

03. 데이터 전처리

1. 누락 데이터 처리

○ 누락 데이터 처리

- 데이터를 파일로 입력할 때 빠트리거나 파일 형식 변환 중 소실되는 경우 발생할 수 있음.
- Pandas에서는 유효한 값이 존재하지 않는 누락데이터를 NaN으로 표시함.
- Seaborn 모듈의 내부데이터인 titanic 데이터셋을 사용하여 누락데이터를 확인하고 누락데이터의 개수를 확인하도록 하겠음.

1) 누락 데이터 확인

- 데이터프레임의 요약 정보를 출력해주는 명령어인 info() 함수를 사용하여 누락 데이터의 개수를 확인할 수 있음.
- info() 함수를 사용하는 방법은 다음과 같음.

데이터프레임객체.info()

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('titanic')
In [3]: print(df)
survived  pclass    sex  age  ... deck  embark_town  alive  alone
0         0      3  male  22.0  ... NaN  Southampton  no    False
1         1      1  female  38.0  ... C    Cherbourg   yes    False
2         1      3  female  26.0  ... NaN  Southampton  yes    True
3         1      1  female  35.0  ... C    Southampton  yes    False
4         0      3  male   35.0  ... NaN  Southampton  no    True
..      ...    ...    ...    ...  ...    ...      ...    ...
886        0      2  male   27.0  ... NaN  Southampton  no    True
887        1      1  female  19.0  ... B    Southampton  yes    True
888        0      3  female   NaN  ... NaN  Southampton  no    False
889        1      1  male   26.0  ... C    Cherbourg   yes    True
890        0      3  male   32.0  ... NaN  Queenstown  no    True

[891 rows x 15 columns]
```

```
In [4]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
#   Column             Non-Null Count  Dtype
---  -
0   survived           891 non-null    int64
1   pclass             891 non-null    int64
2   sex                891 non-null    object
3   age                714 non-null    float64
4   sibsp             891 non-null    int64
5   parch             891 non-null    int64
6   fare              891 non-null    float64
7   embarked           889 non-null    object
8   class             891 non-null    category
9   who                891 non-null    object
10  adult_male         891 non-null    bool
11  deck              203 non-null    category
12  embark_town        889 non-null    object
13  alive             891 non-null    object
14  alone             891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

- 실행 결과를 살펴보면, RangeIndex에 전체 데이터의 개수를 확인할 수 있고, Data columns를 통해 각 변수별 유효한 데이터의 개수를 확인할 수 있음.
- 예를들어 age 변수의 경우 714개의 유효한 정수형 데이터가 있는 것을 알 수 있습니다. 즉, 전체 데이터의 수 891에서 유효한 데이터의 수 714를 빼면 177개의 누락데이터가 있다는 것을 알 수 있음.
- 또한, value_counts() 함수를 이용하여 일부 변수를 선택 후, 변수의 누락 데이터의 개수를 확인할 수 있음. value_counts() 함수는 변수의 빈도분석을 실시하는 명령어로 dropna 옵션에 False를 입력하면 누락 데이터를 표현할 수 있음.
- value_counts() 함수를 사용하는 방법은 다음과 같음.

데이터프레임의 변수 선택.value_counts(dropna=True/False)

```
In [5]: nan_town = df['embark_town'].value_counts(dropna=False)
In [6]: print(nan_town)
Southampton    644
Cherbourg      168
Queenstown     77
NaN             2
Name: embark_town, dtype: int64
```

- 실행 결과를 살펴보면, embark_town의 요인에 대한 빈도수를 확인할 수 있고, NaN을 통해 2개의 누락데이터가 있다는 것을 알 수 있음. 여기서 dropna 옵션에 False를 지정하였을 때 누락데이터의 개수를 확인

할 수 있음.

- 그리고, `isnull()`과 `notnull()` 함수를 이용하여 누락 데이터를 직접적으로 찾을 수 있습니다. `isnull()` 함수는 누락 데이터면 `True`로 값을 반환하고, 유효한 데이터가 존재하면 `False`를 반환하는 함수이고, `notnull()` 함수는 누락 데이터면 `False`로 값을 반환하고, 유효한 데이터가 존재하면 `True`를 반환하는 함수임.
- `isnull()`와 `notnull()` 함수를 사용하는 방법은 다음과 같음.

데이터프레임객체.isnull()

데이터프레임객체.notnull()

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('titanic')
In [3]: print(df.head())
survived  pclass    sex  age  ...  deck  embark_town  alive  alone
0         0      3  male  22.0  ...  NaN  Southampton  no    False
1         1      1  female  38.0  ...   C   Cherbourg  yes    False
2         1      3  female  26.0  ...  NaN  Southampton  yes     True
3         1      1  female  35.0  ...   C   Southampton  yes    False
4         0      3  male  35.0  ...  NaN  Southampton  no     True

[5 rows x 15 columns]
```

- 데이터프레임에 `head()` 함수를 사용하여 첫 5행의 자료를 살펴보면 `deck` 변수에 누락 데이터가 존재하는 것을 확인할 수 있음.

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('titanic')
In [3]: print(df.head())
survived  pclass    sex  age  ...  deck  embark_town  alive  alone
0         0      3  male  22.0  ...  NaN  Southampton  no    False
1         1      1  female  38.0  ...   C   Cherbourg  yes    False
2         1      3  female  26.0  ...  NaN  Southampton  yes     True
3         1      1  female  35.0  ...   C   Southampton  yes    False
4         0      3  male  35.0  ...  NaN  Southampton  no     True

[5 rows x 15 columns]

In [4]: print(df.head().isnull())
survived  pclass    sex  age  ...  deck  embark_town  alive  alone
0     False    False    False  False  ...   True      False    False    False
1     False    False    False  False  ...  False      False    False    False
2     False    False    False  False  ...   True      False    False    False
3     False    False    False  False  ...  False      False    False    False
4     False    False    False  False  ...   True      False    False    False

[5 rows x 15 columns]
```

- 여기에 `isnull()` 함수를 적용하면 `deck` 변수의 누락데이터는 `True`로 반환되고 나머지 유효한 값들은 `False`로 변환된 것을 확인할 수 있음.

