Snowflake
DSA-C02
SnowPro Advanced: Data Scientist Certification Exam

For More Information – Visit link below:
https://www.examsempire.com/

Product Version

1. **Up to Date products, reliable and verified.**
2. **Questions and Answers in PDF Format.**

Latest Version: 6.0

Question: 1

A data scientist is tasked with predicting customer churn for a telecommunications company using Snowflake. The dataset contains a mix of categorical and numerical features, including customer demographics, service usage, and billing information. The target variable is 'churned' (binary: 0 or 1). Which of the following steps are crucial to address potential issues and ensure optimal performance of a supervised learning model (e.g., Logistic Regression or Gradient Boosted Trees) deployed within Snowflake using Snowpark and external functions?

A. Applying one-hot encoding to categorical features before training the model, and ensuring the same encoding is applied during inference via a Snowflake UDE
B. Scaling numerical features using StandardScaler or MinMaxScaler within a Snowpark DataFrame, and saving the scaler parameters to apply consistently during inference using Snowflake UDF.
C. Ignoring missing values in the dataset, as Snowflake automatically handles them during model training.
D. Training the model locally using all available data and then deploying the serialized model to Snowflake as an external function without any pre-processing pipeline.
E. Splitting the data into training and validation sets using 'SNOWFLAKE.ML.RANDOM SPLIT to ensure reliable model evaluation.

Answer: A,B,E

Explanation:
Options A, B, and E are essential for robust supervised learning. One-hot encoding (A) converts categorical data into a numerical format suitable for many algorithms. Feature scaling (B) is crucial for algorithms sensitive to feature ranges, like Logistic Regression and those using gradient descent. Ignoring missing values (C) is generally detrimental. Deploying a model without preprocessing (D) will lead to incorrect predictions during inference. Splitting the data into training/validation set using 'SNOWFLAKE.ML.RANDOM_SPLIT (E) is essential for fair evaluation of model performance on unseen data.

Question: 2

You are building a supervised learning model in Snowflake to predict the likelihood of a customer clicking on an advertisement. You are using a logistic regression model and Snowflake's native machine learning functions. You have noticed the model is performing poorly on new data'. Which of the following approaches would be MOST effective to diagnose and address potential issues with the model's performance and maintainability, considering the dataset has high cardinality features and a large number of records?

A. Increase the regularization strength (e.g., L1 or L2 penalty) and perform cross-validation using 'SNOWFLAKE.ML.CORE.HOLDOUT to evaluate performance on multiple validation sets.

B. Retrain the model more frequently using the entire dataset, without any regularization, to capture the latest trends in the data. This removes any potential bias introduced by regularization.
C. Reduce the number of features used in the model by randomly selecting a subset of the original features. This will simplify the model and prevent overfitting.
D. Employ feature engineering to create interaction terms between high-cardinality features and other relevant features, using 'SNOWFLAKE.ML.FEATURE TRANSFORM' and then assessing their importance using feature importance analysis.
E. Implement a monitoring dashboard to track model performance metrics (e.g., AUC, precision, recall) over time and set up alerts for significant drops in performance or data drift.

Answer: A,D,E

Explanation:
Options A, D, and E are effective strategies. Increasing regularization strength (A) helps prevent overfitting, particularly with high- dimensional data. Cross-validation gives a more reliable estimate of model performance. Feature engineering and importance analysis (D) can improve the model's ability to capture complex relationships in the data. Monitoring (E) is crucial for detecting and addressing performance degradation over time. Retraining without regularization (B) is likely to worsen overfitting. Randomly selecting features (C) might remove important information. Cross validation should be used to choose the optimal model using SNOWFLAKE.ML.CORE.HOLDOUT or some of the other ML provided ways of Cross-Validation.

Question: 3

A financial institution wants to predict fraudulent transactions using a supervised learning model deployed in Snowflake. The dataset is highly imbalanced, with only 1% of transactions being fraudulent. You are using a gradient boosting model. Which of the following techniques are most appropriate to handle the imbalanced dataset and improve the model's ability to detect fraud, and how would you implement them within the Snowflake environment?

