이변량/다변량 분석 (Bivariate/Multivariate Analysis)

- **이변량 분석 (Bivariate Analysis)**: 두 개의 변수 간의 관계를 분석합니다.

x (featrue) \ y (target)	수치형	범주형
수치형	**시각화** ・Scatter(산점도) **수치화** ・상관분석	**시각화** ・Boxplot ・Histogram ・Density plot
범주형	**시각화** ・평균비교 barplot (sns.barplot) ・Boxplot **수치화** ・T 검정 ・ANOVA	**시각화** ・Barplot ・Mosaic **수치화** ・교차표 ・카이 제곱 검정

- **다변량 분석 (Multivariate Analysis)**: 세 개 이상의 변수 간의 관계를 동시에 분석합니다. 이변량 분석을 확장하여, `hue`, `size`, `style` 등의 옵션을 추가하여 제3, 제4의 변수를 시각화에 포함시키는 방식으로 주로 수행됩니다. (e.g., 산점도에 색상과 크기로 추가 변수 표현)

적용 가능한 상황

- **가설 검증**: '객실 등급이 높을수록 생존율이 높을 것이다'(범주형 vs 범주형), '나이가 많을수록 운임 요금을 더 많이 냈을 것이다'(수치형 vs 수치형)와 같은 가설을 데이터로 확인할 때.
- **피처 엔지니어링**: 변수 간의 강한 상관관계를 확인하여 다중공선성 문제를 인지하거나, 변수들의 상호작용을 나타내는 새로운 파생 변수를 만들 아이디어를 얻을 때.
- **모델링 변수 선택**: 타겟 변수(종속 변수)와 높은 상관관계를 갖는 독립 변수를 찾아 모델의 예측 변수로 사용하거나, 변수 간의 관계를 파악하여 모델링 전략을 수립할 때.

예제 데이터 생성

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

# Seaborn 내장 'titanic' 데이터셋 사용
df = sns.load_dataset('titanic')
```

수치형 → 수치형

시각화 예시

```
# 산점도 (Scatter Plot)
sns.scatterplot(x='age', y='fare', data=df)
plt.title('Age vs Fare')
plt.show()

# 산점도 + 히스토그램
sns.jointplot(x='age', y='fare', data = df)
plt.title('Age vs Fare by jointplot')
plt.show()

# 산점도 + 직선관계 + 95% 신뢰구간
sns.regplot(x='age', y='fare', data = df)
plt.title('Age vs Fare by regplot')
plt.show()

# 다변량 분석: 산점도에 'survived' 변수 추가 (hue)
sns.scatterplot(x='age', y='fare', hue='survived', data=df)
plt.title('Age vs Fare by Survival')
plt.show()
```

- 산점도를 보면 age와 fare 사이에 뚜렷한 선형 관계는 보이지 않습니다.
- hue 옵션을 추가한 다변량 산점도를 통해 생존 여부에 따른 분포 차이를 탐색해볼 수 있습니다.

수치화 예시

- 상관관계를 나타내는 2가지 숫자 : 공분산, 상관계수
- 상관계수(r) : 분자가 공분산, 양의 상관관계일수록 값 ↑, 음의 상관관계일수록 값 ↓
    - -1 ~ 1 사이의 값, 양단으로 갈수록 강한 상관관계를 나타냄
        - 경험적인 기준의 절댓값
        - 강한 관계 : 0.5 ~ 1
        - 중간 관계 : 0.2 ~ 0.5
        - 약한 관계 : 0 ~ 0.1

```
# 상관 계수 (Correlation Coefficient)
correlation = df[['age', 'fare']].corr(method='pearson')
print("--- Correlation between Age and Fare ---")
print(correlation)

# 전체 변수에 대한 상관 계수 출력
print("\n--- Correlation between All Features ---")
print(df.corr())

# 수치형 변수에 대한 상관 계수 출력
numeric_values = df.select_dtypes(include=["number"]).corr()
print("\n--- Correlation between Numerical Features ---")
print(numeric_values)