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('titanic')

In [3]: print(df.head())
survived  pclass    sex  age  ... deck embark_town alive alone
0         0      3  male  22.0  ... NaN  Southampton   no  False
1         1      1  female  38.0  ... C    Cherbourg   yes  False
2         1      3  female  26.0  ... NaN  Southampton   yes  True
3         1      1  female  35.0  ... C    Southampton   yes  False
4         0      3   male  35.0  ... NaN  Southampton   no   True

[5 rows x 15 columns]

In [4]: print(df.head().notnull())
survived  pclass    sex  age  ... deck embark_town alive alone
0      True   True   True  True  ... False      True   True  True
1      True   True   True  True  ...  True      True   True  True
2      True   True   True  True  ... False      True   True  True
3      True   True   True  True  ...  True      True   True  True
4      True   True   True  True  ... False      True   True  True

[5 rows x 15 columns]
```

- 그리고 notnull() 함수를 적용하면 deck 변수의 누락 데이터는 False로 반환되고 나머지 유효한 값들은 True로 변환된 것을 확인할 수 있음.

2) 누락 데이터 제거

- 누락 데이터가 존재하는 열 또는 행을 삭제하는 방법을 알아보도록 하겠습니다. 열을 삭제하면 분석 대상이 갖는 특성(변수)을 제거하고, 행을 삭제하면 분석 대상의 관측값 (레코드)을 제거하게 됨.
- 누락 데이터를 제거하는 명령어인 dropna() 함수를 사용하는 방법은 다음과 같음.

데이터프레임객체.dropna(axis=0(행)/1(열), thresh=유효데이터개수 기준)

```
In [4]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   survived    891 non-null    int64
1   pclass      891 non-null    int64
2   sex         891 non-null    object
3   age         714 non-null    float64
4   sibsp       891 non-null    int64
5   parch       891 non-null    int64
6   fare        891 non-null    float64
7   embarked    889 non-null    object
8   class       891 non-null    category
9   who         891 non-null    object
10  adult_male  891 non-null    bool
11  deck        203 non-null    category
12  embark_town 889 non-null    object
13  alive       891 non-null    object
14  alone       891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```


- 주어진 데이터셋을 살펴보면 age 변수에 177개, embarked 변수에 2개, deck 변수에 688개, embark_town 변수에 2개의 누락데이터가 있는 것을 확인할 수 있음.

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('titanic')
In [3]: df_thresh = df.dropna(axis=1, thresh=495)
In [4]: df_thresh.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0   survived    891 non-null    int64
1   pclass      891 non-null    int64
2   sex         891 non-null    object
3   age         714 non-null    float64
4   sibsp       891 non-null    int64
5   parch       891 non-null    int64
6   fare        891 non-null    float64
7   embarked    889 non-null    object
8   class       891 non-null    category
9   who         891 non-null    object
10  adult_male  891 non-null    bool
11  embark_town 889 non-null    object
12  alive       891 non-null    object
13  alone       891 non-null    bool
dtypes: bool(2), category(1), float64(2), int64(4), object(5)
memory usage: 79.4+ KB
```

- 누락의 비율이 50%가 넘는 변수를 제거하기 위해 axis 옵션에 1을 입력하고, thresh 옵션에 유효데이터 개수의 기준을 495로 입력하여 누락이 많이 존재하는 변수를 제거할 수 있음.
- info() 함수를 통해 주어진 데이터셋에서는 deck 변수가 제거된 것을 확인할 수 있음.
- 그리고 데이터프레임의 변수들 중 데이터 누락이 존재하면 안되는 중요한 변수가 존재하는 경우 데이터프레임의 레코드를 제거하는 것이 좋습니다. 이를 위해 dropna() 함수를 사용하는 방법은 다음과 같음.

데이터프레임객체.dropna(subset=[기준변수명], how='any'/'all', axis=0(행)/1(열))

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('titanic')
In [3]: print(df)
survived  pclass    sex  age  ... deck embark_town  alive  alone
0         0        3  male  22.0  ... NaN  Southampton  no  False
1         1        1  female  38.0  ... C  Cherbourg  yes  False
2         1        3  female  26.0  ... NaN  Southampton  yes  True
3         1        1  female  35.0  ... C  Southampton  yes  False
4         0        3  male  35.0  ... NaN  Southampton  no  True
..      ...      ...      ...      ...      ...      ...      ...      ...
886        0        2  male  27.0  ... NaN  Southampton  no  True
887        1        1  female  19.0  ... B  Southampton  yes  True
888        0        3  female  NaN  ... NaN  Southampton  no  False
889        1        1  male  26.0  ... C  Cherbourg  yes  True
890        0        3  male  32.0  ... NaN  Queenstown  no  True

[891 rows x 15 columns]
```

