

Lesson 07: Missing Data and Cleaning

Data Science with Python – BSc Course

Data Science Program

BSc Course

45 Minutes

Module 2: Data Manipulation



You have data. Now: clean, transform, combine.

Module 2 takes your raw data and makes it analysis-ready. Six lessons: missing data, operations, groupby, merging, NumPy, time series.

Module 2: from messy data to clean, analysis-ready datasets

Learning Objectives

After this lesson, you will be able to:

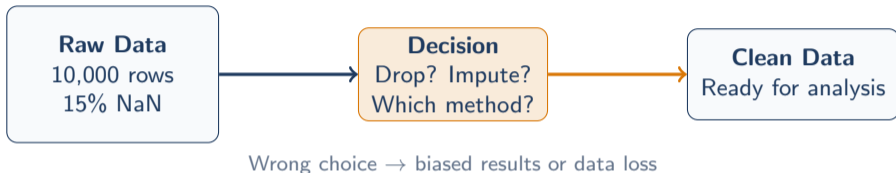
- **Detect** missing values with `isna()` and `isnull()`
- **Visualize** missing-data patterns across columns
- **Decide** when to drop vs. impute missing values
- **Apply** `fillna()` strategies: `ffill`, `bfill`, `mean`, `interpolation`
- **Clean** real financial data with missing prices and volumes

Finance Application: Handle missing stock prices caused by holidays, halts, and corporate actions.

Missing data is the #1 data quality problem in finance

Your Dataset Has 15% NaN Values. Now What?

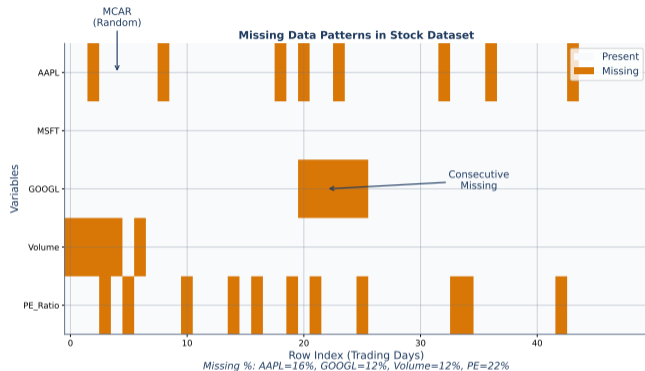
Missing data is an epidemic in real-world datasets



Three scenarios: data is missing **completely at random** (safe to drop), **at random** (can impute), or **not at random** (dangerous – investigate first).

The wrong imputation choice can introduce bias worse than the missing data

Detecting Missing Data



`isna().sum()` gives missing count per column; `isna().mean()` gives percentages

Missing Data Patterns

Why data is missing matters more than how much

MCAR

Missing Completely
At Random

Sensor glitch. Safe to drop – no
bias introduced.

```
df.dropna()
```

MAR

Missing At Random

Small-cap stocks lack analyst
coverage. Predictable from other
columns.

```
fillna(method)
```

MNAR

Missing Not
At Random

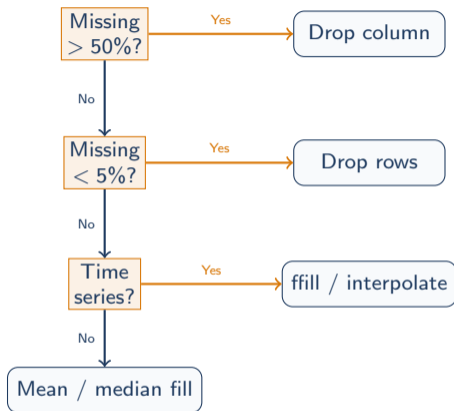
Firms hide bad earnings.
Missingness depends on the
missing value itself.

Investigate first!

Always ask: WHY is this value missing before choosing a strategy.

