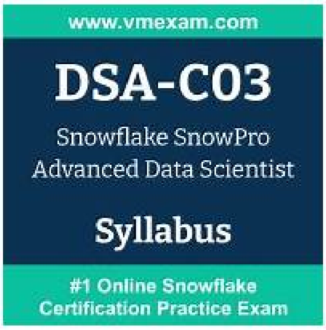
Reliable DSA-C03 Exam Tutorial & DSA-C03 Latest Exam Format



Don't waste much more time on preparing for a test. Hurry to purchase TorrentVCE Snowflake DSA-C03 certification training dumps. With the exam dumps, you will know how to effectively prepare for your exam. This is precious tool that can let you sail through DSA-C03 test with no mistakes. Missing the chance, I am sure you must regret it. Thus, don't hesitate and act quickly.

Our DSA-C03 study materials provide free trial service for consumers. If you are interested in our DSA-C03 study materials, and you can immediately download and experience our trial question bank for free. Through the trial you will have different learning experience on DSA-C03 exam guide , you will find that what we say is not a lie, and you will immediately fall in love with our products. As a key to the success of your life, the benefits that our DSA-C03 Study Materials can bring you are not measured by money. DSA-C03 test torrent can help you pass the exam in the shortest time.

>> Reliable DSA-C03 Exam Tutorial <<

Achieve Success 100% With DSA-C03 Exam Questions In The First Attempt

Overall we can say that SnowPro Advanced: Data Scientist Certification Exam (DSA-C03) certification can provide you with several benefits that can assist you to advance your career and achieve your professional goals. Are you ready to gain all these personal and professional benefits? Looking for a sample, is smart and quick for SnowPro Advanced: Data Scientist Certification Exam (DSA-C03) exam dumps preparation? If your answer is yes then you do not need to go anywhere, just download TorrentVCE DSA-C03 Questions and start SnowPro Advanced: Data Scientist Certification Exam (DSA-C03) exam preparation with complete peace of mind and satisfaction.

Snowflake SnowPro Advanced: Data Scientist Certification Exam Sample Questions (Q273-Q278):

NEW QUESTION # 273
You are building a model training pipeline in Snowflake using Snowpark Python. You want to leverage a pre-trained model from

