데이터 다루기 (3)

그룹별 분석과 조건부 필터링

GOAL

- 조건부 필터링으로 특정 데이터 비교
- 그룹별 통계 요약 (groupby)
- 범주형 vs 숫자형 시각화 차이 (barplot vs histogram/boxplot)

학습 목표

- 조건에 따라 데이터를 필터링할 수 있다.
- 그룹별 요약 통계를 구할 수 있다.
- **그룹 정보를 이용해 결측치를 더 현실적으로 채울 수 있다.**

왜 평균으로만 채우면 문제가 될까?

- 전 승객의 평균 나이로 채우면,
  1등석/3등석, 남성/여성의 **차이**가 사라진다.
- 비유: 모든 반 학생의 결석 점수를 **전교 평균**으로 넣는 것과 같다.
  학급 특성이 지워진다.

→ **상황(그룹)별**로 채우면 현실을 더 잘 반영한다.

타이타닉 데이터

```
import seaborn as sns
import pandas as pd

df = sns.load_dataset("titanic")
df.head()
df.info()
```

주요 컬럼

- 수치형: `age`, `fare`, `sibsp`, `parch`
- 범주형: `sex`, `class`, `embarked`, `embark_town`, `who`, `deck`, `alive`
- 타깃(생존): `survived`

결측치 현황 빠르게 보기

```
(df.isnull()
   .mean()
   .sort_values(ascending=False)
   .to_frame("null_ratio"))
```

보통 `age`, `deck`, `embark_town`에 결측이 많다.

→ 오늘은 **그룹별 로직**으로 이들을 채워본다.

조건부 필터링

```
# 남성만
df_male = df[df["sex"] == "male"]

# 1등석 여성만
df_f1 = df[(df["sex"] == "female") & (df["class"] == "First")]

# 요금이 50 초과 & 3등석
df_rich3 = df[(df["fare"] > 50) & (df["class"] == "Third")]
```

groupby

- “데이터를 그룹별로 요약한다”
- 같은 값을 가진 행끼리 묶어서(그룹핑해서), 평균·합계·개수 등을 계산하는 함수.

groupby: 단일 그룹

```
# 성별 평균 나이
df.groupby("sex")["age"].mean()

# 선실등급별 중앙 요금
df.groupby("class")["fare"].median()
```

한 줄로 그룹별 요약 통계를 쉽게 산출한다.

groupby: 다중 그룹

```
# 성별 x 등급별 평균 나이
df.groupby(["sex", "class"])["age"].mean().unstack()
```

- `unstack()`으로 표 형태로 보기 쉬워진다.
- 다층 그룹은 **현실의 맥락**을 더 잘 반영한다.

주요 메서드

메서드	설명
`.sum()`	합계
`.mean()`	평균
`.median()`	중앙값
`.count()`	개수
`.min()` / `.max()`	최소 / 최대값
`.std()` / `.var()`	표준편차 / 분산
`.size()`	각 그룹의 크기
`.first()` / `.last()`	첫 번째 / 마지막 값
`.nunique()`	고유한 값의 개수

pivot_table

```
pd.pivot_table(
    df, values="age", index="sex", columns="class",
    aggfunc="mean"
)
```

- `pivot_table`은 한 번에 **집계 + 피벗**
- 익숙한 엑셀 피벗과 유사

pivot_table 인자

인자	의미	예시 / 설명
`data`	피벗할 원본 데이터프레임	여기서는 `df`
`values`	요약하고 싶은 "값"이 들어있는 열	`"age"` → 나이 평균을 계산함
`index`	행으로 그룹화할 기준	`"sex"` → 남/여별로 행 구분
`columns`	열로 그룹화할 기준	`"class"` → 1등석, 2등석, 3등석 등으로 열 구분
`aggfunc`	어떤 방식으로 요약할지(집계 함수)	`"mean"` → 평균, `"sum"` → 합계, `"count"` → 개수 가능

시각화: 범주형 vs 숫자형

- 범주형 → **barplot, countplot**
- 숫자형 → **histogram, boxplot, kdeplot**

```
import seaborn as sns
sns.barplot(data=df, x="class", y="age")          # 그룹별 평균
sns.boxplot(data=df, x="class", y="age")          # 분포+이상치
```

결측치 채우기 전략 개요

1. **그룹 중심 통계**로 채우기
    - 예: `age` → (성별, 선실등급) 그룹 **중앙값**
2. **범주형 모드**로 채우기
    - 예: `embark_town` → 선실등급별 **최빈값**
3. **Unknown(미상)** 카테고리 생성
    - 예: `deck` → 정보 자체가 모호할 때 안전한 선택

crosstab

- 두(또는 그 이상) 범주형 변수의 교차 빈도표를 만드는 함수
- 언제 쓰나? → “반별 성별 분포는?”, “등급별 출항지는 어디가 많나?”처럼 범주 vs 범주 관계를 한눈에 보고 싶을 때

corsstab

```
# 클래스 x 출항지 빈도표
pd.crosstab(df["class"], df["embark_town"])
# 비율로도 보기
pd.crosstab(df["class"], df["embark_town"], normalize="index").round(2)
```

Step 1) Age: 성별×등급별 중앙값으로 채우기

왜 중앙값?