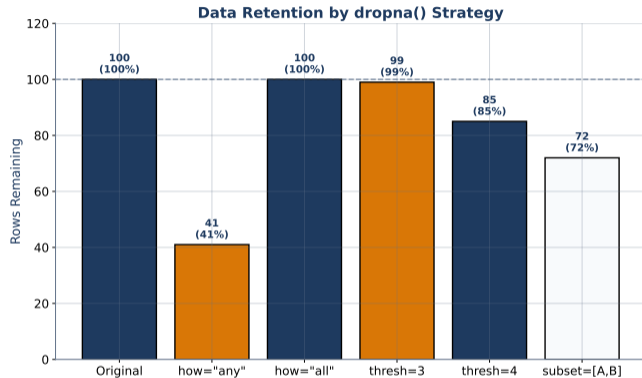
MCAR → drop safely. **MAR** → impute. **MNAR** → domain expertise needed.

Drop vs. Impute: The Decision Framework



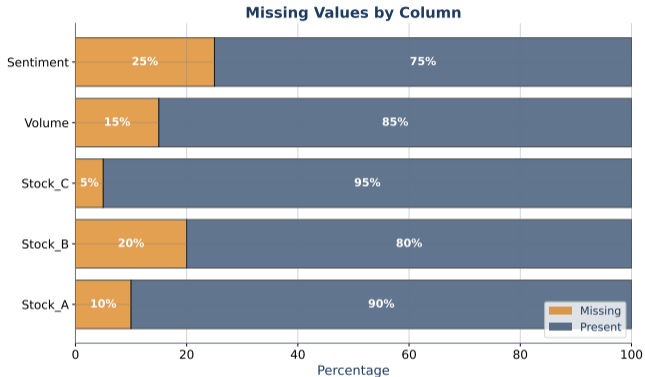
Decision tree: percentage missing + data type determines the best strategy

dropna(): Data Retention



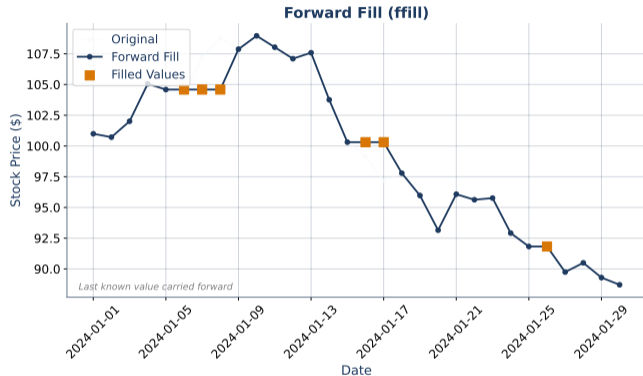
`dropna()` removes rows with NaN – monitor how much data you lose

Missing Data by Column



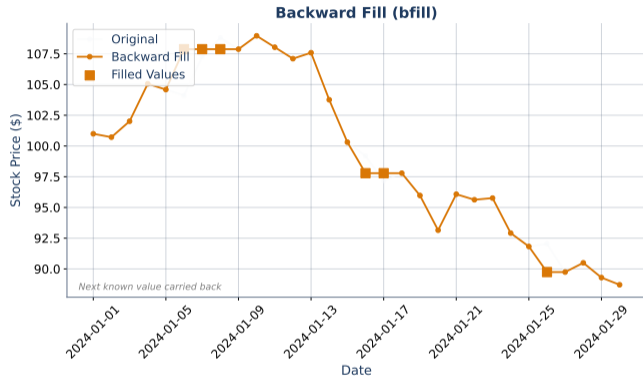
Analyze which columns have the most missing data before deciding strategy

