

# 결측치(missing values) 처리

seaborn은 파이썬의 데이터 시각화 라이브러리 중 하나이다.

- **anscombe**: Anscombe의 사분면 그림 데이터셋으로, 데이터 분석의 중요성을 보여주는 네 가지 서로 다른 데이터셋이 포함되어 있습니다.
- **iris**: 붓꽃 데이터셋으로, 꽃잎과 꽃받침의 길이 및 너비에 대한 측정값을 포함합니다.
- **tips**: 레스토랑에서의 식사 정보를 포함하는 데이터셋으로, 식사 금액, 팁, 성별, 흡연 여부 등이 포함됩니다.
- **titanic**: 데이터셋은 타이타닉호에서 탑승객들의 정보와 생존 여부를 포함한 데이터셋입니다.

- **survived**: 생존 여부를 나타내는 이진 변수 (0: 사망, 1: 생존)
- **pclass**: 승객의 좌석 등급 (1, 2, 3)
- **sex**: 성별 (male: 남성, female: 여성)
- **age**: 나이
- **sibsp**: 함께 탑승한 형제자매/배우자의 수
- **parch**: 함께 탑승한 부모/자녀의 수
- **fare**: 요금
- **embarked**: 탑승 항구 (C: Cherbourg, Q: Queenstown, S: Southampton) class:
- 승객의 좌석 등급 (First, Second, Third)
- **who**: 승객의 성별과 나이를 결합한 열 (man, woman, child)
- **adult\_male**: 성인 남성 여부 (True, False)
- **deck**: 선실 번호
- **embark\_town**: 탑승 항구의 도시 이름
- **alive**: 생존 여부 (yes, no)
- **alone**: 혼자 여행한 여부 (True, False)

seaborn 라이브러리의 titanic 데이터 셋 로드하기

```
import seaborn as sns
df=sns.load_dataset('titanic') # 타이타닉 데이터 로드하기
df
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who
0	0	3	male	22.0	1	0	7.2500	S	Third	man
1	1	1	female	38.0	1	0	71.2833	C	First	woman
2	1	3	female	26.0	0	0	7.9250	S	Third	woman
3	1	1	female	35.0	1	0	53.1000	S	First	woman
4	0	3	male	35.0	0	0	8.0500	S	Third	man
...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man
887	1	1	female	19.0	0	0	30.0000	S	First	woman
888	0	3	female	NaN	1	2	23.4500	S	Third	woman
889	1	1	male	26.0	0	0	30.0000	C	First	man
890	0	3	male	32.0	0	0	7.7500	Q	Third	man

891 rows × 15 columns

데이터 셋 정보확인

```
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
```

```
df.describe()
```

value\_counts()로 who 열을 구성하는 각 값들의 개수 출력

```
df['deck'].value_counts(dropna=False) # 누락 데이터 개수까지 확인시 dropna=False 사용할 것.
```

## 데이터프레임에서 각 셀이 결측치인지 확인하기

- `df.isnull()` # `df.isna()`와 동일

[illegible]

889	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	False

891 rows × 15 columns

각 열의 결측치 개수 확인

```
df.isnull().sum()
```

survived	0
pclass	0
sex	0
age	177
sibsp	0
parch	0
fare	0
embarked	2
class	0
who	0
adult_male	0
deck	688
embark_town	2
alive	0
alone	0
dtype:	int64

특정 열의 결측치 개수

```
df['deck'].isnull().sum() # df['deck'].notnull().sum() : 결측치가 아닌 개수
688
```

결측치 변경

- df.fillna(값)
- df.fillna(method='ffill') # f : foreward, 바로 앞 행의 값으로 채우기
- df.fillna(method='bfill') # b : backward, 바로 다음 행의 값으로 채우기

결측치를 모두 0으로 채우기

```
df_NaN=df
df_NaN['age'].fillna(0,inplace=True)
df_NaN
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who
0	0	3	male	22.0	1	0	7.2500	S	Third	man
1	1	1	female	38.0	1	0	71.2833	C	First	woman
2	1	3	female	26.0	0	0	7.9250	S	Third	woman
3	1	1	female	35.0	1	0	53.1000	S	First	woman
4	0	3	male	35.0	0	0	8.0500	S	Third	man
...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man
887	1	1	female	19.0	0	0	30.0000	S	First	woman
888	0	3	female	0.0	1	2	23.4500	S	Third	woman
889	1	1	male	26.0	0	0	30.0000	C	First	man
890	0	3	male	32.0	0	0	7.7500	Q	Third	man

891 rows × 15 columns

결측치를 이웃 행의 값으로 치환

- df.fillna(method='ffill')
- df.fillna(method='bfill')

```
df_NaN['deck'].fillna(method='bfill',inplace=True) # inplace=True 는 데이터프레임 자체에 적용
df_NaN
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who
0	0	3	male	22.0	1	0	7.2500	S	Third	man
1	1	1	female	38.0	1	0	71.2833	C	First	woman
2	1	3	female	26.0	0	0	7.9250	S	Third	woman
3	1	1	female	35.0	1	0	53.1000	S	First	woman
4	0	3	male	35.0	0	0	8.0500	S	Third	man
...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man
887	1	1	female	19.0	0	0	30.0000	S	First	woman
888	0	3	female	0.0	1	2	23.4500	S	Third	woman
889	1	1	male	26.0	0	0	30.0000	C	First	man
890	0	3	male	32.0	0	0	7.7500	Q	Third	man

891 rows x 15 columns

결측치를 승객이 가장 많이 승선한 도시이름으로 치환.

```
df1=df.copy()
df1['embark_town'][825:830]
```

```
825    Queenstown
826    Southampton
827     Cherbourg
828    Queenstown
829         NaN
Name: embark_town, dtype: object
```

# embark\_town열에서 결측치가 있는 행을 제외한 빈도수가 가장 높은 값의 항목명

```
most_freq=df1['embark_town'].value_counts(dropna=True).idxmax()
most_freq
```

'Southampton'

# 결측치를 승객이 가장 많이 승선한 도시이름으로 치환

```
df1['embark_town'].fillna(most_freq,inplace=True)
df1['embark_town'][825:830]
```

```
825    Queenstown
826    Southampton
827     Cherbourg
828    Queenstown
829    Southampton
Name: embark_town, dtype: object
```

## 결측치가 있는 행 제거하기.

```
# 기존 데이터프레임을 복사
```

```
df2=df
```

```
# embarked 열에서 결측치가 있는 행을 삭제
```

```
df2.dropna(subset=['embarked'], inplace=True)
```

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 15 columns):
#   Column             Non-Null Count  Dtype
---  -
0   survived           889 non-null    int64
1   pclass             889 non-null    int64
2   sex                889 non-null    object
3   age                712 non-null    float64
4   sibsp              889 non-null    int64
5   parch             889 non-null    int64
6   fare               889 non-null    float64
7   embarked           889 non-null    object
8   class              889 non-null    category
9   who                889 non-null    object
10  adult_male         889 non-null    bool
11  deck               201 non-null    category
12  embark_town        889 non-null    object
13  alive              889 non-null    object
14  alone              889 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 87.3+ KB
```

```
# 열에 결측치를 제외한 것이 500개 이상인 열만 남기기
```

```
# 데이터셋에서 deck열의 NaN 비율이 높아서 분석에서 제외하는 것이 의미가 있다. Thresh(임계치)를 사용.
```

```
df2.dropna(axis=1, thresh=500, inplace=True)
```

```
df2.info()
```