- 주어진 데이터셋에서 age 변수가 중요하다고 할 때, age 변수에 누락이 있는 모든 레코드를 제거하는 것이 좋음.
- subset 옵션에 누락의 기준이 되는 변수명을 입력할 수 있음. 중요한 점은 subset 옵션 입력시 기준 변수명을 리스트 형태로 입력을 해야 함.
- how 옵션을 통해 subset 옵션에서 입력한 기준 변수명의 누락 기준을 설정할 수 있음. any옵션을 입력하였을 때는 기준 변수 중 하나라도 누락이 있는 경우 레코드를 제거하는 방식이고, all옵션을 입력하였을 때는 기준 변수 모두에 누락이 있는 경우 레코드가 제거되는 방식임.
- 주어진 데이터셋을 출력하였을 때 888번 행에서 age 변수의 값이 누락되어 있는 것을 확인할 수 있음.

```
In [4]: df_age = df.dropna(subset=['age'], how='any', axis=0)
In [5]: print(df_age)
survived  pclass    sex  age  ... deck embark_town  alive  alone
0         0        3  male  22.0  ... NaN  Southampton  no  False
1         1        1  female  38.0  ... C  Cherbourg  yes  False
2         1        3  female  26.0  ... NaN  Southampton  yes  True
3         1        1  female  35.0  ... C  Southampton  yes  False
4         0        3  male  35.0  ... NaN  Southampton  no  True
..      ...      ...      ...      ...      ...      ...      ...      ...
885        0        3  female  39.0  ... NaN  Queenstown  no  False
886        0        2  male  27.0  ... NaN  Southampton  no  True
887        1        1  female  19.0  ... B  Southampton  yes  True
889        1        1  male  26.0  ... C  Cherbourg  yes  True
890        0        3  male  32.0  ... NaN  Queenstown  no  True

[714 rows x 15 columns]
```

- 기준변수를 age로 하였을 때 age 변수에 누락이 존재하는 888번 행의 레코드가 제거된 것을 확인할 수 있음.

3) 누락 데이터 치환

- 데이터셋의 품질을 높일 목적으로 누락 데이터를 무작정 삭제해버리면 어렵게 수집한 데이터를 활용 못하게 됨.
- 데이터분석의 정확도는 데이터의 품질 외에도 데이터의 양에 의해서도 영향을 받으므로 데이터 중에서 일부가 누락되어 있더라도 나머지 데이터를 최대한 살려서 데이터 분석에 활용하는 것이 좋은 결과를 얻는 경우가 많음.
- 누락 데이터를 바꿔서 대체할 값으로는 데이터의 분포와 특성을 잘 나타낼 수 있는 평균값, 최빈값 등을 활용함.
- 누락 데이터를 치환하는 명령어인 fillna() 함수를 사용하는 방법은 다음과 같음.

데이터프레임객체.fillna(대체값, inplace=False/True)

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('titanic')
In [3]: print(df.head(10))
```

| | survived | pclass | sex | age | ... | deck | embark_town | alive | alone |
|---|----------|--------|--------|------|-----|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.0 | ... | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.0 | ... | C | Cherbourg | yes | False |
| 2 | 1 | 3 | female | 26.0 | ... | NaN | Southampton | yes | True |
| 3 | 1 | 1 | female | 35.0 | ... | C | Southampton | yes | False |
| 4 | 0 | 3 | male | 35.0 | ... | NaN | Southampton | no | True |
| 5 | 0 | 3 | male | NaN | ... | NaN | Queenstown | no | True |
| 6 | 0 | 1 | male | 54.0 | ... | E | Southampton | no | True |
| 7 | 0 | 3 | male | 2.0 | ... | NaN | Southampton | no | False |
| 8 | 1 | 3 | female | 27.0 | ... | NaN | Southampton | yes | False |
| 9 | 1 | 2 | female | 14.0 | ... | NaN | Cherbourg | yes | False |

[10 rows x 15 columns]

- 주어진 데이터셋에서 age 변수는 연속형 자료이므로 age 변수의 누락된 데이터를 mean() 함수를 이용하여 나머지 데이터의 평균값으로 치환할 수 있음. 주어진 데이터셋을 출력하였을 때 5번 행에서 age 변수의 값이 누락되어 있는 것을 확인할 수 있음.
- 여기서 원본 객체를 변경하기 위해서 inplace=True 옵션을 추가해야 함.

```
In [4]: mean_age = df['age'].mean(axis=0)
In [5]: df['age'].fillna(mean_age, inplace=True)
In [6]: print(df.head(10))
```

| | survived | pclass | sex | age | ... | deck | embark_town | alive | alone |
|---|----------|--------|--------|-----------|-----|------|-------------|-------|-------|
| 0 | 0 | 3 | male | 22.000000 | ... | NaN | Southampton | no | False |
| 1 | 1 | 1 | female | 38.000000 | ... | C | Cherbourg | yes | False |
| 2 | 1 | 3 | female | 26.000000 | ... | NaN | Southampton | yes | True |
| 3 | 1 | 1 | female | 35.000000 | ... | C | Southampton | yes | False |
| 4 | 0 | 3 | male | 35.000000 | ... | NaN | Southampton | no | True |
| 5 | 0 | 3 | male | 29.699118 | ... | NaN | Queenstown | no | True |
| 6 | 0 | 1 | male | 54.000000 | ... | E | Southampton | no | True |
| 7 | 0 | 3 | male | 2.000000 | ... | NaN | Southampton | no | False |
| 8 | 1 | 3 | female | 27.000000 | ... | NaN | Southampton | yes | False |
| 9 | 1 | 2 | female | 14.000000 | ... | NaN | Cherbourg | yes | False |

[10 rows x 15 columns]

- 5번 행의 age 변수의 값을 살펴보면, NaN값이 age 변수의 유효 데이터의 평균값(29.699118)으로 변경된 것을 알 수 있음. 만약 평균 대신 중간값을 사용하려면 median() 함수를 사용하면 됨.

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('titanic')
In [3]: print(df[825:835])
```

