Data Cleaning Using Pandas in Data Science

Introduction

Data cleaning is an essential step in the data science pipeline. It involves handling missing values, dealing with duplicates, correcting incorrect data types, managing outliers, and formatting data to ensure consistency. **Pandas**, a powerful Python library, provides multiple functions to clean, preprocess, and manipulate datasets efficiently.

Finding Missing Values in Pandas

Missing values are an inevitable part of real-world data. Pandas provides several methods and properties to detect missing values efficiently.

---

1. Methods & Properties to Find Missing Values

Method/Property	Description	Example
isnull()	Returns a boolean DataFrame where True indicates missing values (NaN).	df.isnull()
notnull()	Returns a boolean DataFrame where True indicates non-missing values.	df.notnull()
isna()	Same as isnull(), detects missing values.	df.isna()

notna()	Same as notnull(), detects non-missing values.	df.notna()
isnull().sum()	Counts missing values column-wise.	df.isnull().sum()
isna().sum()	Same as isnull().sum(), counts missing values.	df.isna().sum()
info()	Displays non-null counts per column, useful for detecting missing data.	df.info()
value_counts(dropna=False)	Counts unique values, including NaN.	df['column'].value_counts(dropna=False)
df[df.isnull().any(axis=1)]	Finds rows that contain missing values.	df[df.isnull().any(axis=1)]
df[df.isnull().all(axis=1)]	Finds rows where all values are missing.	df[df.isnull().all(axis=1)]

Removing Missing Values

If missing values are few and do not impact the dataset significantly, we can remove them.

Remove Rows with Missing Values

```
df_dropped_rows = df.dropna()
print(df_dropped_rows)
```

Removes any row that contains at least one NaN value.

Remove Columns with Missing Values

```
df_dropped_columns = df.dropna(axis=1)
print(df_dropped_columns)
```

Removes columns where at least one value is NaN.

Drop Rows Only if All Values are Missing

```
df.dropna(how='all', inplace=True)
```

Removes rows where all values are missing.

Drop Rows Only if a Certain Column Has Missing Values

```
df.dropna(subset=['Age'], inplace=True)
```

Removes rows where "Age" is missing.

3. Filling Missing Values

Instead of dropping data, we can fill missing values using different techniques.

3.1 Fill with a Specific Value

```
df_filled = df.fillna(0)                    # Replace all NaN values with
print(df_filled)
```

Useful when a missing value represents "zero" or a default category.

3.2 Fill with Mean, Median, or Mode

```
df['Age'].fillna(df['Age'].mean(), inplace=True)                    # Fill with mean
```

```
df['Salary'].fillna(df['Salary'].median(), inplace=True)            # Fill with median
```

Best for numerical columns when missing values are randomly distributed.

```
df['Name'].fillna(df['Name'].mode()[0], inplace=True)               # Fill with mode
```

Useful for categorical data (e.g., replacing missing gender with the most common value).

3.3 Forward Fill (Propagating Previous Value)

```
df.fillna(method='ffill', inplace=True)                 # Forward fill (previous value)
```

Copies the last known value forward. Useful for time series data.

3.4 Backward Fill (Propagating Next Value)

```
df.fillna(method='bfill', inplace=True)                 # Backward fill (next value)
```

Copies the next known value backward.

Filling with Interpolation (Predicting Missing Values)

```
df['Age'] = df['Age'].interpolate(method='linear')
```

Estimates missing values based on neighboring values. Best for continuous numerical data.

- **Dropping values** is a quick solution but may lead to data loss.
- **Filling with mean, median, or mode** is common for numerical data.
- **Forward/Backward fill** is useful for sequential data.
- **Interpolation** predicts missing values based on trends