Q1. Missing Values

Definition:
Missing values occur when **data for certain features are not recorded** or are absent in a dataset.

Why handle them?

- Algorithms may fail or give **biased results**
- Reduces model accuracy
- Prevents incorrect statistical inferences

Algorithms not affected by missing values:

- **Decision Trees** (can handle missing features)
- **Random Forests** (some implementations)
- **XGBoost / LightGBM** (built-in missing value handling)

Q2. Techniques to Handle Missing Data

1. Removing Missing Values

- Remove rows or columns with missing data.

```
import pandas as pd
df = pd.DataFrame({'A':[1,2,None,4],'B':[5,None,7,8]})
df.dropna()  # drops rows with any missing value
```

---

2. Imputation with Mean/Median/Mode

- Replace missing values with **statistical measures**.

```
df['A'].fillna(df['A'].mean(), inplace=True)  # mean imputation
df['B'].fillna(df['B'].median(), inplace=True)  # median imputation
```

---

3. Imputation with Forward/Backward Fill

```
df.fillna(method='ffill', inplace=True)  # forward fill
df.fillna(method='bfill', inplace=True)  # backward fill
```

---

4. Predictive Imputation

- Use ML models to predict missing values.

```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=2)
df_filled = imputer.fit_transform(df)
```

Q3. Imbalanced Data

Definition:
Occurs when the classes in the target variable are **not represented equally** (one class dominates).

Consequences if not handled:

- Model predicts the majority class mostly
- Poor performance on minority class
- Misleading accuracy

Q4. Up-sampling and Down-sampling

1. Up-sampling:

- Increases the number of samples in the **minority class**.

Example:
If class A = 1000, class B = 100 → duplicate/minor resample class B to match class A.

```
from sklearn.utils import resample
minority = df[df['target']==1]
majority = df[df['target']==0]
minority_upsampled = resample(minority, replace=True,
n_samples=len(majority))
df_balanced = pd.concat([majority, minority_upsampled])
```

2. Down-sampling:

- Reduces the number of samples in the **majority class**.

```
majority_downsampled = resample(majority, replace=False,
n_samples=len(minority))
df_balanced = pd.concat([majority_downsampled, minority])
```

When required:

- **Up-sampling:** Rare events prediction
- **Down-sampling:** Avoid bias in large majority class datasets

---

Q5. Data Augmentation and SMOTE

Data Augmentation:

- Artificially increase dataset size by **creating new samples**. Common in images, text, and tabular data.

SMOTE (Synthetic Minority Over-sampling Technique):

- Generates **synthetic samples** of minority class using interpolation.

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_res, y_res = smote.fit_resample(X, y)
```

Q6. Outliers

Definition:
Data points **far from the majority** of the data.

Why handle outliers?

- Distort statistical measures (mean, std)
- Reduce model performance
- Can bias ML algorithms

Handling methods:

- Remove extreme values
- Transformations (log, sqrt)
- Use robust models (Decision Trees, Random Forest)

Q7. Handling missing data in customer analysis

Techniques:

- **Remove rows/columns** with too many missing values
- **Impute missing values** (mean, median, mode)

- **Predictive imputation** using ML
- Use **algorithms robust to missing values** (Decision Trees)

Q8. Strategies to detect missing patterns

- Visualize missing data using **heatmaps** (`seaborn.heatmap(df.isnull())`)
- Use **missingno library** (`missingno.matrix(df)`)
- Statistical tests to check if missing at random (MAR) or not
- Correlation analysis between missingness and other features

---

Q9. Imbalanced medical diagnosis dataset

Strategies to evaluate models:

- Use metrics beyond accuracy: **Precision, Recall, F1-score, ROC-AUC**
- Apply **up-sampling** (SMOTE) or **down-sampling**
- Use **stratified sampling** in train-test split

Q10. Down-sampling majority class (satisfied customers)

- Randomly reduce number of satisfied customers to **match unsatisfied customers**
- Ensure class balance for model training

```
majority_downsampled = resample(majority, replace=False,
n_samples=len(minority))
balanced_df = pd.concat([majority_downsampled, minority])
```

Q11. Up-sampling minority class (rare events)

- Generate **synthetic or duplicate samples** for minority class
- Use **SMOTE** or random resampling

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X, y)
```

When needed: Fraud detection, rare disease prediction, machine failure prediction.

✅ Summary Table

Concept	Definition	Handling Techniques
Missing Values	Absent data	Drop, Mean/Median/Mode, Forward/Backward fill, ML imputation
Imbalanced Data	Unequal class distribution	Up-sampling, Down-sampling, SMOTE, Stratified sampling
Outliers	Extreme values	Remove, Transform, Robust algorithms