# 히트맵을 그려서 시각화 후 확인 가능
sns.heatmap(numeric_values,           # 상관계수 출력한 DataFrame
```

```
            annot = True,           # 숫자(상관계수) 표기 여부
            fmt = '.3f',            # 숫자 포멧 : 소수점 3자리까지 표기
            cmap = 'RdYlBu_r',      # 칼라맵
            vmin = -1, vmax = 1)    # 값의 최소, 최대값
plt.show()
```

1. 같은 변수끼리 구한 값 1은 의미 없다.
2. 상관계수의 절대값이
    - 1에 가까울 수록 강한 상관관계
    - 0에 가까울 수록 약한 상관관계
3. +는 양의 상관관계, -는 음의 상관관계

- `scipy.stats`의 피어슨 상관분석 함수
    - NaN 값이 있으면 계산 안됨 (.notnull() 이후 계산)
    - 결과 : (상관계수, p-value) 튜플 형태로 반

```
# age의 결측치 확인
print(df['age'].isna().sum())

# dropna 를 활용한 age에서 결측치 존재하는 행 제거
df_drop_na = df.dropna(subset=['age'])
# 혹은 결측치가 아닌 행만 선택하는 식으로 추출해 올 수도 있다.
df_drop_na = df.loc[df['age'].notnull()]

# age의 결측치 재확인
print(df_drop_na['age'].isna().sum())

import scipy.stats as spst
# 상관계수와 p-value
spst.pearsonr(df_drop_na['age'], df_drop_na['fare'])
# (0.09606669176903893, 0.010216277504442105)
```

    - `p-value<0.05` 일 때 해당 상관계수가 의미 있다
      → 귀무가설이 참일 때 해당 상관계수가 나올 확률이 5% 미만이므로, 귀무가설을 기각하고 대립가설을 채택할 근거가 있다.
      → 상관분석에서의 귀무가설 : "두 변수는 상관 관계가 없다."
    - `age`와 `fare`의 상관계수는 `0.096` 으로 매우 낮은 관계임을 알 수 있다.
      `p-value`가 `0.01`이므로, 해당 상관계수는 통계적으로 유의미하다.
- 상세한 상관분석 관련 내용은 `2.3. 상관 및 회귀 분석/상관 분석` 확인

범주형변수 → 수치형변수

시각화 예시

```
# 평균비교 막대 그래프 (Bar Plot)
plt.figure(figsize=(8, 6))
sns.barplot(x='pclass', y='age', data=df)
```

```
# age의 전체 평균 추가 (전체 평균과 각 범주별 평균 비교)
plt.axhline(df['age'].mean(), color = 'r')
plt.title('Age Distribution by Pclass')
plt.show()

# 다변량 분석: 막대 그래프에 'sex' 변수 추가 (hue)
plt.figure(figsize=(8, 6))
sns.barplot(x='pclass', y='age', hue='sex', data=df)
plt.title('Age Distribution by Pclass')
plt.show()

# 박스 플롯 (Box Plot)
plt.figure(figsize=(8, 6))
sns.boxplot(x='pclass', y='age', data=df)
plt.title('Age Distribution by Pclass')
plt.show()

# 다변량 분석: 박스 플롯에 'sex' 변수 추가 (hue)
plt.figure(figsize=(10, 6))
sns.boxplot(x='pclass', y='age', hue='sex', data=df)
plt.title('Age Distribution by Pclass and Sex')
plt.show()
```

- `seaborn.barplot`은 자동으로 평균 계산 및 신뢰구간 계산 후 표시해줌
  - 신뢰구간(세로선)이 짧을수록 해당 평균 값에 대한 신뢰도가 높다.
  - 데이터가 많을수록, 편차가 적을수록 신뢰구간은 좁아진다.
  - **두 평균에 차이가 크고, 신뢰구간이 안 겹칠 때, 대립가설을 채택한다.**
- 범주 2개일 경우, 두 평균 차이를 비교하고, 3개 이상일 경우 전체 평균과 각 범주의 평균을 비교한다.
- `seaborn.boxplot`을 통해 중앙값과 이상치 확인이 가능하다.
- `hue` 파라미터를 추가하여 남녀 나이 분포를 다시 확인 가능하다.

수치화 예시

```
# 그룹별 기술 통계량
print("--- Age statistics by Survived ---")
print(df.groupby('survived')['age'].describe())

print("--- Age statistics by Pclass ---")
print(df.groupby('pclass')['age'].describe())
```

- `groupby`를 통해 각 그룹별 통계량 확인 및 수치 비교를 할 수 있다.
- 분석 도구를 통한 방법으로는 `T-TEST` 기법과 `ANOVA(분산 분석)`이 있다.

T-TEST

- 두 그룹의 평균 간 차이를 표준오차로 나눈 값 (두 평균의 차이로 이해해도 무관)
  - 두 그룹 간의 평균 비교를 위해 사용됨

- t 통계량이 -2보다 작거나, 2보다 크면, 두 그룹 간 차이가 있다고 본다.
- `ttest_ind(B, A, equal_var=False)`
    - `equal_var`: A, B 분산이 같을 경우 `True` (`default = True`)

