AI Programming

데이터 마이닝 / 생성형 AI

03. 기초통계

first coding

first coding

기초 통계학 개념 소개

데이터의 중심 경향 (평균, 중앙값, 최빈값)

데이터 분산성 (분산, 표준편차, 사분위수)

데이터 분포 (정규분포, 왜도, 첨도)

데이터의 중심 경향 (평균, 중앙값, 최빈값)

- 데이터의 중심 경향

  - 데이터의 중심 경향은 데이터를 대표하는 값을 나타내는 척도
  - 평균, 중앙값, 최빈값은 각각 데이터를 요약하는 방법으로 사용
  - 각 방법은 서로 다른 상황에서 유용

데이터의 중심 경향 (평균, 중앙값, 최빈값)

- **평균**

  - 평균은 데이터를 모두 더한 뒤 데이터의 개수로 나눈 값으로 데이터를 고르게 나눈 "중심값"으로 볼 수 있음
  - 계산 방법

$$\text{평균} = \frac{\text{데이터의 총합}}{\text{데이터의 개수}}$$

  - 사례 : 학생들의 시험 점수가 80,85,90,70,75라고 할 때 평균

$$\text{평균} = \frac{80 + 85 + 90 + 70 + 75}{5} = \frac{400}{5} = 80$$

    - 평균 점수는 80으로, 학생들의 시험 성적을 대표하는 값이 됨

  - 특징 : 모든 데이터를 고려하므로 데이터가 균일할 때 유용

    - 극단값(이상치)가 포함되면 영향을 크게 받음

      → 위 데이터에 300점이 추가되면 평균이 7006=116.676700 로 왜곡

데이터의 중심 경향 (평균, 중앙값, 최빈값)

- 중앙값 (Median)

  - 중앙값은 데이터를 크기 순으로 정렬했을 때 가운데 위치한 값으로
  - 데이터가 극단값의 영향을 받을 때 평균보다 대표성을 가짐
  - 계산 방법

    ① 데이터를 크기 순으로 정렬.

    ② 데이터 개수가 홀수면 가운데 값이 중앙값.

    ③ 데이터 개수가 짝수면 가운데 두 값을 평균 낸 것이 중앙값.
  - 사례 : 학생들의 시험 점수가 80,85,90,70,75라고 할 때
    - 정렬: 70,75,80,85,90, 중앙값: 가운데 값인 80
  - 특징 : 극단값의 영향을 받지 않음.

    예를 들어, 데이터가 70,75,80,85,300일 때도 중앙값은 여전히 80, 데이터 분포가 비대칭일 때 대표값으로 유용

데이터의 중심 경향 (평균, 중앙값, 최빈값)

- **최빈값 (Mode)**

  - 최빈값은 데이터에서 가장 자주 나타나는 값으로 범주형 데이터나 특정 값의 빈도가 중요한 경우 유용
  - 계산 방법

    ① 데이터에서 각 값의 빈도를 계산.

    ② 가장 빈도가 높은 값을 선택.
  - 사례 : 학생들의 시험 점수가 80,85,90,70 이라고 할 때
    - 빈도 계산: 80(2),85(1),90(1),70(1) → 최빈값: 8080 (가장 자주 나타남)
  - 특징
    - 하나 이상의 최빈값이 있을 수 있음 (이중 최빈값). 예: 80,80,90,90,70 → 최빈값은 80과 90.
    - 연속형 데이터에서는 잘 사용되지 않음.예: 점수가 모두 다르면 최빈값이 없음.

데이터의 중심 경향 (평균, 중앙값, 최빈값)

- 데이터의 중심 경향

  - 비교와 활용

중심 경향	사용 목적	유용한 상황
평균	데이터를 고르게 대표	데이터가 균일할 때
중앙값	극단값의 영향을 배제	이상치가 포함된 경우
최빈값	가장 흔한 값을 찾음	범주형 데이터 분석 시

데이터의 중심 경향 (평균, 중앙값, 최빈값)

- 데이터의 중심 경향
  - statistics

```
import statistics

data = [10, 15, 10, 20, 30, 10, 40]

# 평균값 구하기
mean_value = statistics.mean(data)
print("평균:", mean_value)

# 중앙값 구하기
median_value = statistics.median(data)
print("중앙값:", median_value)

# 최빈값 구하기
mode_value = statistics.mode(data)
print("최빈값:", mode_value)

# 여러 최빈값 구하기
from statistics import multimode
modes = multimode([1, 2, 2, 3, 3])
print("여러 최빈값:", modes)  # 결과: [2, 3]
```

데이터의 중심 경향 (평균, 중앙값, 최빈값)

- 데이터의 중심 경향
  - pandas

```
# pandas를 이용해서 평균, 중앙값, 최빈값 구하기
import pandas as pd

data = [10, 15, 10, 20, 30, 10, 40]

data_series = pd.Series(data)

mean_value_pd = data_series.mean()
median_value_pd = data_series.median()
# 최빈값은 Series로 반환되므로 첫 번째 값을 선택
mode_value_pd = data_series.mode()[0]

print("pandas 평균:", mean_value_pd)
print("pandas 중앙값:", median_value_pd)
print("pandas 최빈값:", mode_value_pd)
```

데이터 분산성 (분산, 표준편차, 사분위수)

- **데이터 분산성**

  - 데이터 분산성은 데이터가 평균을 기준으로 얼마나 퍼져있는 지를 나타냄
  - 분산, 표준편차, 사분위수는 각각 다른 방식으로 데이터를 요약하여 데이터의 분포와 특성을 파악하는 데 유용

