Data Quality with Pandas

Key Functions for Analytics

Andres Vourakis
Data Scientist & Mentor

High-quality data is the **foundation of any successful data analysis**.

Ensuring your data is accurate, complete, and consistent is crucial for making reliable decisions and building robust models.

What We'll Cover

The essential Pandas functions for assessing and improving data quality:

1. head() and tail()
2. info()
3. describe()
4. isnull() and notnull()
5. fillna() and dropna()

Inspecting Your Data

Use the `head()` and `tail()` functions to get an initial feel for your data and spot any immediate issues.

```
# Get a quick look at the first few rows of the DataFrame
df.head()
```

	product_id	product_name	user_ratings	units_sold
0	101	Product A	4.5	1500.0
1	102	Product B	3.8	1200.0
2	103	Product C	NaN	800.0
3	104	Product D	4.2	NaN
4	105	Product E	4.7	1750.0

Both funtions only return 5 rows by detault

Summarizing Your Data

info() is essential for understanding the makeup of your data, particularly when dealing with large datasets.

```
"""
Provides a concise summary of your DataFrame,
including data types and non-null counts.
"""
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   product_id    10 non-null     int64
 1   product_name  10 non-null     object
 2   user_ratings  8 non-null      float64
 3   units_sold    8 non-null      float64
dtypes: float64(2), int64(1), object(1)
memory usage: 452.0+ bytes
```

This!

Generating Summary Statistics

Use `describe()` to get a statistical overview of your data and identify areas that may need further investigation.

```
"""
Quickly assess distributions, spot outliers,
and identify potential data entry errors.
"""
df.describe()
```

	product_id	user_ratings	units_sold
count	10.00000	8.000000	8.000000
mean	105.50000	4.325000	1425.000000
std	3.02765	0.377018	385.449645
min	101.00000	3.800000	800.000000
25%	103.25000	4.050000	1175.000000
50%	105.50000	4.350000	1450.000000
75%	107.75000	4.625000	1675.000000
max	110.00000	4.800000	2000.000000

Detecting Missing Data

`isnull()` helps you locate missing data, allowing you to address gaps before they affect your analysis.

```
"""
Adding sum() helps you see the
total number of missing values.
"""
df.isnull().sum()
```

```
product_id       0
product_name     0
user_ratings     2
units_sold       2
dtype: int64
```

Handling Missing Data

fillna() and dropna() are key to maintaining the integrity of your dataset.

```
# Example: Filling missing values with 0
df_filled = df.fillna(0)

# Example: Dropping rows with missing values
df_clean = df.dropna()
```

💡 Combine These Functions

Use these functions together to perform a thorough assessment of your data's quality.

Start with `info()` to get a high-level view, use `isnull()` to pinpoint missing data, and then decide on a strategy using `fillna()` or `dropna()`.

```
# Example: Assess and clean data quality
df.info()
missing_data = df.isnull().sum()
df_clean = df.dropna() # or use df.fillna(value)
```

Andres Vourakis

Data Scientist & Mentor

Follow for more

Data Science content

• • •