```
# NaN 행 제외
temp = df.loc[df['age'].notnull()]
# 두 그룹으로 데이터 저장
died = temp.loc[temp['survived']==0, 'age']
survived = temp.loc[temp['survived']==1, 'age']

# scipy.stats로 t-test 진행
import scipy.stats as spst

spst.ttest_ind(died, survived)
# Ttest_indResult(statistic=2.06668694625381, pvalue=0.03912465401348249)

spst.ttest_ind(died, survived, equal_var=False)
# Ttest_indResult(statistic=2.046030104393971, pvalue=0.041189651625866304)
```

- 두 그룹(생존자와 사망자) 간 분산이 다를 때, T 통계량은 `2.04` 정도이고, 이 때 p-value는 `0.04`이다.
- p-value를 통해, 생존자와 사망자의 나이 평균은 통계적으로 유의한 차이가 있다.
- 다만 T 통계량을 통해 평균 차이가 크지는 않음을 알 수 있다.

ANOVA (ANalysis Of VAriance)

- 집단 간 평균의 차이가 존재하는지 확인
- 전체 평균을 기준으로 각 범주의 평균을 비교하는 방법
- 범주가 3개 이상일 때, 전체 평균과 각 그룹 평균 간 비교를 위해 사용
- F 통계량 =
    - (집단 간 분산)/(집단 내 분산) = (전체 평균 – 각 집단 평균)/(각 집단의 평균 – 개별 값)
    - 값이 대략 2~3 이상이면 차이가 있다고 판단 가능

```
# NaN 행 제외
temp = df.loc[df['age'].notnull()]
# 그룹별 저장
P_1 = temp.loc[temp['pclass'] == 1, 'age']
P_2 = temp.loc[temp['pclass'] == 2, 'age']
P_3 = temp.loc[temp['pclass'] == 3, 'age']

# scipy.stats로  f 통계량 계산 진행
import scipy.stats as spst
spst.f_oneway(P_1, P_2, P_3)
# F_onewayResult(statistic=57.443484340676214, pvalue=7.487984171959904e-24)
```

- F 통계량은 57.44, p-value는 7.48e-24이다.
- p-value를 통해, 객실별 나이 평균은 통계적으로 유의한 차이가 있다.
- F 통계량을 통해 나이 평균 차이가 존재함을 알 수 있다.
    - 얼마나 차이가 나는지는 사후분석을 통해 알아내야 한다.
    - 자세한건 2.2. 추론 통계/분산분석 확인

범주형변수 → 범주형변수

- 교차표로 집계 후, 시각화나 카이 제곱 검정을 실시할 수 있다.

```
# 교차표 만들기
print("--- Crosstab: Pclass vs Survived ---")
print(pd.crosstab(df['survived'], df['sex']))

# normalize 옵션을 넣으면 정규화한 값으로 출력 (columns, index, all)
print("--- Crosstab: Pclass vs Survived with Ratio ---")
print(pd.crosstab(df['survived'], df['sex'], normalize = 'index'))
```

- crosstab의 normalize 옵션 값
    - columns : 열의 값이 각각 합계가 1이 되도록 정규화
    - index : 행의 값이 각각 합계가 1이 되도록 정규화
    - all : 데이터프레임 전체의 값 합계가 1이 되도록 정규화

시각화 예시

```
# 그룹 막대 그래프 (Grouped Bar Chart)
crosstab_ps_ratio = pd.crosstab(df['survived'], df['sex'], normalize = 'index')
# stacked=True 작성 필수
crosstab_ps_ratio.plot(kind='bar', stacked=False)
# 전체 평균선 추가
plt.axhline(1-df['survived'].mean(), color = 'r')
# 추가적인 꾸미기 및 출력
plt.title('Survival Rate by Pclass')
plt.ylabel('Survival Rate')
plt.xticks(rotation=0)
plt.show()