A. Use a cost-sensitive learning approach by assigning higher weights to the minority class (fraudulent transactions) during model training. This can be achieved by setting the 'class_weight' parameter in the gradient boosting algorithm (if available via an external function) or by manually adjusting the loss function.
B. Oversample the minority class by duplicating fraudulent transactions in the training set. This can be done using a Snowpark DataFrame to resample the dataset before training. = df.filter(col('is_fraud') 1); df_resampled = df.unionAll(df_fraud.sample(fraction=10,
C. Undersample the majority class by randomly removing non-fraudulent transactions. This will always improve performance as it balances the dataset.
D. Use evaluation metrics that are sensitive to imbalanced datasets, such as AUC-ROC, precision, recall, and Fl-score, instead of relying solely on accuracy. Calculate these metrics using Snowflake SQL after predictions are generated.
E. Employ a SMOTE (Synthetic Minority Oversampling Technique) or ADASYN algorithm to generate synthetic samples of the minority class. This would likely require using an external function or Snowpark to apply the algorithm within the Snowflake environment.

Answer: A,D,E

Explanation:
Options A, D, and E are best practices for handling imbalanced datasets. Cost-sensitive learning (A) penalizes misclassification of the minority class more heavily. Appropriate evaluation metrics (D) provide a more accurate picture of performance than accuracy. SMOTE/ADASYN (E) generate synthetic samples to balance the dataset without simply duplicating existing samples. Simply undersampling the majority class (C) might lead to important information being discarded which will result in decreased model performance.

Question: 4

A marketing analyst at a retail company is using Snowflake to perform customer segmentation using unsupervised learning. They have a table 'CUSTOMER TRANSACTIONS with columns: 'CUSTOMER ICY, 'TOTAL SPENT, 'AVG ORDER VALUE 'NUM_TRANSACTIONS, and 'LAST PURCHASE DATE. They want to use k-means clustering to identify distinct customer segments based on their spending behavior. The analyst wants to scale 'TOTAL SPENT' and 'AVG_ORDER VALUE' to the range [0, 1] before clustering. Which of the following SQL statements, leveraging Snowflake's capabilities and unsupervised learning best performs this task and stores results in table 'CUSTOMER SEGMENTS'?

○ ```sql CREATE OR REPLACE TABLE CUSTOMER_SEGMENTS AS SELECT CUSTOMER_ID, CLUSTER_ID(k, ARRAY_CONSTRUCT(TOTAL_SPENT, AVG_ORDER_VALUE)) AS SEGMENT FROM CUSTOMER_TRANSACTIONS; ``` (Requires preprocessing of scaling outside the query)

○ ```sql CREATE OR REPLACE TABLE CUSTOMER_SEGMENTS AS SELECT CUSTOMER_ID, CLUSTER_ID(3, ARRAY_CONSTRUCT((TOTAL_SPENT - MIN(TOTAL_SPENT) OVER ()) / (MAX(TOTAL_SPENT) OVER () - MIN(TOTAL_SPENT) OVER ()), (AVG_ORDER_VALUE - MIN(AVG_ORDER_VALUE) OVER ()) / (MAX(AVG_ORDER_VALUE) OVER () - MIN(AVG_ORDER_VALUE) OVER ()))) AS SEGMENT FROM CUSTOMER_TRANSACTIONS; ``` (Assuming k=3 and Min-Max scaling within the query)

○ ```sql CREATE OR REPLACE TABLE CUSTOMER_SEGMENTS AS SELECT CUSTOMER_ID, CLUSTER_ID(3, TOTAL_SPENT, AVG_ORDER_VALUE) AS SEGMENT FROM CUSTOMER_TRANSACTIONS; ``` (Assuming k=3, but no scaling)

○ ```sql CREATE OR REPLACE TABLE CUSTOMER_SEGMENTS AS SELECT CUSTOMER_ID, CLUSTER_ID(k, ARRAY_CONSTRUCT(STDDEV(TOTAL_SPENT) OVER (), STDDEV(AVG_ORDER_VALUE) OVER ())) AS SEGMENT FROM CUSTOMER_TRANSACTIONS; ``` (Uses standard deviation, not min-max scaling)

○ ```sql CREATE OR REPLACE TABLE CUSTOMER_SEGMENTS AS SELECT CUSTOMER_ID, CLUSTER_ID(3, ARRAY_CONSTRUCT((TOTAL_SPENT - MIN(TOTAL_SPENT)) / (MAX(TOTAL_SPENT) - MIN(TOTAL_SPENT)), (AVG_ORDER_VALUE - MIN(AVG_ORDER_VALUE)) / (MAX(AVG_ORDER_VALUE) - MIN(AVG_ORDER_VALUE)))) AS SEGMENT FROM CUSTOMER_TRANSACTIONS; ``` (Incorrect use of aggregate function)

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

Answer: B

Explanation:
Option B correctly implements k-means clustering using and performs Min-Max scaling within the query using window functions to calculate the minimum and maximum values for 'TOTAL_SPENT and This scales the features to a range between 0 and 1 before clustering, preventing features with larger magnitudes from dominating the clustering process. Option A requires preprocessing which isnt self contained. Options CID and E are syntactically incorrect and does not implement Min-Max Scaling.

