Handling Null Values in a Dataset

1. Dropping Rows or Columns

When to Use:

- Dropping Rows:
  - Missing values affect only a small fraction of rows (e.g., <5% of total data).
  - The rows with missing data don't carry significant information.
- Dropping Columns:
  - A column has more than 50% missing values, making it unreliable for analysis.
  - The column is not critical for your analysis or predictive model.

Code:

```python
# Drop rows with any NaN value
df_cleaned = df.dropna()

# Drop columns with more than 50% missing values
threshold = len(df) * 0.5
df_cleaned = df.dropna(thresh=threshold, axis=1)
```

2. Simple Imputation (Mean, Median, Mode)

When to Use:

- Missing values are randomly distributed (Missing Completely at Random - MCAR).
- The dataset has a small number of missing values per column (e.g., <20%).
- Mean/Median:

- Use mean for normal distributions (symmetric data).
  - Use median for skewed distributions.
- Mode:
  - Use for categorical data with missing values.

Code:

```python
from sklearn.impute import SimpleImputer

# Mean Imputation
imputer = SimpleImputer(strategy='mean')  # Use 'median' or 'most_frequent' as needed
df.iloc[:, :] = imputer.fit_transform(df)
```

3. K-Nearest Neighbors (KNN) Imputation

When to Use:

- Missing values depend on patterns in other features (e.g., Missing at Random - MAR).
- The dataset has moderate missingness (e.g., 10-30%).
- You have numerical or continuous data where similarity between rows can provide meaningful imputation.

Code:

```python
from sklearn.impute import KNNImputer

# KNN Imputation
knn_imputer = KNNImputer(n_neighbors=5)
```

```
df.iloc[:, :] = knn_imputer.fit_transform(df)
```

4. Interpolation

When to Use:

- Data is time-series or has a logical ordering.
- Missing values are few and occur in consecutive rows.

Code:

```python
# Linear interpolation
df['Column1'] = df['Column1'].interpolate(method='linear')

# Polynomial interpolation
df['Column1'] = df['Column1'].interpolate(method='polynomial', order=2)
```

5. Forward/Backward Fill

When to Use:

- Data is time-series or sequential, and missing values are small and sporadic.
- Previous (forward fill) or subsequent (backward fill) values are reliable estimates.

Code:

```python
# Forward Fill
df['Column1'] = df['Column1'].fillna(method='ffill')
```

Backward Fill

df['Column1'] = df['Column1'].fillna(method='bfill')

```

6. Conditional Imputation (Group-Based Imputation)

When to Use:

- Missing values correlate with another feature in the dataset.

- Example: Missing age values depend on gender or location.

Code:

```python
# Fill missing values with the mean grouped by another column

df['Age'] = df.groupby('Gender')['Age'].transform(lambda x: x.fillna(x.mean()))
```

7. Replace with Predicted Values (Regression/Classification)

When to Use:

- Missing values are correlated with other features.

- You have enough data to train a model to predict missing values.

- Missing values are in a target column or a critical feature.

Code:

```python
from sklearn.linear_model import LinearRegression


# Separate rows with and without NaN in the target column

train = df[df['Target'].notnull()]

```
test = df[df['Target'].isnull()]

# Train a regression model
regressor = LinearRegression()
regressor.fit(train[['Feature1', 'Feature2']], train['Target'])

# Predict missing values
df.loc[df['Target'].isnull(), 'Target'] = regressor.predict(test[['Feature1', 'Feature2']])
```

8. Models That Handle NaNs Natively

When to Use:

- Missing values cannot be reasonably imputed.
- You're using tree-based models like:
  - HistGradientBoostingRegressor
  - XGBoost
  - LightGBM
- Missing values convey meaningful information.

Code:

```python
from sklearn.ensemble import HistGradientBoostingRegressor

model = HistGradientBoostingRegressor()
model.fit(X_train, y_train)  # X_train can contain NaNs
```

Summary of Methods:

Method	Best Use Case
Drop Rows/Columns	Small percentage of missing data (<5%) or non-critical columns.
Simple Imputation	Missing values are random and a small proportion (<20%).
KNN Imputation	Patterns in data, moderate missingness (10-30%), and numerical data.
Interpolation	Time-series data or logical ordering.
Forward/Backward Fill	Time-series with small gaps.
Conditional Imputation	Missing values depend on another column (e.g., grouped means).
Predictive Models	Missing values are critical, and sufficient data is available to train a model.
Native Models	Large missingness, using tree-based models that handle NaNs (e.g., XGBoost).