# 모자이크 플롯은 statsmodels에서 불러와야 한다. mosaic(dataframe, [feature,
target])
from statsmodels.graphics.mosaicplot import mosaic
# Pclass별 생존여부를 mosaic plot으로 그리기
mosaic(df, [ 'pclass','survived'])
# 전체 평균선 추가
plt.axhline(1-df['survived'].mean(), color = 'r')
# 파일 내부 그래프 전체의 default 사이즈를 변경해주는 함수
plt.rcParams["figure.figsize"] = (12, 12)
# mosaic plot은 기존의 방식으로 그래프 크기 변경을 못하여 위와 같은 식으로 변경해야함
plt.show()
```

- 빨간 선은 전체 평균 (전체 사망률, 전체 생존율)
    - 평균선 추가시 1에서 뺀 값으로 추가한 이유:
      그래프에 그려질 때, 생존 표시가 상단에 표시가 되어서 평균 값(생존율)을 상단에 그리기 위함
- 모자이크 플롯 해석
    - x축 길이는 각 객실 등급별 승객 비율
    - y축 길이는 각 객실 승객 중에서 사망, 생존 비율
    - 만약 두 범주형 변수 간 상관이 없다면, 비율 차이가 거의 없다.

수치화 예시

- 카이제곱 검정 : 범주형 변수들 사시에 어떤 관계가 있는지, 수치화 하는 방법
    - 귀무가설 : 두 범주형 변수는 독립이다. (즉, 관계 없음)
    - 대립가설 : 두 범주형 변수는 독립이 아니다. (즉, 관계 있음) $$ \chi^2 = \sum\frac{(관측빈도 - 기대빈도)^2}{기대빈도} $$
- 기대빈도 : 아무런 관련이 없을 때 나올 수 있는 빈도 (확률적 독립일 때 기대되는 값)
- 카이 제곱 통계량은
    - 클수록 기대빈도로부터 실제 값에 차이가 크다는 의미.
    - 계산식으로 볼 때, 범주의 수가 늘어날 수록 값은 커지게 되어 있음.
    - 보통, 자유도의 2~3배 보다 크면, 차이가 있다고 본다.
- 범주형 변수의 자유도 = 범주의 수 - 1
    - 범주 중 하나는 무조건 선택되야하므로, 실질적으로 자유로운 선택 가능한 값은 범주의 수-1가지이다.
- 카이제곱검정에서는
    - x 변수의 자유도 × y 변수의 자유도
    - 예 : Pclass --> Survived
        - Pclass : 범주가 3개, Survived : 2개
        - (3-1) * (2-1) = 2
        - 그러므로, 2의 2~3배인 4~6 보다 카이제곱 통계량이 크면, 변수 간 차이가 있다고 볼수 있음.
- spst.chi2_contingency(교차표)

```
import scipy.stats as spst

# 1) 먼저 교차표 집계 (normalize 실행은 하면 안됨!!)
table = pd.crosstab(df['survived'], df['pclass'])

# 2) 카이제곱검정
print(spst.chi2_contingency(table))
'''
Chi2ContingencyResult(statistic=102.88898875696056,
                      pvalue=4.549251711298793e-23,
                      dof=2,
```

```
expected_freq=array([[133.09090909, 113.37373737, 302.53535354],
                     [ 82.90909091,  70.62626263, 188.46464646]]))
'''
```

- `statistic` : 카이제곱 통계량 → 자유도의 2배보다 크면 관계가 있다.
- `pvalue` : p-value → 0.05보다 작으면 관계가 있다 / 유의하다.
- `dof(자유도)` : Pclass 자유도(3-1) * Survived 자유도(2-1) = 2
- `기대빈도` : 계산된 값
- 해당 결과는 카이제곱 통계량이 자유도의 2배, 3배보다 크므로 두 변수 간 관계가 있으며, p-value가 0.05보다 낮으므로 통계적으로 유의하다.
- 기대 빈도는, 생존 여부와 객실 등급이 독립일 경우, 교차표의 각 셀에 대한 기대빈도이다.

survived	pclass=1	pclass=2	pclass=3
0 (사망)	80	97	372
1 (생존)	136	87	119

  - 위 표가 현재의 교차표이고, 기대 빈도와 맞춰서 해석하자면 "1등급 객실에서의 사망자는 133명일 것라고 해석"할 수 있다.

숫자형변수 → 범주형변수

히스토그램

- `seaborn` 0.11.0 이상일 경우, `sns.histplot()`을 통해 해결 가능