- 나이는 한쪽 꼬리가 긴 분포가 흔함 → 평균보다 중앙값이 안정적.

```
df_imp = df.copy()

# 그룹별 중앙값 계산(브로드캐스트용 transform)
age_median = (df_imp
              .groupby(["sex", "class"])["age"]
              .transform("median"))

# age 결측만 그룹 중앙값으로 채우기
df_imp["age"] = df_imp["age"].fillna(age_median)
```

Step 1) 전/후 비교

```
df[["age"]].describe()
df_imp[["age"]].describe()

# 시각화 비교
sns.kdeplot(df["age"], label="orig", fill=True)
sns.kdeplot(df_imp["age"], label="imputed", fill=True)
```

- 분포의 중심이 **그룹 맥락**을 반영해 이동/안정화되는지 확인

Step 2) embark_town: class등급 별로 높은 비중 채우기

왜 최빈값?

- 범주형 변수는 평균이 의미 없음 → **가장 자주 등장**하는 값을 사용.

```
# 1등석에서 비어 있으면 'Cherbourg'로
mask = (df_imp["class"] == "First") & (df_imp["embark_town"].isna())
df_imp.loc[mask, "embark_town"] = "Cherbourg"

# 2등석에서 비어 있으면 'Southampton'으로
mask = (df_imp["class"] == "Second") & (df_imp["embark_town"].isna())
df_imp.loc[mask, "embark_town"] = "Southampton"

# 3등석에서 비어 있으면 'Southampton'으로
mask = (df_imp["class"] == "Third") & (df_imp["embark_town"].isna())
df_imp.loc[mask, "embark_town"] = "Southampton"
```

Step 3) deck: Unknown 카테고리로 안전하게

- `deck`는 결측 비율이 매우 높고, 추정이 불확실하다.
- 잘못된 추정은 **오히려 모델 성능을 해침**
  → 명시적으로 `"Unknown"`을 부여하여 정보 손실을 드러낸다.

```
df_imp["deck"] = df_imp["deck"].astype("category")
df_imp["deck"] = df_imp["deck"].cat.add_categories(["Unknown"])
df_imp["deck"] = df_imp["deck"].fillna("Unknown")
```

검증: 결측률 다시 확인

```
before = df.isnull().mean().to_frame("null_ratio_before")
after  = df_imp.isnull().mean().to_frame("null_ratio_after")

before.join(after, how="outer").sort_values("null_ratio_after", ascending=False)
```

- 목적 컬럼의 결측이 의도대로 줄었는가?
- **덜 중요한 변수**는 과감히 Unknown 처리했는가?

시각화 비교: 그룹별 분포 유지 여부

```
sns.boxplot(data=df, x="class", y="age")
sns.boxplot(data=df_imp, x="class", y="age")
```

- 그룹별 분포 패턴이 **합리적**으로 유지되는지 육안 점검
- 결측치 채우기 전후로 **왜곡**이 심해지지 않았는지 확인

조건부 필터링으로 현상 탐색

```
# 1등석만 놓고 age 분포 비교
sns.histplot(df_imp[df_imp["class"]=="First"]["age"], kde=True)

# 남성/여성, 성인/아동 등 다양한 조건으로 슬라이싱
sns.boxplot(data=df_imp[df_imp["sex"]=="female"], x="class", y="age")
```

- “누구에게서 결측이 많았나?”를 조건별로 확인하여
  편향된 보정이 일어나지 않도록 한다.

실습 A: 나만의 그룹 정의로 age 채우기

- 제안: `(who, class)` 또는 `(sex, embark_town)`
- 나이 분포가 더 **안정적인 조합**을 스스로 찾아본다.

```
group_cols = ["who", "class"]
age_med2 = df.groupby(group_cols)["age"].transform("median")
age_new = df["age"].fillna(age_med2)
```

실습 B: fare 결측이 있다면?

- 보통 `fare`는 결측이 적지만, 있을 경우
  `(class, embark_town)` 조합의 **중앙값** 추천

```
fare_med = df.groupby(["class","embark_town"])["fare"].transform("median")
df["fare"] = df["fare"].fillna(fare_med)
```

실습 C: 다단계 보정 흐름 만들기

1. 우선순위 높은 그룹 기준 채움
2. 여전히 남는 결측은 상위 레벨 그룹으로 채움
3. 그래도 남으면 전역 중앙값/Unknown 처리

```
tmp = df.copy()

# 1차: sex x class
g1 = tmp.groupby(["sex","class"])["age"].transform("median")
tmp["age"] = tmp["age"].fillna(g1)

# 2차: class만
g2 = tmp.groupby("class")["age"].transform("median")
tmp["age"] = tmp["age"].fillna(g2)

# 3차: 전역 중앙값
tmp["age"] = tmp["age"].fillna(tmp["age"].median())
```

요약

- 조건부 필터링으로 **대상 그룹**을 정확히 규정한다.
- `groupby` / `pivot_table`로 **그룹별 통계**를 파악한다.
- 결측치는 **그룹별 대표값**(중앙값/모드)으로 채운다.
- 불확실한 범주형은 **Unknown 카테고리**로 보존한다.
- 시각화로 **전후 분포**와 **그룹 비율**을 점검한다.

참고 코드 스니펫 묶음

```
# 조건부 필터링
df[(df["sex"]=="female") & (df["class"]=="First")]

# 그룹 요약
df.groupby(["sex","class"])["age"].agg(["count","mean","median"])

# 피벗/크로스탭
pd.pivot_table(df, values="age", index="sex", columns="class", aggfunc="median")
pd.crosstab(df["class"], df["embark_town"], normalize="index")

# 시각화
sns.barplot(data=df, x="class", y="age", estimator=pd.Series.median, hue="sex")
sns.boxplot(data=df, x="class", y="age")
sns.histplot(df["age"], kde=True)
```