| | survived | pclass | sex | age | ... | deck | embark_town | alive | alone |
|-----|----------|--------|--------|-------|-----|------|-------------|-------|-------|
| 825 | 0 | 3 | male | NaN | ... | NaN | Queenstown | no | True |
| 826 | 0 | 3 | male | NaN | ... | NaN | Southampton | no | True |
| 827 | 1 | 2 | male | 1.00 | ... | NaN | Cherbourg | yes | False |
| 828 | 1 | 3 | male | NaN | ... | NaN | Queenstown | yes | True |
| 829 | 1 | 1 | female | 62.00 | ... | B | NaN | yes | True |
| 830 | 1 | 3 | female | 15.00 | ... | NaN | Cherbourg | yes | False |
| 831 | 1 | 2 | male | 0.83 | ... | NaN | Southampton | yes | False |
| 832 | 0 | 3 | male | NaN | ... | NaN | Cherbourg | no | True |
| 833 | 0 | 3 | male | 23.00 | ... | NaN | Southampton | no | True |
| 834 | 0 | 3 | male | 18.00 | ... | NaN | Southampton | no | True |

[10 rows x 15 columns]

- 그리고 주어진 데이터셋에서 embark_town 변수는 범주형 자료이므로 embark_town 변수의 누락된 데이터를 value_counts() 함수와 idxmax() 함수를 이용하여 나머지 데이터의 최빈값으로 치환할 수 있음. 주어진 데이터셋을 출력하였을 때 829번 행에서 embark_town 변수의 값이 누락되어 있는 것을 확인할 수 있음.


```
In [4]: most_freq = df['embark_town'].value_counts(dropna=True).idxmax()
...: print(most_freq)
Southampton

In [5]: df['embark_town'].fillna(most_freq, inplace=True)

In [6]: print(df[825:835])
   survived  pclass  sex  age  ...  deck  embark_town  alive  alone
825         0      3  male  NaN  ...   NaN  Queenstown    no    True
826         0      3  male  NaN  ...   NaN  Southampton  no    True
827         1      2  male  1.00  ...   NaN  Cherbourg   yes  False
828         1      3  male  NaN  ...   NaN  Queenstown   yes    True
829         1      1  female 62.00  ...   B  Southampton   yes    True
830         1      3  female 15.00  ...   NaN  Cherbourg   yes  False
831         1      2  male   0.83  ...   NaN  Southampton   yes  False
832         0      3  male  NaN  ...   NaN  Cherbourg   no    True
833         0      3  male 23.00  ...   NaN  Southampton   no    True
834         0      3  male 18.00  ...   NaN  Southampton   no    True

[10 rows x 15 columns]
```

- NaN값이 embark_town 변수의 유효 데이터의 최빈값 (Southampton)으로 변경된 것을 알 수 있음.
- 서로 이웃하고 있는 데이터끼리 유사성을 가질 가능성이 있는 경우도 있습니다. 이러한 경우 fillna() 함수에 method 옵션을 통하여 앞이나 뒤에서 이웃하고 있는 값으로 치환해줄 수 있음.
- method 옵션에 'ffill'을 입력하면 NaN이 있는 행의 직전 행에 있는 값으로 변경하고, 'bfill'을 입력하면 NaN이 있는 행의 바로 다음 행에 있는 값으로 변경함.

데이터프레임객체.fillna(method='ffill'/'bfill', inplace=False/True)

```
In [1]: import seaborn as sns

In [2]: df = sns.load_dataset('titanic')

In [3]: print(df[825:835])
   survived  pclass  sex  age  ...  deck  embark_town  alive  alone
825         0      3  male  NaN  ...   NaN  Queenstown    no    True
826         0      3  male  NaN  ...   NaN  Southampton  no    True
827         1      2  male  1.00  ...   NaN  Cherbourg   yes  False
828         1      3  male  NaN  ...   NaN  Queenstown   yes    True
829         1      1  female 62.00  ...   B  NaN        yes    True
830         1      3  female 15.00  ...   NaN  Cherbourg   yes  False
831         1      2  male   0.83  ...   NaN  Southampton   yes  False
832         0      3  male  NaN  ...   NaN  Cherbourg   no    True
833         0      3  male 23.00  ...   NaN  Southampton   no    True
834         0      3  male 18.00  ...   NaN  Southampton   no    True

[10 rows x 15 columns]
```

- 주어진 데이터셋을 출력하였을 때 829번 행에서 embark_town 변수의 값이 누락되어 있는 것을 확인할 수 있음.

```
In [4]: df['embark_town'].fillna(method='ffill', inplace=True)

In [5]: print(df[825:835])
```

| | survived | pclass | sex | age | ... | deck | embark_town | alive | alone |
|-----|----------|--------|--------|-------|-----|------|-------------|-------|-------|
| 825 | 0 | 3 | male | NaN | ... | NaN | Queenstown | no | True |
| 826 | 0 | 3 | male | NaN | ... | NaN | Southampton | no | True |
| 827 | 1 | 2 | male | 1.00 | ... | NaN | Cherbourg | yes | False |
| 828 | 1 | 3 | male | NaN | ... | NaN | Queenstown | yes | True |
| 829 | 1 | 1 | female | 62.00 | ... | B | Queenstown | yes | True |
| 830 | 1 | 3 | female | 15.00 | ... | NaN | Cherbourg | yes | False |
| 831 | 1 | 2 | male | 0.83 | ... | NaN | Southampton | yes | False |
| 832 | 0 | 3 | male | NaN | ... | NaN | Cherbourg | no | True |
| 833 | 0 | 3 | male | 23.00 | ... | NaN | Southampton | no | True |
| 834 | 0 | 3 | male | 18.00 | ... | NaN | Southampton | no | True |