```
# Age에 대한 히스토그램을, Survived 범주로 나눠서 그리기
sns.histplot(x='age', data = df, hue = 'survived')
plt.show()
```

- `seaborn` 0.9.0에서는 아래와 같은 방식으로 가능

```
# distplot + 루프
import seaborn as sns
import matplotlib.pyplot as plt

# 결측치 제거
temp = df.loc[df['age'].notnull()]
# 반복문을 통해 생존 여부별 히스토그램 작성
for survived_value in [0, 1]:
    subset = temp[temp['survived'] == survived_value]
    sns.distplot(subset['age'], hist=True, kde=False, label=f"Survived=
{survived_value}")
```

```
plt.legend()
plt.show()

# FacetGrid 활용 (범주별 색 자동 분리)
g = sns.FacetGrid(df, hue="survived", height=5, aspect=1.5)
g.map(sns.distplot, "age", hist=True, kde=False)
g.add_legend()
plt.show()
```

밀도 함수

- 일부 방식은 `seaborn.kdeplot`에서 `hue` 파라미터를 지원해야한다.
- `seaborn` 0.11.0 이상부터 지원 / 0.9.0에서는 불가

1. `kdeplot( , hue = 'Survived')`
    - 생존여부의 비율이 유지된 채로 그려짐
      → 두 범주 간 비율을 비교 가능
    - 두 그래프의 아래 면적의 합이 1

```
# seaborn 0.11.0 이상에서 진행 방식
sns.kdeplot(x='age', data=df, hue ='survived')
plt.show()
```

2. `kdeplot( , hue = 'Survived', common_norm = False)`
    - 생존여부 각각 아래 면적의 합이 1인 그래프
    - 생존자 그래프 면적 1, 사망자 그래프 면적 1이므로,
      만약 나이에 따른 생존여부가 관련 없다면 두 그래프가 완전 겹친다.

```
# seaborn 0.11.0 이상에서 진행 방식
sns.kdeplot(x='age', data=df, hue ='survived', common_norm = False)
plt.show()
```

```
# seaborn 0.9.0 에서 진행 방식
for label, group in df.groupby('survived'):
    sns.kdeplot(group['age'], label=f'Survived={label}')

plt.legend()
plt.show()
```

3. `kdeplot( , hue = 'Survived', multiple = 'fill')`
    - 나이에 따라 생존여부 **비율** 비교 가능 (양의 비교가 아닌 비율)
    - 2번 그래프에서 두 그래프가 겹치는 구간 = 3번 그래프에서 평균선이 그래프와 겹치는 구간

    - 서로 관련이 없다면(독립이라면), 평균선을 기준으로 상하가 정확히 나눠진다.
    - 그래프 양 끝이 올라가는 이유는 전체를 확률로 표현하기 때문이다.

```
# seaborn 0.11.0 이상에서 진행 방식
sns.kdeplot(x='age', data=df, hue ='survived', multiple = 'fill')
plt.axhline(df['survived'].mean(), color = 'r')
plt.show()
```

    - 히스토그램 그려서 kde그래프와 비교해볼 수 있다.
    - 서로 관련이 없다면(독립이라면), 나이에 따른 생존여부 히스토그램이 평균선 근처를 지난다.

```
# seaborn 0.11.0 이상에서 진행 방식
sns.histplot(x='age', data = df, bins = 16, hue ='survived', multiple =
'fill')
plt.axhline(df['survived'].mean(), color = 'r')
plt.show()
```

- **결과 해석**
    - 약 15세 이하는 생존율이 전체 평균보다 높다.
    - 20~30대 생존율이 전체 평균보다 낮다.
    - 60~70대는 대부분 사망.

대표적인 다변량 시각화 도구

- `seaborn.pairplot`: 데이터프레임의 모든 수치형 변수 쌍에 대한 산점도와 각 변수의 분포를 한 번에 그려주어, 변수 간의 전반적인 관계를 탐색하는 데 매우 강력합니다.
- `seaborn.heatmap`: 상관계수 행렬을 시각화하여 여러 변수 간의 상관관계를 한눈에 파악하는 데 유용합니다.
- `seaborn.catplot`, `relplot`, `displot`: `hue`, `col`, `row`, `size`, `style` 등의 인자를 활용하여 3개 이상의 변수를 포함하는 다차원적인 시각화를 쉽게 구현할 수 있습니다.