fillna(): Forward Fill



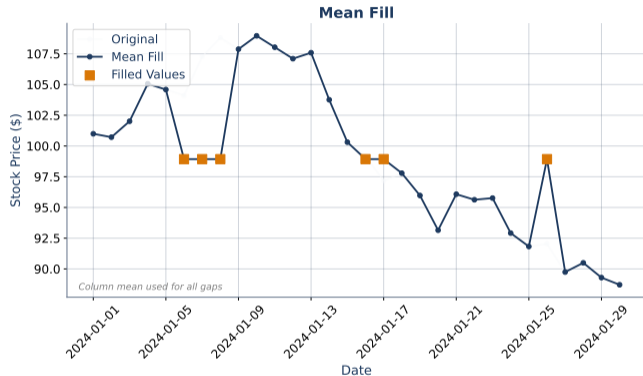
ffill carries the last known value forward – ideal for time series prices

fillna(): Backward Fill



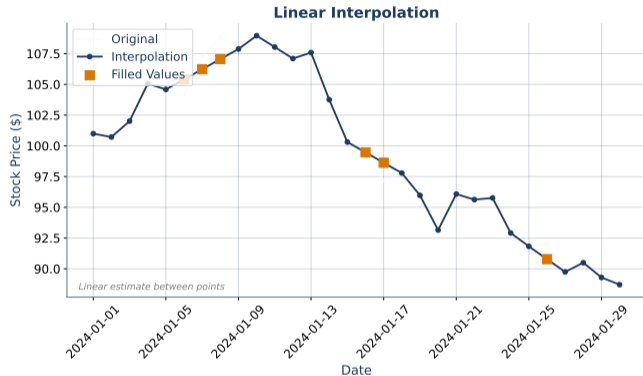
bfill uses the next known value – useful when future data is available

fillna(): Mean Fill



Mean fill replaces NaN with column average – simple but reduces variance

Interpolation



`interpolate()` estimates values between known points – best for smooth trends

Checkpoint: Test Your Understanding

Q1: When should you use `dropna()` vs `fillna()`?

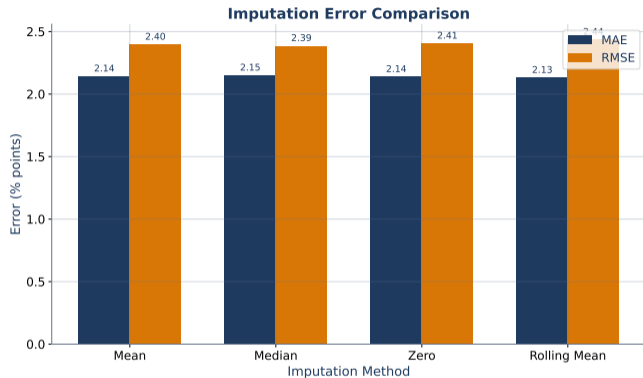
Q2: A stock price column has 3% NaN values. What imputation method would you choose and why?

Q3: What is the risk of using mean fill on a column with a strong time trend?

Think for 30 seconds. Answers: Q1: Drop when $< 5\%$ missing; fill otherwise. Q2: `ffill` (preserves last known price). Q3: Mean ignores trend, creating flat spots.

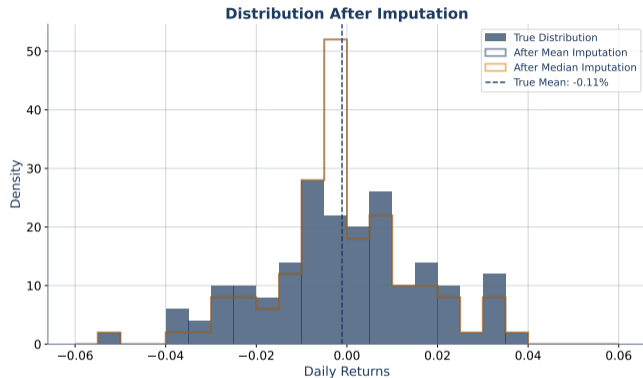
The right strategy depends on data type, percentage missing, and use case