Question: 5

A fraud detection team at a financial institution is using Snowflake to analyze transaction data'. They have a table 'TRANSACTIONS containing features like 'TRANSACTION_AMOUNT', 'MERCHANT CATEGORY, , and 'TIMESTAMP. They suspect that fraudulent transactions may form distinct clusters based on these features. Which of the following unsupervised learning techniques, available within or integratable with Snowflake, would be MOST appropriate for identifying potentially fraudulent clusters, and why?

A. K-Means Clustering, because it partitions data into K distinct clusters based on minimizing the within-cluster variance.
B. Principal Component Analysis (PCA), because it reduces the dimensionality of the data and helps visualize the most important features.
C. Isolation Forest, because it identifies anomalies by isolating them in a decision tree structure.
D. Hierarchical Clustering, because it builds a hierarchy of clusters, allowing for exploration at different levels of granularity.
E. Linear Regression, because it predicts a continuous target variable based on input features.

Answer: C

Explanation:
Isolation Forest is the most appropriate technique. While k-means (A) and hierarchical clustering (D) can identify clusters, Isolation Forest is specifically designed to identify anomalies. Fraudulent transactions are, by definition, anomalies. PCA (B) is useful for dimensionality reduction but doesn't directly identify clusters or anomalies. Linear Regression (E) is a supervised learning technique and is not applicable here. Isolation Forest would identify transactions that are easily isolated from the rest of the data, indicating potential fraud.

Question: 6

A data scientist is tasked with understanding customer behavior on an e-commerce platform using Snowflake. They have access to a large table named 'WEBSITE SESSIONS with columns such as 'USER ID', 'PAGE_VISITED, 'TIME_SPENT, and 'ACTIONS TAKEN'. They aim to discover common user navigation patterns (e.g., sequences of pages visited) without predefined labels. Which of the following unsupervised learning techniques, or combinations thereof, could be effectively used to achieve this goal?

A. K-Means Clustering on features derived from page visits, followed by Association Rule Mining to identify frequent page sequences within each cluster.
B. Principal Component Analysis (PCA) to reduce the dimensionality of page visit data, followed by K-Means Clustering to group similar sessions.
C. Time Series Analysis to predict future page visits based on past session data.
D. Hierarchical Clustering directly on the raw page visit data to form a hierarchy of user sessions.
E. Association Rule Mining directly on the raw page visit data to uncover common sequences, and then Clustering (K-Means or Hierarchical) based on the resulting rule strengths and confidences.

Answer: A,E

Explanation:
Options A and E are both viable approaches. Option A leverages K-Means to segment users based on aggregated features of their sessions and then uses Association Rule Mining within each segment to discover page sequences specific to those segments, leading to more refined insights. Option E directly uses Association Rule Mining to find frequent sequences across all sessions, and then uses Clustering to find groups of rules and therefore patterns. Options B and D are less effective, as PCA reduces dimensionality but does not reveal sequences, and raw page visit data without transformation may lead to poor clustering results. Option C is a predictive time series analysis technique and is not primarily for discovering patterns.

Question: 7

A data science team is using Snowflake to store historical sales data, including and promotion_spend'. They want to predict future sales based on these features using linear regression. However, they suspect and have a non-linear relationship. Which of the following strategies would be MOST effective in addressing this non-linearity within Snowflake without exporting data to an external platform?

A. Apply a logarithmic transformation to the ‘unit_price’ column within Snowflake using before using it in the linear regression model.
B. Create a new feature in Snowflake that is the square of ‘unit_price’ (i.e., ‘unit_price and include both and its squared term in the linear regression model.
C. Use Snowflake's built-in feature store capabilities to engineer a custom feature that quantizes into discrete price tiers (e.g., low, medium, high) and use those tiers as categorical variables in the linear regression.
D. Fit separate linear regression models for different ranges of 'unit_price’. This involves segmenting the data based on price bands and training a unique model for each segment directly in Snowflake.
E. Export the data to a Python environment, perform polynomial regression using scikit-learn, and then import the model's coefficients back into Snowflake for prediction.