[10 rows x 15 columns]

- 주어진 데이터셋에서 embark_town 변수의 누락된 데이터를 method 옵션의 'ffill'을 입력하여 바로 앞에 위치한 행의 값으로 변경할 수 있음.
- NaN값이 embark_town 변수의 NaN 바로 앞에 위치한 유효 데이터의 값 (Queenstown)으로 변경된 것을 알 수 있음.

```
In [4]: df['embark_town'].fillna(method='bfill', inplace=True)

In [5]: print(df[825:835])
```

| | survived | pclass | sex | age | ... | deck | embark_town | alive | alone |
|-----|----------|--------|--------|-------|-----|------|-------------|-------|-------|
| 825 | 0 | 3 | male | NaN | ... | NaN | Queenstown | no | True |
| 826 | 0 | 3 | male | NaN | ... | NaN | Southampton | no | True |
| 827 | 1 | 2 | male | 1.00 | ... | NaN | Cherbourg | yes | False |
| 828 | 1 | 3 | male | NaN | ... | NaN | Queenstown | yes | True |
| 829 | 1 | 1 | female | 62.00 | ... | B | Cherbourg | yes | True |
| 830 | 1 | 3 | female | 15.00 | ... | NaN | Cherbourg | yes | False |
| 831 | 1 | 2 | male | 0.83 | ... | NaN | Southampton | yes | False |
| 832 | 0 | 3 | male | NaN | ... | NaN | Cherbourg | no | True |
| 833 | 0 | 3 | male | 23.00 | ... | NaN | Southampton | no | True |
| 834 | 0 | 3 | male | 18.00 | ... | NaN | Southampton | no | True |

[10 rows x 15 columns]

- 주어진 데이터셋에서 embark_town 변수의 누락된 데이터를 method 옵션의 'bfill'을 입력하여 바로 뒤에 위치한 행의 값으로 변경할 수 있음.
- NaN값이 embark_town 변수의 NaN 바로 뒤에 위치한 유효 데이터의 값 (Cherbourg)으로 변경된 것을 알 수 있음.

2. 중복 데이터 처리

○ 중복데이터 처리

- 동일한 대상이 중복으로 존재하게 되면 분석결과를 왜곡할 수 있음.
- 하나의 데이터셋에서 동일한 관측값이 2개 이상 중복되는 경우 중복 데이터를 찾아서 삭제해야 함.

1) 중복 데이터 확인

- 동일한 관측값이 중복되는지 여부를 확인하기 위해 duplicated() 함수를

사용하여 데이터의 중복 여부를 확인할 수 있음.

- duplicated() 함수를 사용하는 방법은 다음과 같음.

데이터프레임객체.duplicated(subset=기준열 리스트)

- 전에 나온 행들과 비교하여 중복되는 행이면 True를 반환하고, 처음 나오는 행에 대해서는 False를 반환함. 즉, duplicated() 함수를 적용하면 각 행의 중복 여부를 나타내는 불린 시리즈를 반환함.
- 0행의 데이터는 비교할 데이터가 없기 때문에 무조건 False로 판정함.

```
In [1]: import pandas as pd
In [2]: df_dict = {'c1': ['a', 'a', 'b', 'a', 'b'],
...:               'c2': [1, 1, 1, 2, 2],
...:               'c3': [1, 1, 2, 2, 2]}
In [3]: df = pd.DataFrame(df_dict)
...: print(df)
  c1  c2  c3
0  a   1   1
1  a   1   1
2  b   1   2
3  a   2   2
4  b   2   2
```

```
In [5]: df_dup = df.duplicated()
...: print(df_dup)
0    False
1     True
2    False
3    False
4    False
dtype: bool
```

- 주어진 데이터프레임에 duplicated() 함수를 적용하면 1행이 이전에 나온 행과 중복되므로 True가 출력되고, 중복되지 않은 나머지 열은 False가 출력된 것을 확인할 수 있음.

```
In [1]: import pandas as pd
In [2]: df_dict = {'c1': ['a', 'a', 'b', 'a', 'b'],
...:               'c2': [1, 1, 1, 2, 2],
...:               'c3': [1, 1, 2, 2, 2]}
In [3]: df = pd.DataFrame(df_dict)
...: print(df)
  c1  c2  c3
0  a   1   1
1  a   1   1
2  b   1   2
3  a   2   2
4  b   2   2
```

```
In [4]: df_dup = df.duplicated(subset=['c2', 'c3'])
...: print(df_dup)
0    False
1     True
2    False
3    False
4     True
dtype: bool
```

- subset 옵션을 이용하여 주어진 데이터프레임의 c2와 c3열에 duplicated() 함수를 적용하면 1행과 4행이 이전에 나온 행과 중복되므로 True가 출력되고, 중복되지 않은 나머지 열은 False가 출력된 것을 확인할 수 있음.

2) 중복 데이터 제거



- 동일한 관측값이 중복되었을 때 이를 제거하는 `drop_duplicates()` 함수를 사용하여 중복된 데이터를 제거할 수 있음.
- `drop_duplicates()` 함수를 사용하는 방법은 다음과 같음.

