1. Fill with Mean (For Numerical Data)

If the data is numerical, you can fill the missing values with the **mean** (average) of that column.

```
df['column_name'].fillna(df['column_name'].mean(), inplace=True)
```

Use Case: When the data is normally distributed and there aren't extreme outliers.

---

2. Fill with Median (For Numerical Data)

If the data is **skewed** or contains **outliers**, it's better to use the **median** instead of the mean. The median is the middle value when the data is sorted.

```
df['column_name'].fillna(df['column_name'].median(), inplace=True)
```

Use Case: When the data is skewed, for example, in cases like salaries or prices.

---

3. Fill with Mode (For Categorical Data)

The **mode** is the **most frequent value** in the column. You can fill missing values in categorical columns (such as 'embarked', 'sex', 'country') with the mode.

```
df['column_name'].fillna(df['column_name'].mode()[0], inplace=True)
```

Use Case: For categorical columns, where the missing values can be replaced by the most frequent category.

---

4. Fill with a Constant Value

You can fill missing values with a **specific constant value** like **0**, **'Unknown'**, or any default value.

```
df['column_name'].fillna(0, inplace=True)  # Example with 0
df['column_name'].fillna('Unknown', inplace=True)  # Example with 'Unknown'
```

Use Case: When you want to replace missing data with a fixed value, for example, filling missing countries with 'Unknown'.

---

5. Forward Fill (ffill)

Forward fill replaces missing values with the value from the **previous row**. This method is very useful for time-series data.

```
df['column_name'].fillna(method='ffill', inplace=True)
```

Use Case: In time-series or sequential data when you want to fill missing values with the previous valid entry.

---

6. Backward Fill (bfill)

Backward fill replaces missing values with the value from the **next row**.

```
df['column_name'].fillna(method='bfill', inplace=True)
```

Use Case: When you want to fill missing values with the subsequent value in the dataset.

---

7. Interpolate

Interpolation uses surrounding values to estimate and fill missing values. This is particularly useful for numerical data.

```
df['column_name'].interpolate(method='linear', inplace=True)
```

Use Case: When you want to estimate missing values using linear interpolation based on surrounding data points.

---

8. Drop Missing Values

If you feel that filling the missing values isn't appropriate, you can **drop** the rows or columns that have missing values.

```
df.dropna(inplace=True)  # Drop rows with any missing values
df.dropna(subset=['column_name'], inplace=True)  # Drop rows where 'column_name' is NaN
```

Use Case: When the missing values are too many, or filling them would distort the data.

---

9. Replace with Predictive Models

You can also **predict** missing values using **machine learning models**. In this method, you train a model on the data with missing values and use it to predict the missing entries.

```
# Example: Use regression or classification model to predict missing values
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)  # Train model to predict missing data
df['column_name'] = model.predict(X_test)
```

Use Case: When you have a large dataset and you want to accurately predict missing values based on patterns in the data.

---

10. KNN Imputation

K-Nearest Neighbors (KNN) imputation is another method where missing values are filled based on **similar rows** in the dataset.

```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=3)
df = imputer.fit_transform(df)
```

Use Case: When you want to fill missing values based on the similarity of data points in the dataset.

Summary of Methods:

1. **Mean**: Use the average value for numerical data.
2. **Median**: Use the middle value when the data is skewed.
3. **Mode**: Use the most frequent value for categorical data.
4. **Constant Value**: Fill missing values with a specific fixed value.

5. **Forward Fill**: Fill missing values using the previous row's value.
6. **Backward Fill**: Fill missing values using the next row's value.
7. **Interpolate**: Estimate missing values based on surrounding data points.
8. **Drop**: Remove rows or columns that contain missing values.
9. **Predictive Models**: Use machine learning models to predict missing values.
10. **KNN Imputation**: Fill missing values based on similarity to nearest neighbors.