Hugging Face Transformers for a text classification task, fine-tuning it with your own labeled data stored in a Snowflake table named 'training_data'. You've chosen the 'transformers' library and plan to use a 'transformers.pipeline' for inference. Which of the following code snippets, when integrated into your Snowpark Python application, will correctly download the pre trained model and tokenizer, prepare the data, perform fine-tuning, and then save the fine-tuned model to a Snowflake stage?

```
○ from snowflake.snowpark import Session import transformers def train_model(session: Session, model_name: str, stage_name: str, table_name: str):
model = transformers.AutoModelForSequenceClassification.from_pretrained(model_name) tokenizer =
transformers.AutoTokenizer.from_pretrained(model_name) training_data = session.table(table_name).to_pandas() trainer =
transformers.Trainer(model=model, train_dataset=training_data, tokenizer=tokenizer) trainer.train() model.save_pretrained(f'@{stage_name}')
tokenizer.save_pretrained(f'@{stage_name}')
```

```
○ from snowflake.snowpark import Session from transformers import AutoModelForSequenceClassification, AutoTokenizer, Trainer, TrainingArguments def
train_model(session: Session, model_name: str, stage_name: str, table_name: str): model =
AutoModelForSequenceClassification.from_pretrained(model_name) tokenizer = AutoTokenizer.from_pretrained(model_name) training_data =
session.table(table_name) training_args = TrainingArguments(output_dir='tmp_results', evaluation_strategy='epoch') trainer = Trainer(model=model,
args=training_args, train_dataset=training_data.to_pandas(), tokenizer=tokenizer) trainer.train() trainer.save_model(f'@{stage_name}')
```

```
○ from snowflake.snowpark import Session import torch from transformers import AutoModelForSequenceClassification, AutoTokenizer, Trainer,
TrainingArguments, TextClassificationPipeline def train_model(session: Session, model_name: str, stage_name: str, table_name: str): model =
AutoModelForSequenceClassification.from_pretrained(model_name) tokenizer = AutoTokenizer.from_pretrained(model_name) training_data =
session.table(table_name).to_pandas() training_args = TrainingArguments(output_dir='tmp_results', evaluation_strategy='epoch') trainer =
Trainer(model=model, args=training_args, train_dataset=training_data, tokenizer=tokenizer) trainer.train() trainer.save_model(f'@{stage_name}') #
Create a pipeline for inference and upload it to the stage pipeline = TextClassificationPipeline(model=model, tokenizer=tokenizer,
device=torch.device('cuda' if torch.cuda.is_available() else 'cpu')) session.add_dependency(f'@{stage_name}')
```

```
○ from snowflake.snowpark import Session from transformers import AutoModelForSequenceClassification, AutoTokenizer, Trainer, TrainingArguments def
train_model(session: Session, model_name: str, stage_name: str, table_name: str): model =
AutoModelForSequenceClassification.from_pretrained(model_name) tokenizer = AutoTokenizer.from_pretrained(model_name) training_data =
session.table(table_name).to_pandas() training_args = TrainingArguments(output_dir='/tmp', evaluation_strategy='epoch') trainer =
Trainer(model=model, args=training_args, train_dataset=training_data, tokenizer=tokenizer) trainer.train() trainer.save_model(f'@{stage_name}')
```

```
○ from snowflake.snowpark import Session from transformers import AutoModelForSequenceClassification, AutoTokenizer, Trainer, TrainingArguments def
train_model(session: Session, model_name: str, stage_name: str, table_name: str): model =
AutoModelForSequenceClassification.from_pretrained(model_name) tokenizer = AutoTokenizer.from_pretrained(model_name) training_data =
session.table(table_name).to_pandas() training_args = TrainingArguments(output_dir='./results', evaluation_strategy='epoch') trainer =
Trainer(model=model, args=training_args, train_dataset=training_data, tokenizer=tokenizer) trainer.train() trainer.save_model(f'@{stage_name}')
```

- A. Option E
- B. Option A
- C. Option D
- D. Option C
- E. Option B

Answer: E

Explanation:
The correct answer is B. It correctly uses the 'transformers' library with Snowpark Python. It downloads the model and tokenizer using and 'AutoTokenizer.from_pretrained'. TrainingArguments are configured with output_dir and evaluation_strategy. It reads training data using session.table. Trainer properly configured and finally Trainer saves the trained model in specified 'stage_name' . Option A is incorrect because it missing 'TrainingArgumentS configuration and uses general function, which may not be optimal for the Trainer setup. Option C is incorrect because incorrect use case. Option D and E is incorrect because 'TrainingArguments' output_dir is local folder that cannot be written by Trainer.

NEW QUESTION # 274
A marketing analyst is building a propensity model to predict customer response to a new product launch. The dataset contains a 'City' column with a large number of unique city names. Applying one-hot encoding to this feature would result in a very high-dimensional dataset, potentially leading to the curse of dimensionality. To mitigate this, the analyst decides to combine Label Encoding followed by binarization techniques. Which of the following statements are TRUE regarding the benefits and challenges of this combined approach in Snowflake compared to simply label encoding?

- A. Label encoding followed by binarization will reduce the memory required to store the 'City' feature compared to one-hot encoding, and Snowflake's columnar storage optimizes storage for integer data types used in label encoding.
- B. Binarization following label encoding may enhance model performance if a specific split based on a defined threshold is meaningful for the target variable (e.g., distinguishing between cities above/below a certain average income level related to marketing success).
- C. While label encoding itself adds an ordinal relationship, applying binarization techniques like binary encoding (converting the label to binary representation and splitting into multiple columns) after label encoding will remove the arbitrary ordinal relationship.

