결측치 처리: dropna, fillna, interpolate

적용 가능한 상황

- `dropna()`:
  - 결측치가 포함된 데이터의 양이 전체 데이터에 비해 매우 적을 때 (e.g., 5% 미만).
  - 특정 변수(열)에 결측치가 너무 많아(e.g., 50% 이상) 변수 자체가 유용하지 않다고 판단될 때.
  - 타겟 변수(label)에 결측치가 있는 경우 (해당 데이터는 모델 학습에 사용할 수 없으므로).
- `fillna()`:
  - **상수 대체**: 결측치가 '없음' 또는 '0'과 같은 특정 의미를 가질 때. (e.g., 할인 금액의 결측치는 할인이 없음을 의미)
  - **통계량 대체**: 데이터가 특정 분포를 따를 때. 평균값(mean)은 정규분포에 가까울 때, 중앙값(median)은 이상치(outlier)가 많을 때, 최빈값(mode)은 범주형 데이터에 적합합니다.
  - **Forward/Backward Fill**: 시계열 데이터에서 이전 시점의 값으로 현재 시점의 결측치를 채우거나(`ffill`), 이후 시점의 값으로 채울 때(`bfill`).
- `interpolate()`:
  - 시계열 데이터(주가, 센서 데이터 등)에서 누락된 값을 양쪽 값들을 이용하여 추정할 때.
  - 데이터 포인트들이 일정한 간격이나 추세를 가질 때, `fillna`보다 더 정교한 값으로 대체하고 싶을 때.

예제 데이터프레임 생성

```
import pandas as pd
import numpy as np

df = pd.DataFrame({'A': [1, 2, np.nan, 4, 5],
                   'B': [np.nan, 10, 20, 30, np.nan],
                   'C': ['p', 'q', 'r', np.nan, 't'],
                   'D': [100, 200, 300, 400, 500]})
```

1. dropna()

- 결측치가 포함된 행(row)이나 열(column)을 데이터프레임에서 제거
- 가장 간단한 방법이지만, 정보 손실 발생 가능성 있음
- 원본 데이터프레임을 직접 수정하려면 `inplace=True` 옵션 사용
- `how`와 `thresh` 인자를 적절히 사용하여 제거 조건을 세밀하게 조정 가능

```
# 결측치가 하나라도 있는 행을 제거 (기본값)
df_dropped_row = df.dropna()
print("--- Dropped rows with any NaN ---")
print(df_dropped_row)
```

```
'''
--- Dropped rows with any NaN ---
     A     B  C    D
1  2.0  10.0  q  200
'''

# 결측치가 있는 열을 제거
df_dropped_col = df.dropna(axis=1)
print("\n--- Dropped columns with any NaN ---")
print(df_dropped_col)
'''
--- Dropped columns with any NaN ---
     D
0  100
1  200
2  300
3  400
4  500
'''

# 특정 열('C')에 결측치가 있는 행만 제거
df_dropped_subset = df.dropna(subset=['C'])
print("\n--- Dropped rows with NaN in column C ---")
print(df_dropped_subset)
'''
--- Dropped rows with NaN in column C ---
     A     B  C    D
0  1.0   NaN  p  100
1  2.0  10.0  q  200
2  NaN  20.0  r  300
4  5.0   NaN  t  500
'''
```

- **결과 해석**
  - `dropna()`: `A`, `B`, `C` 열에 결측치가 있는 0, 1, 2, 3, 4번 인덱스 행이 모두 제거되어 1개의 행만 남습니다.
  - `dropna(axis=1)`: 결측치가 포함된 `A`, `B`, `C` 열이 모두 제거되고 `D` 열만 남습니다.
  - `dropna(subset=['C'])`: `C` 열에 결측치가 있는 3번 인덱스 행만 제거됩니다.
- **주요 하이퍼파라미터 (인자)**
  - `axis`: `0` 또는 `'index'` (행 제거, 기본값), `1` 또는 `'columns'` (열 제거).
  - `how`: `'any'` (결측치가 하나라도 있으면 제거, 기본값), `'all'` (모든 값이 결측치일 때만 제거).
  - `thresh`: 정수값. 해당 행/열에서 결측치가 아닌 값의 개수가 이 값보다 적을 경우에만 제거. (e.g., `thresh=3`이면 유효한 값이 3개 미만인 행/열을 제거)
  - `subset`: 검사를 수행할 열 또는 행의 레이블 리스트.
  - `inplace`: `True`로 설정하면 원본 데이터프레임을 직접 수정. 기본값은 `False`.

