

Feature Engineering Practice questions:

❖ Missing Value:

Easy Level:

1. Detect Missing Values

You are given a Pandas DataFrame df.

Task:

- Count the number of missing values in each column.
- Print only the columns that contain at least one missing value.

Concept tested:

isnull(), sum(), basic missing value detection

2. Drop Rows with Missing Values

Given a DataFrame df:

Task:

- Remove all rows that contain any missing value.
- Print the shape of the DataFrame before and after deletion.

Concept tested:

Row-wise deletion, MCAR assumption

Medium Level:

3. Mean & Median Imputation

You are given a DataFrame with a numerical column age containing missing values.

Task:

- Create two new DataFrames:
 - One where missing age value are filled using mean
 - One where missing age value are filled using median
- Compare the mean and variance of age in both DataFrames.

Concept tested:

Mean vs median imputation, effect on variance

4. Mode Imputation for Categorical Data

Given a categorical column embarked with missing values:

Task:

- Find the mode of the column.
- Replace missing values with the mode.
- Print the frequency count before and after imputation.

Concept tested:

Mode imputation, categorical data handling

Hard Level:

5. Conditional Imputation (MAR Case)

You are given a DataFrame with columns:

- age (numeric, has missing values)
- gender (categorical, no missing values)

Task:

- Fill missing age values using the mean age within each gender group.
- Do not use a global mean.

Concept tested:

MAR assumption, group-based imputation

6. Identify Missing Data Mechanism

Given a dataset with columns:

- income (has missing values)
- education
- age

Task:

- Write code to check whether missing income values depend on education or age.
- Print a short conclusion (MCAR or MAR) based on observed patterns.

Concept tested:

Missing data mechanism reasoning using observed variables

Advanced / Interview Level

7. Compare Deletion vs Imputation

Given a dataset with missing values in multiple columns:

Task:

- Train the same ML model twice:
 1. After dropping rows with missing values
 2. After imputing missing values
- Compare model accuracy and dataset size.

Concept tested:

Bias–variance tradeoff, impact of missing data handling

8. Custom Imputer Function

Write a Python function:

```
def impute_data(df, strategy):  
    """  
    strategy can be 'mean', 'median', or 'mode'  
    """
```

Task:

- Automatically apply the correct imputation based on column datatype.
- Return the imputed DataFrame.

Concept tested:

Practical imputation logic, reusable code design

❖ Handling Imbalanced Dataset

Question 1 :

Create an imbalanced dataset with:

- 200 samples
- 2 features
- 80% class 0 and 20% class 1

Print the class distribution.

Question 2 (Upsampling)

Using the dataset from Question 1:

- Separate majority and minority classes
 - Perform **up-sampling** on the minority class
 - Make both classes equal
 - Print the new class distribution
-

Question 3 (Downsampling)

Using the same dataset:

- Perform **down-sampling** on the majority class
 - Make both classes equal
 - Print the class distribution after downsampling
-

Question 4 (Concept + Code)

Write Python code to:

1. Check if the dataset is imbalanced
2. If yes, apply **Upsampling**
3. Print "Dataset Balanced" after processing

(Hint: use `value_counts()`)

Question 5 (MCQ + Coding)

Which parameter in `resample()` allows duplication of samples?

- A. `random_state`
- B. `n_samples`
- C. `replace`
- D. `shuffle`

Then write a small code snippet using that parameter.

Question 6 (Real-World Style)

You are working on a **fraud detection dataset** where:

- 0 → Not Fraud (95%)
- 1 → Fraud (5%)

Write code to:

- Balance the dataset using **downsampling**
 - Explain in 1 line why downsampling may be risky here
-

Question 7 (Interview-Style Coding)

Write a function:

```
def balance_data(df, target_col):  
    """  
    Balances the dataset using upsampling  
    """
```

Function should:

- Take a DataFrame and target column name
 - Return a **balanced DataFrame**
-

Bonus Challenge

Modify Question 7 to:

- Automatically choose **upsampling if dataset is small**
- Choose **downsampling if dataset is large**

One-Hot Encoding:

1.

A dataset contains a column named *Color* with the values Red, Blue, Green, Red, and Blue. Write Python code to apply One-Hot Encoding to this column and combine the encoded columns with the original dataset.

2.

A dataset has two columns: *City* (Delhi, Mumbai, Chennai, Delhi) and *Sales* (100, 200, 150, 180). Write Python code to apply One-Hot Encoding to the *City* column while keeping the *Sales* column unchanged.

3.

A dataset contains a column named *Department* with the values HR, IT, Finance, IT, and HR. Write Python code to perform One-Hot Encoding on this column using the *drop first* option to avoid the dummy variable trap.

4.

A dataset contains two categorical columns: *Gender* (Male, Female, Male, Female) and *City* (Delhi, Mumbai, Delhi, Chennai). Write Python code to apply One-Hot Encoding to both columns simultaneously.

5.

You are given a training dataset with a column *Product* containing Laptop, Mobile, and Tablet. A test dataset contains Laptop and Camera. Write Python code to apply One-Hot Encoding such that the encoder can handle unknown categories present in the test dataset.

Label Encoding:

6.

A dataset contains a column named *Status* with values Placed, Not Placed, Placed, and Placed. Write Python code to convert this column into numerical format using Label Encoding.

7.

A dataset contains two columns: *Marks* and *Result*, where *Result* has values Pass and Fail. Write Python code to apply Label Encoding only to the *Result* column.

8.

A dataset contains two categorical columns named *City* and *Department*. Write Python code to apply Label Encoding separately to both columns.

9.

A dataset contains a column *Gender* with values Male and Female. Write Python code to apply Label Encoding and store the encoder so it can be reused later for new data.

Ordinal Encoding:

11.

A dataset contains a column *Size* with values Small, Medium, Large, and Medium. Write Python code to apply Ordinal Encoding using the order Small < Medium < Large.

12.

A dataset contains a column *Education Level* with values High School, Bachelor, Master, and PhD. Write Python code to apply Ordinal Encoding while preserving the correct educational hierarchy.

13.

A dataset contains a column *Rating* with values Poor, Average, Good, and Excellent. Write Python code to apply Ordinal Encoding based on the natural order of these categories.

14.

A dataset contains two ordered categorical columns: *Size* (Small, Medium, Large) and *Quality* (Low, Medium, High). Write Python code to apply Ordinal Encoding to both columns using their respective category orders.

Target guide ordinal encoding:-

1. Given a categorical column city and target column price, write code to apply target guided ordinal encoding using the mean.
-