Answer: B,D

Explanation:
Options B and D are most effective. Option B addresses non-linearity by introducing polynomial features. Option D handles non- linearity by creating piecewise linear models. Option A might help, but squaring is often a more robust approach. Option C changes it to categorization, which might not capture all variation. Option E is undesirable because the question asks to solve it within Snowflake.

Question: 8

You're tasked with classifying customer reviews (structured data) as either 'positive' or 'negative' using Snowflake. You have a table 'CUSTOMER REVIEWS' with columns ‘REVIEW TEXT (VARCHAR) and 'LABEL' (BOOLEAN, TRUE for positive, FALSE for negative). You've preprocessed the text data and created features using TF-IDF. Now, you need to train a Logistic Regression model directly within Snowflake using Snowpark. Which of the following Python code snippets, when executed within a Snowpark session, correctly trains a Logistic Regression model and evaluates its performance using a train/test split, assuming 'features' and 'labels' are already defined Snowpark DataFrames?

A.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(features.to_pandas(), labels.to_pandas(), test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

B.

```
import snowflake.ml.modeling
from snowflake.ml.modeling import linear_model
from snowflake.ml.model_selection import train_test_split
from snowflake.ml.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
model = linear_model.LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

C.

```
import snowflake.ml.modeling
from snowflake.ml.modeling import linear_model
from sklearn.model_selection import train_test_split #Incorrect import, use snowflake implementation
from sklearn.metrics import accuracy_score #Incorrect import, use snowflake implementation

X_train, X_test, y_train, y_test = train_test_split(features.to_pandas(), labels.to_pandas(), test_size=0.2, random_state=42)
model = linear_model.LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

D.

```
from snowflake.ml.modeling import linear_model
from snowflake.ml.model_selection import train_test_split
from snowflake.ml.modeling.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
model = linear_model.LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

E.

```
import snowflake.ml.modeling.linear_model as sn_linear_model
import snowflake.ml.model_selection as sn_model_selection
import snowflake.ml.modeling.metrics as sn_metrics

X_train, X_test, y_train, y_test = sn_model_selection.train_test_split(features, labels, test_size=0.2, random_state=42)
model = sn_linear_model.LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = sn_metrics.accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

Answer: B,E

Explanation:
Option A uses scikit-learn which is generally not meant to be used directly inside a Snowpark Stored Procedure, as data transfer to pandas is required (defeating the point of Snowpark). Option B and E correctly utilize Snowflake's ML modeling and evaluation framework for in- database machine learning, avoiding data egress. Option C attempts to mix standard python libraries with snowpark, creating problems with data types and library availability. Option D contains incorrect import path

Question: 9

You're building a fraud detection system for credit card transactions using Snowflake. You have structured data including transaction amount, merchant category, time of day, and customer demographics. The target variable is a binary flag indicating fraudulent or legitimate transactions. You train a Logistic Regression model and obtain a high overall accuracy (e.g., 99.5%). However, when you deploy the model, you observe that it consistently fails to identify actual fraudulent transactions, leading to significant financial losses.
Which of the following strategies would be most effective in addressing this issue?

A. Increase the threshold for classifying a transaction as fraudulent, aiming to reduce the number of false positives.
B. Collect more data and retrain the model, ensuring the training dataset is larger and more representative of real-world transaction patterns.
C. Apply techniques to handle class imbalance, such as oversampling the fraudulent transactions or undersampling the legitimate transactions, before retraining the model.
D. Switch to a more complex model, such as a deep neural network, to capture intricate patterns in the data.
E. Focus on feature engineering, creating new features that better capture the characteristics of fraudulent transactions, such as transaction frequency and amount deviation from the customer's usual spending habits.

Answer: C,E

Explanation:
High accuracy with poor fraud detection suggests a class imbalance problem (few fraudulent transactions). Oversampling/undersampling addresses this by providing a more balanced representation

to the model. Feature engineering is also crucial to better represent distinguishing characteristics of fraudulent transactions.

Question: 10

A marketing team wants to classify customer reviews into five sentiment categories: 'Very Negative', 'Negative', 'Neutral', 'Positive', and 'Very Positive'. They have a Snowflake table named 'CUSTOMER REVIEWS' with columns 'REVIEW TEXT (string) and 'SENTIMENT CATEGORY' (string, the target variable). They intend to use Snowpark ML Modeling for this multi-class classification task. Which of the following code snippets BEST demonstrates the end-to-end process of data preparation, model training, and prediction using a pipeline with a text vectorizer (e.g., TF-IDF) and a RandomForestClassifier?