데이터프레임객체.drop_duplicates(subset=기준열 리스트, inplace=False/True)

- 원본 객체를 변경하려면 `inplace` 옵션을 통하여 수정할 수 있음.
- `subset` 옵션에 열 이름 리스트를 입력하여 중복 여부를 판별할 열을 지정할 수 있음. 만약 `subset` 옵션을 지정하지 않으면 모든 열에 대해서 중복 여부를 판별함.

```
In [1]: import pandas as pd
In [2]: df_dict = {'c1': ['a', 'a', 'b', 'a', 'b'],
...:               'c2': [1, 1, 1, 2, 2],
...:               'c3': [1, 1, 2, 2, 2]}
In [3]: df = pd.DataFrame(df_dict)
...: print(df)
   c1  c2  c3
0  a   1   1
1  a   1   1
2  b   1   2
3  a   2   2
4  b   2   2
```

```
In [4]: df2 = df.drop_duplicates()
...: print(df2)
   c1  c2  c3
0  a   1   1
2  b   1   2
3  a   2   2
4  b   2   2
```

- 주어진 데이터프레임에 `drop_duplicates()` 함수를 적용하면 1행이 이전에 나온 행과 중복되므로 삭제된 것을 확인할 수 있음.

```
In [1]: import pandas as pd
In [2]: df_dict = {'c1': ['a', 'a', 'b', 'a', 'b'],
...:               'c2': [1, 1, 1, 2, 2],
...:               'c3': [1, 1, 2, 2, 2]}
In [3]: df = pd.DataFrame(df_dict)
...: print(df)
   c1  c2  c3
0  a   1   1
1  a   1   1
2  b   1   2
3  a   2   2
4  b   2   2
```

```
In [4]: df3 = df.drop_duplicates(subset=['c2', 'c3'])
...: print(df3)
   c1  c2  c3
0  a   1   1
2  b   1   2
3  a   2   2
```

- `subset` 옵션을 이용하여 주어진 데이터프레임의 `c2`와 `c3`열에 `drop_duplicates()` 함수를 적용하면 1행과 4행이 이전에 나온 행과 중복되므로 삭제된 것을 확인할 수 있음.

3. 누락 및 중복 데이터 처리 실습

※ seaborn 모듈의 penguins 데이터를 가져온 후, 데이터의 유효데이터 및 자료유형을 확인해보세요.

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('penguins')
...: print(df)
0      species      island  bill_length_mm  ...  flipper_length_mm  body_mass_g  sex
1  Adelie  Torgersen    39.1  ...      181.0      3750.0  Male
2  Adelie  Torgersen    39.5  ...      186.0      3800.0  Female
3  Adelie  Torgersen    40.3  ...      195.0      3250.0  Female
4  Adelie  Torgersen    NaN  ...      NaN      NaN      NaN
5  Adelie  Torgersen    36.7  ...      193.0      3450.0  Female
...
339  Gentoo  Biscoe    NaN  ...      NaN      NaN      NaN
340  Gentoo  Biscoe    46.8  ...      215.0      4850.0  Female
341  Gentoo  Biscoe    50.4  ...      222.0      5750.0  Male
342  Gentoo  Biscoe    45.2  ...      212.0      5200.0  Female
343  Gentoo  Biscoe    49.9  ...      213.0      5400.0  Male
[344 rows x 7 columns]
```

```
In [3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   species                344 non-null   object
1   island                 344 non-null   object
2   bill_length_mm         342 non-null   float64
3   bill_depth_mm          342 non-null   float64
4   flipper_length_mm       342 non-null   float64
5   body_mass_g            342 non-null   float64
6   sex                    333 non-null   object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

- info() 함수를 통해 penguins 데이터셋의 변수명, 유효 데이터 개수 및 자료의 유형을 확인할 수 있습니다.

※ penguins 데이터에서 성별(sex) 변수의 요인별 유효데이터 개수와 누락 데이터의 개수를 확인해보세요.

```
In [4]: nan_sex = df['sex'].value_counts(dropna=False)
...: print(nan_sex)
Male      168
Female    165
NaN         11
Name: sex, dtype: int64
```

- value_counts() 함수를 통해 penguins 데이터셋의 성별 변수에는 Male 이 168개, Female이 165개 유효한 것을 알 수 있고, 11개의 누락자료가 있는 것을 확인할 수 있습니다.

※ penguins 데이터에서 유효데이터의 수가 340개 이하인 변수는 제거해보세요.

```
In [3]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   species                344 non-null   object
1   island                 344 non-null   object
2   bill_length_mm         342 non-null   float64
3   bill_depth_mm          342 non-null   float64
4   flipper_length_mm       342 non-null   float64
5   body_mass_g            342 non-null   float64
6   sex                    333 non-null   object
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