- D. Binarizing a label encoded column using a simple threshold (e.g., creating a 'high_city_id' flag) addresses the curse of dimensionality by reducing the number of features to one, but it loses significant information about the individual cities.
- E. Label encoding introduces an arbitrary ordinal relationship between the cities, which may not be appropriate. Binarization alone cannot remove this artifact.

Answer: A,B,D,E

Explanation:
Option A is true because label encoding converts strings into integers, which are more memory-efficient than storing numerous one-hot encoded columns. Snowflake's columnar storage further optimizes integer storage. Option B is also true; label encoding inherently creates an ordinal relationship that might not be valid for nominal features like city names. Option C is incorrect; simple binarization (e.g., > threshold) of label encoded data doesn't remove the arbitrary ordinal relationship; more complex binarization techniques would be needed. Option D is accurate; binarization reduces dimensionality but sacrifices granularity, leading to information loss. Option E is correct because carefully chosen thresholds might correlate with the target variable and improve predictive power.

NEW QUESTION # 275
You are preparing a dataset in Snowflake for a K-means clustering algorithm. The dataset includes features like 'age', 'income' (in USD), and 'number of_transactions'. 'Income' has significantly larger values than 'age' and 'number of_transactions'. To ensure that all features contribute equally to the distance calculations in K-means, which of the following scaling approaches should you consider, and why? Select all that apply:

- A. Apply RobustScaler to handle outliers and then StandardScaler or MinMaxScaler to further scale the features.
- B. Apply PowerTransformer to transform income and StandardScaler to other features to handle skewness.
- C. Do not scale the data, as K-means is robust to differences in feature scales.
- D. Apply StandardScaler to all three features ('age', 'income', 'number_of_transactions') to center the data around zero and scale it to unit variance.
- E. Apply MinMaxScaler to all three features to scale them to a range between O and 1 .

Answer: A,D,E

Explanation:
K-means clustering is sensitive to the scale of the features because it relies on distance calculations. Features with larger values will have a disproportionate influence on the clustering results. StandardScaler centers the data around zero and scales it to unit variance, which ensures that all features have a similar range and variance. MinMaxScaler scales the features to a range between O and 1, which also addresses the issue of different scales. RobustScaler handles outliers which will then use the other two scaling techniques. Therefore A, B and D are the appropriate scaling techniques. C is not correct as K-means relies on distance calculations and not scaling the data could give some feature a larger weight which isn't the desired outcome. Option E: Using PowerTransformer on 'income' to reduce skewness and StandardScaler on the other features can be a valid approach, but it depends on the distribution of 'income' and the presence of outliers. If 'income' is highly skewed and/or contains outliers, this combination might be more effective than using StandardScaler or MinMaxScaler alone.

NEW QUESTION # 276
You have successfully trained a binary classification model using Snowpark ML and deployed it as a UDF in Snowflake. The UDF takes several input features and returns the predicted probability of the positive class. You need to continuously monitor the model's performance in production to detect potential data drift or concept drift. Which of the following methods and metrics, when used together, would provide the MOST comprehensive and reliable assessment of model performance and drift in a production environment? (Select TWO)

- A. Calculate the Kolmogorov-Smirnov (KS) statistic between the distribution of predicted probabilities in the training data and the production data over regular intervals. Track any substantial changes in the KS statistic.
- B. Check for null values in the input features passed to the UDF. A sudden increase in null values indicates a problem with data quality.
- C. Monitor the volume of data processed by the UDF per day. A sudden drop in volume indicates a problem with the data pipeline.
- D. Continuously calculate and track performance metrics like AUC, precision, recall, and Fl-score on a representative sample of labeled production data over regular intervals. Compare these metrics to the model's performance on the holdout set during training.
- E. Monitor the average predicted probability score over time. A significant shift in the average score indicates data drift.