A.

```
from snowflake.snowpark.context import get_active_session
from snowflake.snowpark.ml.modeling import RandomForestClassifier
from snowflake.snowpark.ml.feature_extraction import TfidfVectorizer
from snowflake.snowpark.ml.modeling import Pipeline

session = get_active_session()

df = session.table('CUSTOMER_REVIEWS')

tfidf = TfidfVectorizer(input_col='REVIEW_TEXT', output_col='features')
rfc = RandomForestClassifier(label_col='SENTIMENT_CATEGORY', features_col='features')

pipeline = Pipeline(stages=[tfidf, rfc])
model = pipeline.fit(df)

predictions = model.transform(df)
predictions.select('REVIEW_TEXT', 'SENTIMENT_CATEGORY', 'prediction').show()
```

B.

```
from snowflake.snowpark.context import get_active_session
from snowflake.snowpark.ml.modeling import RandomForestClassifier

session = get_active_session()
df = session.table('CUSTOMER_REVIEWS')

rfc = RandomForestClassifier(label_col='SENTIMENT_CATEGORY', features_col='REVIEW_TEXT')
model = rfc.fit(df)

predictions = model.transform(df)
predictions.select('REVIEW_TEXT', 'SENTIMENT_CATEGORY', 'prediction').show()
```

C.

```
from snowflake.snowpark.context import get_active_session
from snowflake.snowpark.ml.modeling import RandomForestClassifier
from snowflake.snowpark.ml.feature_extraction import HashingTF

session = get_active_session()
df = session.table('CUSTOMER_REVIEWS')

htf = HashingTF(input_col='REVIEW_TEXT', output_col='features', numFeatures=1000)
rfc = RandomForestClassifier(label_col='SENTIMENT_CATEGORY', features_col='features')

model = rfc.fit(df)

predictions = model.transform(df)
predictions.select('REVIEW_TEXT', 'SENTIMENT_CATEGORY', 'prediction').show()
```

D.

```
from snowflake.snowpark.context import get_active_session
from snowflake.snowpark.ml.modeling import RandomForestClassifier
from snowflake.snowpark.ml.feature_extraction import TfidfVectorizer

session = get_active_session()
df = session.table('CUSTOMER_REVIEWS')

tfidf = TfidfVectorizer(input_col='REVIEW_TEXT', output_col='features')

rfc = RandomForestClassifier(label_col='SENTIMENT_CATEGORY', features_col='features')
model = rfc.fit(tfidf.transform(df))

predictions = model.transform(tfidf.transform(df))
predictions.select('REVIEW_TEXT', 'SENTIMENT_CATEGORY', 'prediction').show()
```

E.

```
from snowflake.snowpark.context import get_active_session
from snowflake.snowpark.ml.modeling import LogisticRegression
from snowflake.snowpark.ml.feature_extraction import TfidfVectorizer
from snowflake.snowpark.ml.modeling import Pipeline

session = get_active_session()

df = session.table('CUSTOMER_REVIEWS')

tfidf = TfidfVectorizer(input_col='REVIEW_TEXT', output_col='features')
lr = LogisticRegression(label_col='SENTIMENT_CATEGORY', features_col='features')

pipeline = Pipeline(stages=[tfidf, lr])
model = pipeline.fit(df)

predictions = model.transform(df)
predictions.select('REVIEW_TEXT', 'SENTIMENT_CATEGORY', 'prediction').show()
```

Answer: A

Explanation:
Option A correctly demonstrates the end-to-end process. It uses 'TfidfVectorizer' to convert the text data into numerical features, and a 'RandomForestClassifier' for multi-class classification. It also properly uses 'Pipeline' to combine the feature extraction and model training steps. Option B misses feature extraction, directly using text as features. Option C uses "HashingTF but doesn't use a pipeline. Option D applies the transformations multiple times outside of pipeline. Option E uses LogisticRegression classifier in this scenario, we need to use RandomForestClassifer.

Thank You for Trying Our Product

Special 16 USD Discount Coupon: NSZUBG3X

Email: support@examsempire.com

Check our Customer Testimonials and ratings available on every product page.

Visit our website.

https://examsempire.com/