DataFrame into training and test sets is by using the randomSplit method. This method allows you to specify the proportions for the split as a list of weights and returns multiple DataFrames according to those weights. This is directly intended for splitting DataFrames randomly and is the appropriate choice for preparing data for training and testing in machine learning workflows.
Reference:
Apache Spark DataFrame API documentation (DataFrame Operations: randomSplit).

NEW QUESTION: 11
Which of the following tools can be used to distribute large-scale feature engineering without the use of a UDF or pandas Function API for machine learning pipelines?
A. Keras
B. pandas
C. PvTorch
D. Spark ML
E. Scikit-learn
Answer: D (LEAVE A REPLY)
Spark ML (Machine Learning Library) is designed specifically for handling large-scale data processing and machine learning tasks directly within Apache Spark. It provides tools and APIs for large-scale feature engineering without the need to rely on user-defined functions (UDFs) or pandas Function API, allowing for more scalable and efficient data transformations directly distributed across a Spark cluster. Unlike Keras, pandas, PyTorch, and scikit-learn, Spark ML operates natively in a distributed environment suitable for big data scenarios.
Reference:
Spark MLlib documentation (Feature Engineering with Spark ML).

NEW QUESTION: 12
In which of the following situations is it preferable to impute missing feature values with their median value over the mean value?
A. When the features are of the categorical type
B. When the features are of the boolean type
C. When the features contain a lot of extreme outliers
D. When the features contain no outliers
E. When the features contain no missing no values
Answer: C (LEAVE A REPLY)
Imputing missing values with the median is often preferred over the mean in scenarios where the data contains a lot of extreme outliers. The median is a more robust measure of central tendency in such cases, as it is not as heavily influenced by outliers as the mean. Using the median ensures that the imputed values are more representative of the typical data point, thus preserving the integrity of the dataset's distribution. The other options are not specifically relevant to the question of handling outliers in numerical data.
Reference:
Data Imputation Techniques (Dealing with Outliers).

NEW QUESTION: 13
Which of the following machine learning algorithms typically uses bagging?
A. Gradient boosted trees
B. K-means
C. Random forest

D. Linear regression
E. Decision tree
Answer: C (LEAVE A REPLY)
Random Forest is a machine learning algorithm that typically uses bagging (Bootstrap Aggregating). Bagging involves training multiple models independently on different random subsets of the data and then combining their predictions. Random Forests consist of many decision trees trained on random subsets of the training data and features, and their predictions are averaged to improve accuracy and control overfitting. This method enhances model robustness and predictive performance.
Reference:
Ensemble Methods in Machine Learning (Understanding Bagging and Random Forests).

NEW QUESTION: 14
A data scientist is developing a machine learning pipeline using AutoML on Databricks Machine Learning.
Which of the following steps will the data scientist need to perform outside of their AutoML experiment?
A. Model tuning
B. Model evaluation
C. Model deployment
D. Exploratory data analysis
Answer: D (LEAVE A REPLY)
AutoML platforms, such as the one available in Databricks Machine Learning, streamline various stages of the machine learning pipeline including feature engineering, model selection, hyperparameter tuning, and model evaluation. However, exploratory data analysis (EDA) is typically performed outside the AutoML process. EDA involves understanding the dataset, visualizing distributions, identifying anomalies, and gaining insights into data before feeding it into a machine learning pipeline. This step is crucial for ensuring that the data is clean and suitable for model training but is generally done manually by the data scientist.
Reference
Databricks documentation on AutoML: https://docs.databricks.com/applications/machine-learning/automl.html

NEW QUESTION: 15
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: (SHOW ANSWER)
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

NEW QUESTION: 16
A machine learning engineer is trying to scale a machine learning pipeline by distributing its single-node model tuning process. After broadcasting the entire training data onto each core, each core in the cluster can train one model at a time. Because the tuning process is still running slowly, the engineer wants to increase the level of parallelism from 4 cores to 8 cores to speed up the tuning process. Unfortunately, the total memory in the cluster cannot be increased.
In which of the following scenarios will increasing the level of parallelism from 4 to 8 speed up the tuning process?
A. When the tuning process in randomized
B. When the entire data can fit on each core
C. When the model is unable to be parallelized
D. When the data is particularly long in shape
E. When the data is particularly wide in shape
Answer: B (LEAVE A REPLY)
Increasing the level of parallelism from 4 to 8 cores can speed up the tuning process if each core can handle the entire dataset. This ensures that each core can independently work on training a model without running into memory constraints. If the entire dataset fits into the memory of each core, adding more cores will allow more models to be trained in parallel, thus speeding up the process.
Reference:
Parallel Computing Concepts

