20 march

Q1. What is data encoding? How is it useful in data science?

Data encoding refers to the process of converting categorical data (textual labels or non-numeric values) into a numerical format that can be used as input for machine learning algorithms. It is essential in data science because most machine learning algorithms require numerical data for training. Data encoding ensures that categorical information is appropriately represented in a way that the algorithms can understand and process.

Q2. What is nominal encoding? Provide an example of how you would use it in a real-world scenario?

Nominal encoding assigns a unique integer to each category in a categorical feature without any inherent order or ranking. For example, if you have a "Color" feature with categories "Red," "Blue," "Green," and "Yellow," nominal encoding would assign integer values like 1, 2, 3, and 4 to these categories.

Example: In a real-world scenario, suppose you're working with a dataset about cars, and one of the features is "Car Brand" with categories like "Toyota," "Ford," "Honda," and "Chevrolet." You can use nominal encoding to convert these categories into numerical values, making it suitable for machine learning algorithms.

Q3. In what situations is nominal encoding preferred over one-hot encoding? Provide a practical example?

Nominal encoding is preferred over one-hot encoding when the categorical feature has a high cardinality (a large number of unique categories) and including one-hot encoded columns would lead to high dimensionality. For example, if you're working with a feature like "Country," which has hundreds of unique values, nominal encoding could be more practical.

Example: Consider a dataset containing customer information, including their "Country." One-hot encoding for the "Country" feature would create numerous additional columns, which might lead to a large number of features and potential performance issues. In such cases, nominal encoding can help manage dimensionality while still providing meaningful representation.

Q4. Suppose you have a dataset containing categorical data with 5 unique values. Which encoding technique would you use to transform this data into a format suitable for machine learning algorithms? Explain why you made this choice.?

For a categorical feature with 5 unique values, you could use nominal encoding. Since the number of unique values is relatively small, nominal encoding assigns a unique integer to each category, making it efficient and straightforward for machine learning algorithms to process.

Q5. In a machine learning project, you have a dataset with 1000 rows and 5 columns. Two of the columns are categorical, and the remaining three columns are numerical. If you were to use nominal encoding to transform the categorical data, how many new columns would be created? Show your calculations?

If you use nominal encoding for two categorical columns, you would replace each categorical value with a unique integer. This results in 2 new columns (one for each categorical feature), each containing 1000 integer values. So, you would create 2 new columns.

Q6. You are working with a dataset containing information about different types of animals, including their species, habitat, and diet. Which encoding technique would you use to transform the categorical data into a format suitable for machine learning algorithms? Justify your answer?

For this scenario, where there are categories like "species," "habitat," and "diet," you should use one-hot encoding. One-hot encoding creates binary columns for each category, effectively representing the presence or absence of each category. This technique is ideal when the categories have no inherent order and shouldn't be numerically ranked.

Q7. You are working on a project that involves predicting customer churn for a telecommunications company. You have a dataset with 5 features, including the customer's gender, age, contract type, monthly charges, and tenure. Which encoding technique(s) would you use to transform the categorical data into numerical data? Provide a step-by-step explanation of how you would implement the encoding?

In this scenario, you can use **label encoding** for the "gender" feature (assuming it has two categories, like "Male" and "Female"). For the "contract type" feature, you can use **ordinal encoding** since there might be an inherent order like "Month-to-Month," "One year," and "Two year."

Here's how you would implement it:
1. **Label Encoding for Gender:** Assign labels "0" and "1" to "Female" and "Male," respectively.
   - Female -> 0
   - Male -> 1

2. **Ordinal Encoding for Contract Type:** Assign numerical values based on the inherent order.
   - Month-to-Month -> 0
   - One year -> 1
   - Two year -> 2

The other numerical features ("age," "monthly charges," and "tenure") don't need encoding since they're already in a numerical format.

Remember that for predictive modelling, ensure that the encoding aligns with the underlying meaning of the data to avoid introducing unintended relationships.