```
In [5]: df_thresh = df.dropna(axis=1, thresh=340)
In [6]: df_thresh.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 6 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   species                344 non-null   object
1   island                 344 non-null   object
2   bill_length_mm         342 non-null   float64
3   bill_depth_mm          342 non-null   float64
4   flipper_length_mm       342 non-null   float64
5   body_mass_g            342 non-null   float64
dtypes: float64(4), object(2)
memory usage: 16.2+ KB
```

- dropna() 함수의 thresh 옵션을 통해 penguins 데이터셋에서 유효한 데이터의 수가 340개 이하인 변수가 제거되는 것을 확인할 수 있습니다.

여기서는 성별 변수가 제거된 것을 확인할 수 있습니다.

※ penguins 데이터에서 성별(sex) 변수에서 누락된 레코드를 모두 제거해 보세요.

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('penguins')
...: print(df)
species    island  bill_length_mm  ... flipper_length_mm  body_mass_g  sex
0  Adelie  Torgersen      39.1  ...      181.0      3750.0  Male
1  Adelie  Torgersen      39.5  ...      186.0      3800.0  Female
2  Adelie  Torgersen      40.3  ...      195.0      3250.0  Female
3  Adelie  Torgersen      NaN  ...      NaN      NaN      NaN
4  Adelie  Torgersen      36.7  ...      193.0      3450.0  Female
...  ...  ...  ...  ...  ...  ...
339 Gentoo  Biscoe      NaN  ...      NaN      NaN      NaN
340 Gentoo  Biscoe      46.8  ...      215.0      4850.0  Female
341 Gentoo  Biscoe      50.4  ...      222.0      5750.0  Male
342 Gentoo  Biscoe      45.2  ...      212.0      5200.0  Female
343 Gentoo  Biscoe      49.9  ...      213.0      5400.0  Male
[344 rows x 7 columns]
```

```
In [7]: df_sex = df.dropna(subset=['sex'], how='any', axis=0)
In [8]: print(df_sex)
species    island  bill_length_mm  ... flipper_length_mm  body_mass_g  sex
0  Adelie  Torgersen      39.1  ...      181.0      3750.0  Male
1  Adelie  Torgersen      39.5  ...      186.0      3800.0  Female
2  Adelie  Torgersen      40.3  ...      195.0      3250.0  Female
4  Adelie  Torgersen      36.7  ...      193.0      3450.0  Female
5  Adelie  Torgersen      39.3  ...      190.0      3650.0  Male
...  ...  ...  ...  ...  ...
338 Gentoo  Biscoe      47.2  ...      214.0      4925.0  Female
340 Gentoo  Biscoe      46.8  ...      215.0      4850.0  Female
341 Gentoo  Biscoe      50.4  ...      222.0      5750.0  Male
342 Gentoo  Biscoe      45.2  ...      212.0      5200.0  Female
343 Gentoo  Biscoe      49.9  ...      213.0      5400.0  Male
[333 rows x 7 columns]
```

- dropna() 함수의 subset 옵션을 통해 성별 변수에서 3번, 339번 행과 같은 누락된 레코드가 제거된 것을 확인할 수 있습니다.

※ penguins 데이터에서 체중(body_mass_g) 변수에서 누락된 레코드를 유효한 데이터의 평균으로 치환해 보세요.

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('penguins')
...: print(df)
species    island  bill_length_mm  ... flipper_length_mm  body_mass_g  sex
0  Adelie  Torgersen      39.1  ...      181.0      3750.0  Male
1  Adelie  Torgersen      39.5  ...      186.0      3800.0  Female
2  Adelie  Torgersen      40.3  ...      195.0      3250.0  Female
3  Adelie  Torgersen      NaN  ...      NaN      NaN      NaN
4  Adelie  Torgersen      36.7  ...      193.0      3450.0  Female
...  ...  ...  ...  ...  ...
339 Gentoo  Biscoe      NaN  ...      NaN      NaN      NaN
340 Gentoo  Biscoe      46.8  ...      215.0      4850.0  Female
341 Gentoo  Biscoe      50.4  ...      222.0      5750.0  Male
342 Gentoo  Biscoe      45.2  ...      212.0      5200.0  Female
343 Gentoo  Biscoe      49.9  ...      213.0      5400.0  Male
[344 rows x 7 columns]
```

```
In [9]: mean_body = df['body_mass_g'].mean(axis=0)
...: df['body_mass_g'].fillna(mean_body, inplace=True)
In [10]: print(df)
species    island  bill_length_mm  ... flipper_length_mm  body_mass_g  sex
0  Adelie  Torgersen      39.1  ...      181.0  3750.000000  Male
1  Adelie  Torgersen      39.5  ...      186.0  3800.000000  Female
2  Adelie  Torgersen      40.3  ...      195.0  3250.000000  Female
3  Adelie  Torgersen      NaN  ...      NaN  4201.754386  NaN
4  Adelie  Torgersen      36.7  ...      193.0  3450.000000  Female
...  ...  ...  ...  ...  ...
339 Gentoo  Biscoe      NaN  ...      NaN  4201.754386  NaN
340 Gentoo  Biscoe      46.8  ...      215.0  4850.000000  Female
341 Gentoo  Biscoe      50.4  ...      222.0  5750.000000  Male
342 Gentoo  Biscoe      45.2  ...      212.0  5200.000000  Female
343 Gentoo  Biscoe      49.9  ...      213.0  5400.000000  Male
[344 rows x 7 columns]
```

- mean() 함수를 통해 유효한 데이터의 체중 변수 평균을 구할 수 있고, fillna() 함수를 통해 3번, 339번 행과 같은 누락된 데이터가 평균값으로 치환되는 것을 확인할 수 있습니다.

※ penguins 데이터에서 성별(sex) 변수에서 누락된 레코드를 유효한 데이터의 최빈값으로 치환해 보세요.

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('penguins')
...: print(df)
species    island  bill_length_mm  ... flipper_length_mm  body_mass_g  sex
0  Adelie  Torgersen      39.1  ...      181.0      3750.0  Male
1  Adelie  Torgersen      39.5  ...      186.0      3800.0  Female
2  Adelie  Torgersen      40.3  ...      195.0      3250.0  Female
3  Adelie  Torgersen      NaN  ...      NaN      NaN      NaN
4  Adelie  Torgersen      36.7  ...      193.0      3450.0  Female
...  ...  ...  ...  ...  ...
339 Gentoo  Biscoe      NaN  ...      NaN      NaN      NaN
340 Gentoo  Biscoe      46.8  ...      215.0      4850.0  Female
341 Gentoo  Biscoe      50.4  ...      222.0      5750.0  Male
342 Gentoo  Biscoe      45.2  ...      212.0      5200.0  Female
343 Gentoo  Biscoe      49.9  ...      213.0      5400.0  Male
[344 rows x 7 columns]
```

```
In [14]: most_freq = df['sex'].value_counts(dropna=True).idxmax()
...: print(most_freq)
Male
In [15]: df['sex'].fillna(most_freq, inplace=True)
...: print(df)
species    island  bill_length_mm  ... flipper_length_mm  body_mass_g  sex
0  Adelie  Torgersen      39.1  ...      181.0      3750.0  Male
1  Adelie  Torgersen      39.5  ...      186.0      3800.0  Female
2  Adelie  Torgersen      40.3  ...      195.0      3250.0  Female
3  Adelie  Torgersen      NaN  ...      NaN      NaN      Male
4  Adelie  Torgersen      36.7  ...      193.0      3450.0  Female
...  ...  ...  ...  ...  ...
339 Gentoo  Biscoe      NaN  ...      NaN      NaN      Male
340 Gentoo  Biscoe      46.8  ...      215.0      4850.0  Female
341 Gentoo  Biscoe      50.4  ...      222.0      5750.0  Male
342 Gentoo  Biscoe      45.2  ...      212.0      5200.0  Female
343 Gentoo  Biscoe      49.9  ...      213.0      5400.0  Male
[344 rows x 7 columns]
```