NEW QUESTION: 17
A data scientist is using MLflow to track their machine learning experiment. As a part of each of their MLflow runs, they are performing hyperparameter tuning. The data scientist would like to have one parent run for the tuning process with a child run for each unique combination of hyperparameter values. All parent and child runs are being manually started with mlflow.start_run. Which of the following approaches can the data scientist use to accomplish this MLflow run organization?
A. They can turn on Databricks Autologging
B. They can specify nested=True when starting the child run for each unique combination of hyperparameter values
C. They can start each child run inside the parent run's indented code block using mlflow.start runO
D. They can start each child run with the same experiment ID as the parent run
E. They can specify nested=True when starting the parent run for the tuning process
Answer: (SHOW ANSWER)
To organize MLflow runs with one parent run for the tuning process and a child run for each unique combination of hyperparameter values, the data scientist can specify nested=True when starting the child run. This approach ensures that each child run is properly nested under the parent run, maintaining a clear hierarchical structure for the experiment. This nesting helps in tracking and comparing different hyperparameter combinations within the same tuning process.
Reference:
MLflow Documentation (Managing Nested Runs).

NEW QUESTION: 18
A data scientist uses 3-fold cross-validation when optimizing model hyperparameters for a regression problem. The following root-mean-squared-error values are calculated on each of the validation folds:
* 10.0
* 12.0
* 17.0
Which of the following values represents the overall cross-validation root-mean-squared error?
A. 13.0
B. 17.0
C. 12.0
D. 39.0
E. 10.0
Answer: A (LEAVE A REPLY)

To calculate the overall cross-validation root-mean-squared error (RMSE), you average the RMSE values obtained from each validation fold. Given the RMSE values of 10.0, 12.0, and 17.0 for the three folds, the overall cross-validation RMSE is calculated as the average of these three values:
Overall CV RMSE=10.0+12.0+17.03=39.03=13.0Overall CV RMSE=310.0+12.0+17.0=339.0=13.0 Thus, the correct answer is 13.0, which accurately represents the average RMSE across all folds.
Reference:
Cross-validation in Regression (Understanding Cross-Validation Metrics).

NEW QUESTION: 19
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: (SHOW ANSWER)
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

NEW QUESTION: 20
A data scientist wants to efficiently tune the hyperparameters of a scikit-learn model. They elect to use the Hyperopt library's fmin operation to facilitate this process. Unfortunately, the final model is

not very accurate. The data scientist suspects that there is an issue with the objective_function being passed as an argument to fmin.
They use the following code block to create the objective_function:

```
def objective_function(params):
    max_depth = params["max_depth"]
    max_features = params["max_features"]
    regressor = RandomForestRegressor(
        max_depth=max_depth,
        max_features=max_features
    r2 = mean(cross_val_score(regressor, X_train, y_train, cv=3))
    return r2
```


Which of the following changes does the data scientist need to make to their objective_function in order to produce a more accurate model?
A. Add test set validation process
B. Add a random_state argument to the RandomForestRegressor operation
C. Remove the mean operation that is wrapping the cross_val_score operation
D. Replace the r2 return value with -r2
E. Replace the fmin operation with the fmax operation
Answer: D (LEAVE A REPLY)
When using the Hyperopt library with fmin, the goal is to find the minimum of the objective function. Since you are using cross_val_score to calculate the R2 score which is a measure of the proportion of the variance for a dependent variable that's explained by an independent variable(s) in a regression model, higher values are better. However, fmin seeks to minimize the objective function, so to align with fmin's goal, you should return the negative of the R2 score (-r2). This way, by minimizing the negative R2, fmin is effectively maximizing the R2 score, which can lead to a more accurate model.
Reference
Hyperopt Documentation: http://hyperopt.github.io/hyperopt/
Scikit-Learn documentation on model evaluation: https://scikit-learn.org/stable/modules/model_evaluation.html

