결측치 확인: isnull, isna

확인 필요한 상황

- **데이터 품질 검증**: 데이터를 처음 불러온 후, 각 변수에 결측치가 얼마나 포함되어 있는지 확인하여 데이터의 완전성을 평가할 때.
- **결측치 처리 전략 수립**: 결측치의 개수와 비율을 파악하여, 해당 변수를 제거할지, 특정 값으로 채울지(대체), 아니면 결측치를 예측하는 모델을 만들지 등의 처리 방법을 결정하기 전 단계.
- **데이터 시각화**: `seaborn`의 `heatmap`과 결합하여 데이터셋 전체의 결측치 분포 패턴을 시각적으로 확인할 때.

예제 데이터프레임 생성

- Numpy의 `np.nan`과 Python 내장 `None`은 모두 Pandas에서 결측치로 인식된다.

```
import pandas as pd
import numpy as np

data = {'A': [1, 2, np.nan, 4],
        'B': [5, np.nan, np.nan, 8],
        'C': ['x', 'y', 'z', 'w'],
        'D': [np.nan, None, np.nan, np.nan]}
df = pd.DataFrame(data)
```

1. `isnull()` / `isna()`

- 데이터프레임 또는 시리즈의 각 원소가 결측치인지 아닌지를 검사하여, 결측치이면 `True`, 아니면 `False`를 담은 불리언(Boolean) 객체를 반환
- 메서드 자체만으로는 결측치의 위치만 알 수 있으며, 개수를 파악하려면 `sum()`을, 비율을 파악하려면 `mean()`을 함께 사용해야함

```
# isnull()을 사용하여 결측치 위치 확인
print("--- isnull() result ---")
print(df.isnull())
'''
--- isnull() result ---
       A      B      C     D
0  False  False  False  True
1  False   True  False  True
2   True   True  False  True
3  False  False  False  True
'''
# isna()는isnull()과 동일한 결과를 반환합니다.
# print(df.isna())
```

2. `isnull().sum()`

- 각 열에 있는 결측치의 총 개수를 계산
- `sum()` 함수는 기본적으로 열(axis=0) 단위로 합계를 계산
    - 행(axis=1) 단위로 결측치 개수를 세려면 `axis=1` 인자를 추가 필요

```
# 각 열의 결측치 개수 확인
print("--- Missing values per column ---")
print(df.isnull().sum())
'''
--- Missing values per column ---
A    1
B    2
C    0
D    4
dtype: int64
'''

# 전체 데이터프레임의 총 결측치 개수 확인
print("\n--- Total missing values in DataFrame ---")
print(df.isnull().sum().sum())
'''
--- Total missing values in DataFrame ---
7
'''
```

3. 결측치 비율 확인

- 전체 데이터 대비 결측치가 차지하는 비율을 확인하여, 결측의 심각성을 판단

```
# 각 열의 결측치 비율 확인
print("--- Missing value ratio per column ---")
print(df.isnull().mean() * 100)
'''
--- Missing value ratio per column ---
A     25.0
B     50.0
C      0.0
D    100.0
dtype: float64
'''
```

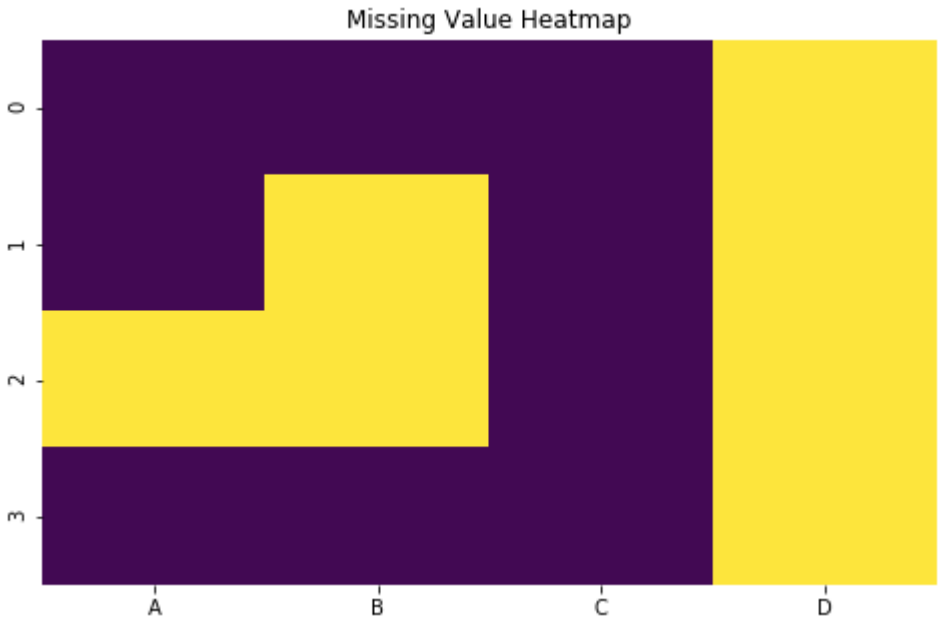
- **결과 해석**
    - `D` 열은 100% 결측치이므로, 분석에 사용할 수 없어 제거 대상이 될 가능성이 높습니다.
    - `B` 열은 50%의 높은 결측 비율을 가지므로, 단순한 평균값 대체보다는 더 정교한 처리 방법이 필요할 수 있습니다.

4. 결측치 시각화

- 결측치의 분포 패턴을 시각적으로 파악
- **주의사항**: `seaborn` 라이브러리가 필요합니다. (`pip install seaborn`)

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Value Heatmap')
plt.show()
```



- **결과 해석**: 히트맵에서 노란색(또는 지정된 색상)으로 표시된 부분이 결측치를 의미합니다. 이를 통해 특정 행이나 열에 결측치가 집중되어 있는지, 또는 무작위로 분포하는지 등의 패턴을 한눈에 파악할 수 있습니다.

장단점 및 대안

메서드	장점	단점	대안/보완
`isnull()` / `isna()`	직관적이고 사용하기 쉬움. Pandas의 기본 기능으로 추가 라이브러리 불필요.	그 자체만으로는 요약 정보를 주지 못하고, `sum()`이나 `mean()`과 결합해야 함.	`notnull()` / `notna()`: `isnull()`/`isna()`와 정반대로, 결측치가 아닌 유효한 값의 위치를 `True`로 반환함. 유효한 데이터의 개수를 셀 때 `df.notnull().sum()`과 같이 사용할 수 있음.
`isnull().sum()`	각 열의 결측치 개수를 빠르고 간결하게 요약해 줌.	결측치의 위치나 패턴에 대한 정보는 제공하지 않음.	`df.info()`: `Non-Null Count`를 통해 결측치 개수를 간접적으로 확인할 수 있으며, 데이터 타입 정보까지 함께 제공함.

메서드	장점	단점	대안/보완
시각화 (Heatmap)	결측치 분포의 전체적인 패턴과 집중도를 시각적으로 파악하는 데 매우 효과적.	데이터가 매우 클 경우(행/열이 많을 경우) 히트맵이 잘 보이지 않거나 렌더링이 느려질 수 있음.	`missingno` 라이브러리 (`pip install missingno`): 결측치 시각화에 특화된 라이브러리로, `msno.matrix()`, `msno.bar()` 등 더 다양하고 상세한 시각화 기능을 제공함.