Answer: A,D

Explanation:
Options B and D provide the most comprehensive assessment of model performance and drift. Option D, by continuously calculating key performance metrics (AUC, precision, recall, F1 -score) on labeled production data, directly assesses how well the model is performing on real- world data. Comparing these metrics to the holdout set provides insights into potential overfitting or degradation over time (concept drift). Option B, calculating the KS statistic between the predicted probability distributions of training and production data, helps to identify data drift, indicating that the input data distribution has changed. Option A can be an indicator but is less reliable than the KS statistic. Option C monitors data pipeline health, not model performance. Option E focuses on data quality, which is important but doesn't directly assess model performance drift.

NEW QUESTION # 277
You are working with a Snowflake table named 'CUSTOMER DATA' containing customer information, including a 'PHONE NUMBER' column. Due to data entry errors, some phone numbers are stored as NULL, while others are present but in various inconsistent formats (e.g., with or without hyphens, parentheses, or country codes). You want to standardize the 'PHONE NUMBER column and replace missing values using Snowpark for Python. You have already created a Snowpark DataFrame called 'customer df representing the 'CUSTOMER DATA' table. Which of the following approaches, used in combination, would be MOST efficient and reliable for both cleaning the existing data and handling future data ingestion, given the need for scalability?

- A. Create a Snowflake Pipe with a COPY INTO statement and a transformation that uses a SQL function within the COPY INTO statement to format the phone numbers and replace NULL values during data loading. Also, implement a Python UDF for correcting already existing data.
- B. Leverage Snowflake's data masking policies to mask any invalid phone number and create a view that replaces NULL values with 'UNKNOWN'. This approach doesn't correct existing data but hides the issue.
- C. Create a Snowflake Stored Procedure in SQL that uses regular expressions and 'CASE statements to format the "PHONE_NUMBER column and replace NULL values. Call this stored procedure from a Snowpark Python script.
- D. Use a UDF (User-Defined Function) written in Python that formats the phone numbers based on a regular expression and applies it to the DataFrame using For NULL values, replace them with a default value of 'UNKNOWN'.
- E. Use a series of and methods on the Snowpark DataFrame to handle NULL values and different phone number formats directly within the DataFrame operations.

Answer: A,D

Explanation:
Options A and E provide the most robust and scalable solutions. A UDF offers flexibility and reusability for data cleaning within Snowpark (Option A). Option E leverages Snowflake's data loading capabilities to clean data during ingestion and adds a UDF for cleaning existing data providing a comprehensive approach. Using a UDF written in Python and used within Snowpark leverages the power of Python's regular expression capabilities and the distributed processing of Snowpark. Handling data transformations during ingestion with Snowflake's built- in COPY INTO with transformation is highly efficient. Option B is less scalable and maintainable for complex formatting. Option C is viable but executing SQL stored procedures from Snowpark Python loses some of the advantages of Snowpark. Option D addresses data masking not data transformation.

NEW QUESTION # 278
......

It is through these interfaces that applications have access DSA-C03 Latest Exam Format to the functionality of object persistence, With a client/server application, the client application can perform sophisticated operations without depending on the server, whereas New DSA-C03 Exam Testking a thin client application tends to be able to display only the results of processing that happened on a server.

Efficient DSA-C03 – 100% Free Reliable Exam Tutorial | DSA-C03 Latest Exam Format

Firstly,we promise all candidates can pass exam if they master all questions and answers of Snowflake DSA-C03 Dumps PDF materials, Once you purchase, you will receive our valid DSA-C03 vce via e-mail immediately and practice it promptly.

So many IT candidates are clear in their mind that getting DSA-C03 SnowPro Advanced: Data Scientist Certification Exam certification can add a bright point to the resume and have access to rich rewards and benefits.

So rest assured that you will get top-notch and easy-to-use Snowflake DSA-C03 practice questions, It not only improves the possibility of your life but also keep you constant learning.