NEW QUESTION: 21
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. Bias is avoidable when using a train-validation split
D. Fewer hyperparameter values need to be tested when using a train-validation split
E. Fewer models need to be trained when using a train-validation split
Answer: E (LEAVE A REPLY)

NEW QUESTION: 22
A data scientist wants to parallelize the training of trees in a gradient boosted tree to speed up the training process. A colleague suggests that parallelizing a boosted tree algorithm can be difficult. Which of the following describes why?
A. Gradient boosting is not a linear algebra-based algorithm which is required for parallelization
B. Gradient boosting requires access to all data at once which cannot happen during parallelization.
C. Gradient boosting calculates gradients in evaluation metrics using all cores which prevents parallelization.
D. Gradient boosting is an iterative algorithm that requires information from the previous iteration to perform the next step.
Answer: D (LEAVE A REPLY)
Gradient boosting is fundamentally an iterative algorithm where each new tree is built based on the errors of the previous ones. This sequential dependency makes it difficult to parallelize the training of trees in gradient boosting, as each step relies on the results from the preceding step. Parallelization in this context would undermine the core methodology of the algorithm, which depends on sequentially improving the model's performance with each iteration.
Reference:
Machine Learning Algorithms (Challenges with Parallelizing Gradient Boosting).
Gradient boosting is an ensemble learning technique that builds models in a sequential manner. Each new model corrects the errors made by the previous ones. This sequential dependency means that each iteration requires the results of the previous iteration to make corrections. Here is a step-by-step explanation of why this makes parallelization challenging:
Sequential Nature: Gradient boosting builds one tree at a time. Each tree is trained to correct the residual errors of the previous trees. This requires the model to complete one iteration before starting the next.
Dependence on Previous Iterations: The gradient calculation at each step depends on the predictions made by the previous models. Therefore, the model must wait until the previous tree has been fully trained and evaluated before starting to train the next tree.
Difficulty in Parallelization: Because of this dependency, it is challenging to parallelize the training process. Unlike algorithms that process data independently in each step (e.g., random forests), gradient boosting cannot easily distribute the work across multiple processors or cores for simultaneous execution.
This iterative and dependent nature of the gradient boosting process makes it difficult to parallelize effectively.
Reference

Gradient Boosting Machine Learning Algorithm
Understanding Gradient Boosting Machines

NEW QUESTION: 23
A data scientist has developed a machine learning pipeline with a static input data set using Spark ML, but the pipeline is taking too long to process. They increase the number of workers in the cluster to get the pipeline to run more efficiently. They notice that the number of rows in the training set after reconfiguring the cluster is different from the number of rows in the training set prior to reconfiguring the cluster.
Which of the following approaches will guarantee a reproducible training and test set for each model?
A. Manually configure the cluster
B. Write out the split data sets to persistent storage
C. Set a speed in the data splitting operation
D. Manually partition the input data
Answer: B (LEAVE A REPLY)
To ensure reproducible training and test sets, writing the split data sets to persistent storage is a reliable approach. This allows you to consistently load the same training and test data for each model run, regardless of cluster reconfiguration or other changes in the environment.
Correct approach:
Split the data.
Write the split data to persistent storage (e.g., HDFS, S3).
Load the data from storage for each model training session.
train_df, test_df = spark_df.randomSplit([0.8, 0.2], seed=42)
train_df.write.parquet("path/to/train_df.parquet") test_df.write.parquet("path/to/test_df.parquet") # Later, load the data train_df = spark.read.parquet("path/to/train_df.parquet") test_df = spark.read.parquet("path/to/test_df.parquet") Reference:
Spark DataFrameWriter Documentation

NEW QUESTION: 24
A data scientist has created two linear regression models. The first model uses price as a label variable and the second model uses log(price) as a label variable. When evaluating the RMSE of each model by comparing the label predictions to the actual price values, the data scientist notices that the RMSE for the second model is much larger than the RMSE of the first model.
Which of the following possible explanations for this difference is invalid?
A. The second model is much more accurate than the first model
B. The data scientist failed to exponentiate the predictions in the second model prior to computing the RMSE
C. The data scientist failed to take the log of the predictions in the first model prior to computing the RMSE
D. The first model is much more accurate than the second model

E. The RMSE is an invalid evaluation metric for regression problems

Answer: E (LEAVE A REPLY)

The Root Mean Squared Error (RMSE) is a standard and widely used metric for evaluating the accuracy of regression models. The statement that it is invalid is incorrect. Here's a breakdown of why the other statements are or are not valid:

Transformations and RMSE Calculation: If the model predictions were transformed (e.g., using log), they should be converted back to their original scale before calculating RMSE to ensure accuracy in the evaluation. Missteps in this conversion process can lead to misleading RMSE values.