2. `fillna()`

- 결측치를 특정 값으로 채움
  - 특정 값 : 상수(e.g., 0), 평균값, 중앙값, 최빈값 등 통계량 / 앞이나 뒤의 값
- 수치형 데이터에 문자열을 채우면 해당 열의 데이터 타입이 `object`로 변경될 수 있음
  - 각 열의 특성에 맞는 대체 방법을 사용하는 것이 중요

```
# 특정 상수로 채우기
df_fill_zero = df.fillna(0)
print("--- Filled with 0 ---")
print(df_fill_zero)
'''
--- Filled with 0 ---
     A     B  C    D
0  1.0   0.0  p  100
1  2.0  10.0  q  200
2  0.0  20.0  r  300
3  4.0  30.0  0  400
4  5.0   0.0  t  500
'''

# 각 열의 평균값(수치형) 또는 최빈값(범주형)으로 채우기
df_filled = df.copy()
df_filled['A'] = df_filled['A'].fillna(df_filled['A'].mean())
df_filled['B'] = df_filled['B'].fillna(df_filled['B'].median())
df_filled['C'] = df_filled['C'].fillna(df_filled['C'].mode()[0])
print("\n--- Filled with mean/median/mode ---")
print(df_filled)
'''
--- Filled with mean/median/mode ---
     A     B  C    D
0  1.0  20.0  p  100
1  2.0  10.0  q  200
2  3.0  20.0  r  300
3  4.0  30.0  p  400
4  5.0  20.0  t  500
'''

# 이전 값으로 채우기 (Forward Fill)
df_ffill = df.fillna(method='ffill') # 또는 df.ffill()
print("\n--- Forward Fill ---")
print(df_ffill)
'''
--- Forward Fill ---
     A     B  C    D
0  1.0   NaN  p  100
1  2.0  10.0  q  200
2  2.0  20.0  r  300
3  4.0  30.0  r  400
4  5.0  30.0  t  500
'''
```

- **결과 해석**

    - `fillna(0)`: 모든 `NaN` 값이 `0`으로 대체됩니다.
    - 평균/중앙/최빈값 대체: 각 열의 통계적 특성을 반영하여 결측치가 채워집니다. `mode()`는 시리즈를 반환하므로 `[0]`으로 첫 번째 최빈값을 선택해야 합니다.
    - `ffill`: `A`열의 `NaN`은 바로 앞의 `2.0`으로, `B`열의 첫 `NaN`은 채워지지 않고(앞의 값이 없으므로), 두 번째 `NaN`은 `30.0`으로 채워집니다.

- **주요 하이퍼파라미터 (인자)**

    - `value`: 결측치를 대체할 값 (스칼라, 딕셔너리, 시리즈, 데이터프레임).
    - `method`: 채우기 방법. `'ffill'` 또는 `'pad'` (앞의 값으로 채움), `'bfill'` 또는 `'backfill'` (뒤의 값으로 채움).
    - `axis`: `fillna`를 적용할 축. 기본값은 `0`.
    - `limit`: `ffill` 또는 `bfill` 사용 시, 연속적으로 채울 최대 결측치 개수.
    - `inplace`: `True`로 설정하면 원본을 직접 수정.

3. `interpolate()`

- 결측치를 보간법(interpolation)을 사용하여 추정된 값으로 채움
- 주로 시계열 데이터와 같이 데이터 포인트 간에 순서와 관계가 있을 때 유용하며, 선형 보간이 기본적으로 사용됨
- 데이터가 정렬되어 있어야 의미 있는 보간이 가능
- 시계열 데이터의 경우 시간 인덱스를 기준으로 보간하는 것이 더 정확할 수 있음 (`method='time'`)

```
df_inter = df.copy()
df_inter['A'] = df_inter['A'].interpolate()
df_inter['B'] = df_inter['B'].interpolate(method='linear') # 기본값
print("--- Interpolated ---")
print(df_inter)
'''
--- Interpolated ---
     A     B    C    D
0  1.0   NaN    p  100
1  2.0  10.0    q  200
2  3.0  20.0    r  300
3  4.0  30.0  NaN  400
4  5.0  30.0    t  500
'''
```

- **결과 해석**

    - `A`열의 `NaN` (인덱스 2)은 앞의 값 `2.0`과 뒤의 값 `4.0`의 중간인 `3.0`으로 채워졌습니다.
    - `B`열의 첫 `NaN`은 앞에 값이 없어 채워지지 않고, 마지막 `NaN`은 뒤에 값이 없어 채워지지 않습니다.

- **주요 하이퍼파라미터 (인자)**:

    - `method`: 보간 방법. `'linear'` (기본값, 선형), `'polynomial'` (다항식), `'spline'` (스플라인), `'time'` (시계열 인덱스 기반) 등. `polynomial`과 `spline`은 `order` 인자가 필요합니다.
    - `axis`: 보간을 적용할 축. 기본값은 `0`.
    - `limit_direction`: `'forward'`, `'backward'`, `'both'` 중 선택. 연속된 `NaN`을 채울 방향을 지정.

- `inplace`: `True`로 설정하면 원본을 직접 수정.

장단점 및 대안

방법	장점	단점	대안
`dropna()`	구현이 매우 간단하고 빠름.	데이터의 정보 손실이 발생하여 분석 결과를 왜곡할 수 있음.	결측치 비율이 낮은 경우에만 제한적으로 사용. `thresh` 인자를 활용하여 제거 기준을 완화할 수 있음.
`fillna()`	데이터의 특성을 반영하여 결측치를 채울 수 있음 (평균, 중앙값 등). 정보 손실을 방지함.	대체하는 값에 따라 데이터의 분포가 변하고, 분산이 감소할 수 있음. 어떤 값으로 채울지에 대한 주관적인 결정이 필요.	다중 대체(Multiple Imputation), KNN 기반 대체 등 더 정교한 통계적 방법 사용.
`interpolate()`	데이터의 추세를 반영하여 `fillna`보다 더 그럴듯한 값으로 채울 수 있음. 특히 시계열 데이터에 효과적.	데이터가 특정 순서나 추세를 따르지 않으면 부적절함. 데이터 양 끝의 결측치는 처리하지 못할 수 있음.	`scikit-learn`의 `IterativeImputer`: 다른 변수들을 피처로 사용하여 결측치를 예측하는 모델(e.g., 회귀)을 만들어 결측치를 채우는 정교한 방법.