- value_counts() 및 idxmax() 함수를 통해 유효한 데이터의 성별 변수

최빈값을 구할 수 있고, fillna() 함수를 통해 3번, 339번 행과 같은 누락된 데이터가 최빈값으로 치환되는 것을 확인할 수 있습니다.

※ penguins 데이터에서 성별(sex) 변수에서 누락된 레코드를 바로 앞의 유효한 값으로 치환해 보세요.

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('penguins')
...: print(df)
species island bill_length_mm ... flipper_length_mm body_mass_g sex
0 Adelle Torgersen 39.1 ... 181.0 3750.0 Male
1 Adelle Torgersen 39.5 ... 186.0 3800.0 Female
2 Adelle Torgersen 40.3 ... 195.0 3250.0 Female
3 Adelle Torgersen NaN ... NaN NaN
4 Adelle Torgersen 36.7 ... 193.0 3450.0 Female
... ..
339 Gentoo Biscoe NaN ... NaN NaN
340 Gentoo Biscoe 46.8 ... 215.0 4850.0 Female
341 Gentoo Biscoe 50.4 ... 222.0 5750.0 Male
342 Gentoo Biscoe 45.2 ... 212.0 5200.0 Female
343 Gentoo Biscoe 49.9 ... 213.0 5400.0 Male
[344 rows x 7 columns]
```

```
In [10]: df['sex'].fillna(method='ffill', inplace=True)
...: print(df)
species island bill_length_mm ... flipper_length_mm body_mass_g sex
0 Adelle Torgersen 39.1 ... 181.0 3750.0 Male
1 Adelle Torgersen 39.5 ... 186.0 3800.0 Female
2 Adelle Torgersen 40.3 ... 195.0 3250.0 Female
3 Adelle Torgersen NaN ... NaN NaN
4 Adelle Torgersen 36.7 ... 193.0 3450.0 Female
... ..
339 Gentoo Biscoe NaN ... NaN NaN
340 Gentoo Biscoe 46.8 ... 215.0 4850.0 Female
341 Gentoo Biscoe 50.4 ... 222.0 5750.0 Male
342 Gentoo Biscoe 45.2 ... 212.0 5200.0 Female
343 Gentoo Biscoe 49.9 ... 213.0 5400.0 Male
[344 rows x 7 columns]
```

- fillna() 함수의 method 옵션을 통해 3번, 339번 행과 같은 누락된 데이터가 바로 앞의 유효한 값으로 치환되는 것을 확인할 수 있습니다.

※ penguins 데이터에서 species와 island를 기준으로 중복되는 데이터를 제거해보세요.

```
In [1]: import seaborn as sns
In [2]: df = sns.load_dataset('penguins')
...: print(df)
species island bill_length_mm ... flipper_length_mm body_mass_g sex
0 Adelle Torgersen 39.1 ... 181.0 3750.0 Male
1 Adelle Torgersen 39.5 ... 186.0 3800.0 Female
2 Adelle Torgersen 40.3 ... 195.0 3250.0 Female
3 Adelle Torgersen NaN ... NaN NaN
4 Adelle Torgersen 36.7 ... 193.0 3450.0 Female
... ..
339 Gentoo Biscoe NaN ... NaN NaN
340 Gentoo Biscoe 46.8 ... 215.0 4850.0 Female
341 Gentoo Biscoe 50.4 ... 222.0 5750.0 Male
342 Gentoo Biscoe 45.2 ... 212.0 5200.0 Female
343 Gentoo Biscoe 49.9 ... 213.0 5400.0 Male
[344 rows x 7 columns]
```

```
In [3]: df_dup = df.drop_duplicates(subset=['species', 'island'])
In [4]: print(df_dup)
species island ... body_mass_g sex
0 Adelle Torgersen ... 3750.0 Male
20 Adelle Biscoe ... 3400.0 Female
30 Adelle Dream ... 3250.0 Female
152 Chinstrap Dream ... 3500.0 Female
220 Gentoo Biscoe ... 4500.0 Female
[5 rows x 7 columns]
```

- drop_duplicates() 함수의 subset 옵션을 통해 species와 island 변수를 기준으로 중복되는 데이터가 제거된 것을 확인할 수 있습니다.