Imputation Error Analysis



Different imputation methods produce different error magnitudes

Imputation Distribution Impact



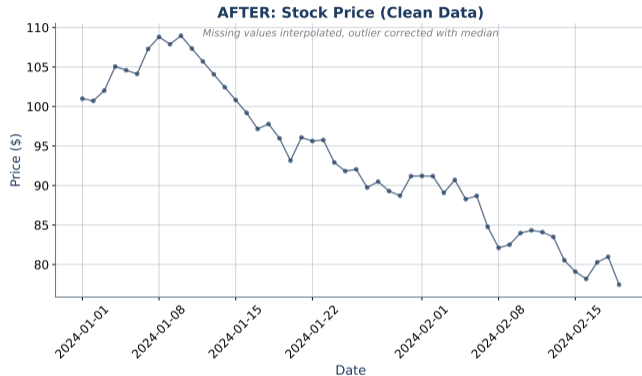
Mean fill compresses variance; interpolation preserves the distribution better

Finance: Price Data Before Cleaning



Raw price data often contains gaps from holidays, halts, and data errors

Finance: Price Data After Cleaning



ffill is the standard approach for missing stock prices

Finance: Handling Corporate Actions

Not all missing data is accidental

Common causes in finance:

- Trading halts (circuit breakers)
- Market holidays (different by country)
- Stock splits (price discontinuity)
- Delistings (permanent NaN)
- IPOs (no prior history)

Standard approaches:

```
# Holiday gaps: forward fill
prices.fillna(method='ffill')
# Volume on holidays: zero
volume.fillna(0)
# Delisted stocks: drop column
df.dropna(axis=1, thresh=N)
```

Always check: is the data truly missing, or does the gap carry information?

Trading halts = information event. Don't blindly impute.

Validation is critical: After imputation, compare the distribution of the filled column against the original. If variance collapsed or mean shifted, try a different method.

Always validate: imputation that distorts the distribution is worse than NaN

Detecting and Removing Duplicates

Detecting Duplicates in Stock Data

Index	Date	Price	Volume	Duplicate?
0	2024-01-01	\$100.0	1,000	No
1	2024-01-02	\$101.5	1,500	No
2	2024-01-02	\$101.5	1,500	Yes
3	2024-01-03	\$102.0	2,000	No
4	2024-01-03	\$102.3	2,100	No
5	2024-01-03	\$102.0	2,000	Yes
6	2024-01-04	\$103.0	1,800	No
7	2024-01-05	\$104.5	2,200	No
8	2024-01-05	\$104.5	2,200	Yes

Total Rows: 9 | Duplicates: 3 | Unique: 6
df.duplicated() marks all rows after first occurrence as duplicates

`df.duplicated()` finds exact row matches; `drop_duplicates()` removes them

Hands-on Exercise (25 min)

Clean a messy stock dataset:

1. Load stock data and check missing values: `df.isna().sum()`
2. Visualize: which columns have the most NaN values?
3. Use `dropna()` on rows with $< 5\%$ missing across all columns
4. Use `fillna(method='ffill')` for the price column
5. Use `fillna(df['Volume'].mean())` for volume
6. Check for duplicates with `df.duplicated().sum()`
7. Compare distributions before and after imputation

Bonus: Try `interpolate()` on prices and compare to `ffill`.

Real datasets need 60–80% of your time on cleaning

The Big Idea



Missing data handled wrong = biased analysis. Handled right = reliable results.

Data cleaning is unglamorous but determines the quality of every downstream analysis

Lesson Summary

Key Takeaways:

- `isna().sum()` detects missing values; `isna().mean()` gives percentages
- MCAR/MAR/MNAR patterns determine the right cleaning strategy
- `dropna()` when few values missing; `fillna()` to preserve data
- `ffill` for time series; mean/median for cross-sectional; interpolate for smooth trends
- Always validate imputation by comparing distributions before and after

Up Next: L08 – Basic Operations

You have clean data. Now learn `apply()`, arithmetic, sorting, and rolling calculations to transform it.

Module 2 started – from missing data to transformed, analysis-ready data