Accuracy of Models: Without additional information, we can't definitively say which model is more accurate without considering their RMSE values properly scaled back to the original price scale.

Appropriateness of RMSE: RMSE is entirely valid for regression problems as it provides a measure of how accurately a model predicts the outcome, expressed in the same units as the dependent variable.

Reference

"Applied Predictive Modeling" by Max Kuhn and Kjell Johnson (Springer, 2013), particularly the chapters discussing model evaluation metrics.

NEW QUESTION: 25

A data scientist is utilizing MLflow Autologging to automatically track their machine learning experiments. After completing a series of runs for the experiment experiment_id, the data scientist wants to identify the run_id of the run with the best root-mean-square error (RMSE).

Which of the following lines of code can be used to identify the run_id of the run with the best RMSE in experiment_id?

A.
```
mlflow.search_runs(
    experiment_id,
    order_by=["metrics.rmse DESC"]
)["run_id"][0]
```

B.
```
mlflow.best_run(
    experiment_id,
    order_by=["metrics.rmse"]
)
```

C.
```
mlflow.search_runs(
    experiment_id,
    order_by=["metrics.rmse"]
)["run_id"][0]
```

```
mlflow.best_run(
    experiment_id,
    order_by = ["metrics.rmse DESC"]
)
```


D.

Answer: (SHOW ANSWER)

To find the run_id of the run with the best root-mean-square error (RMSE) in an MLflow experiment, the correct line of code to use is:

mlflow.search_runs(experiment_id, order_by=["metrics.rmse"])["run_id"][0] This line of code searches the runs in the specified experiment, orders them by the RMSE metric in ascending order (the lower the RMSE, the better), and retrieves the run_id of the best-performing run.

Option C correctly represents this logic.

Reference

MLflow documentation on tracking experiments:

https://www.mlflow.org/docs/latest/python_api/mlflow.html#mlflow.search_runs

NEW QUESTION: 26

The implementation of linear regression in Spark ML first attempts to solve the linear regression problem using matrix decomposition, but this method does not scale well to large datasets with a large number of variables.

Which of the following approaches does Spark ML use to distribute the training of a linear regression model for large data?

A. Logistic regression

B. Singular value decomposition

C. Iterative optimization

D. Least-squares method

Answer: C (LEAVE A REPLY)

For large datasets, Spark ML uses iterative optimization methods to distribute the training of a linear regression model. Specifically, Spark MLlib employs techniques like Stochastic Gradient Descent (SGD) and Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimization to iteratively update the model parameters. These methods are well-suited for distributed computing environments because they can handle large-scale data efficiently by processing mini-batches of data and updating the model incrementally.

Reference:

Databricks documentation on linear regression: Linear Regression in Spark ML

NEW QUESTION: 27

A data scientist has written a feature engineering notebook that utilizes the pandas library. As the size of the data processed by the notebook increases, the notebook's runtime is drastically increasing, but it is processing slowly as the size of the data included in the process increases.

Which of the following tools can the data scientist use to spend the least amount of time refactoring their notebook to scale with big data?

A. PySpark DataFrame API

B. pandas API on Spark

C. Spark SQL

D. Feature Store

Answer: B (LEAVE A REPLY)

The pandas API on Spark provides a way to scale pandas operations to big data while minimizing the need for refactoring existing pandas code. It allows users to run pandas operations on Spark DataFrames, leveraging Spark's distributed computing capabilities to handle large datasets more efficiently. This approach requires minimal changes to the existing code, making it a convenient option for scaling pandas-based feature engineering notebooks.

Reference:

Databricks documentation on pandas API on Spark: pandas API on Spark