    - 각 데이터가 평균에서 얼마나 떨어져 있는지를 나타내는 값
      예: 평균이 70일 때, 80은 +10의 편차, 60은 -10의 편차
    - 편차들을 하나로 묶어 대표값처럼 나타낸 것
      편차들을 제곱해서 평균 내고 √(제곱근) 씌운 값
      → 즉, 전체 데이터가 평균에서 얼마나 퍼져 있는지를 나타내는 대표값

데이터 분산성 (분산, 표준편차, 사분위수)

- **데이터 분산성**

  - 분산 (Variance)
    - 분산은 데이터가 평균을 기준으로 얼마나 퍼져 있는지를 측정하는 값
      → 값이 클수록 데이터가 더 퍼져 있음을 의미
    - 계산 방법
      ① 각 데이터 값에서 평균을 뺀다 **(편차).**
      ② 편차를 제곱한다 (음수를 없애기 위해).
      ③ 제곱된 편차의 평균을 계산한다.

$$\text{분산} = \frac{\sum (x_i - \bar{x})^2}{n}$$

데이터 분산성 (분산, 표준편차, 사분위수)

- **데이터 분산성**

  - 분산 값이 크면 데이터가 평균으로부터 멀리 퍼져 있음.
  - 단위가 원래 데이터 단위의 제곱 단위여서 해석이 어려울 수 있음.

  – 분산 (Variance)

    - 사례 : 학생들의 점수가 80,85,90,70,75 라고 하면

      ① 평균 (x^-)

      $$\frac{80+85+90+70+75}{5} = 80$$

      ② 각 점수에서 평균을 뺀 편차

      80−80=0, 85−80=5, 90−80=10, 70−80=−10, 75−80=−5

      ③ 편차를 제곱

      $$0^2 = 0, 5^2 = 25, 10^2 = 100, (-10)^2 = 100, (-5)^2 = 25$$

      ④ 제곱된 값의 평균

      $$\text{분산} = \frac{0 + 25 + 100 + 100 + 25}{5} = \frac{250}{5} = 50$$

데이터 분산성 (분산, 표준편차, 사분위수)

- **데이터 분산성**

  - 표준편차 (Standard Deviation)
    - 표준편차는 분산의 제곱근
    - 데이터의 퍼짐 정도를 원래 데이터와 동일한 단위로 나타냄

$$\text{표준편차} = \sqrt{\text{분산}}$$

    - 사례 : 분산이 50이라고 하면

$$\text{표준편차} = \sqrt{50} \approx 7.07$$

      - 이 표준편차 값은 데이터가 평균 80을 기준으로 약 7.07 정도 퍼져 있음을 의미
    - 특징
      - 분산과 달리 데이터와 동일한 단위를 사용하므로 해석하기 쉬움
      - 값이 작으면 데이터가 평균에 가까이 모여 있고, 값이 크면 데이터가 퍼져 있음을 의미

데이터 분산성 (분산, 표준편차, 사분위수)

- **데이터 분산성**

  - 사분위수 (Quartiles)
    - 사분위수는 데이터를 네 개의 구간으로 나누는 값으로, 데이터의 분포를 요약하는 데 사용
      - Q1 (1사분위수): 하위 25% 데이터 경계값.
      - Q2 (2사분위수, 중앙값): 하위 50% 데이터 경계값.
      - Q3 (3사분위수): 하위 75% 데이터 경계값.
      - IQR (사분범위): $Q3-Q1$로, 중간 50% 데이터의 범위를 나타냄.
    - 특징
      - 분산과 달리 데이터와 동일한 단위를 사용하므로 해석하기 쉬움
      - 값이 작으면 데이터가 평균에 가까이 모여 있고, 값이 크면 데이터가 퍼져 있음을 의미

데이터 분산성 (분산, 표준편차, 사분위수)

- **데이터 분산성**

  - 사분위수 (Quartiles)
    - 계산 방법
      - 데이터를 오름차순으로 정렬.
      - Q1, Q2, Q3를 계산
        - » $Q1$= 하위 25% 위치 값.
        - » $Q2$= 중앙값.
        - » $Q3$= 상위 25% 위치 값.

    - 사례 : 데이터가 70,75,80,85,90일 때
      - ① 정렬된 데이터: 70,75,80,85,90
      - ② Q1: 75 (하위 25% 경계값)
      - ③ Q2: 80 (중앙값)
      - ④ Q3: 85 (상위 25% 경계값)
      - ⑤ 사분범위 (IQR): $Q3-Q1=85-75=10$
    - 특징
      - 이상치(outliers)를 파악하는 데 유용.
      - **이상치는 $Q1-1.5\times IQR$보다 작거나 $Q3+1.5\times IQR$보다 큰 값으로 정의.**

데이터 분산성 (분산, 표준편차, 사분위수)

- 데이터 분산성
  - 비교와 활용

척도	정의 및 특징	활용 상황
분산	데이터의 퍼짐 정도를 제곱 단위로 측정	데이터를 수학적으로 분석할 때
표준편차	데이터의 퍼짐 정도를 원래 단위로 측정	평균과 함께 데이터의 분포를 해석할 때
사분위수	데이터 분포의 경계값 및 중간 50% 범위를 측정	데이터 분포의 범위를 시각화 하거나 이상치를 파악할 때

데이터 분산성 (분산, 표준편차, 사분위수)

- 데이터 분산성
  - statistics

```
import statistics

data = [10, 15, 10, 20, 30, 10, 40]

# 분산 구하기
var_value = statistics.variance(data)
print("분산:", var_value)

# 표준편차 구하기
std_dev_value = statistics.stdev(data)
print("표준편차:", std_dev_value)

# 사분위수 구하기
q1_value = statistics.quantiles(data, n=4)[0]  # 1사분위수
q2_value = statistics.quantiles(data, n=4)[1]  # 2사분위수
q3_value = statistics.quantiles(data, n=4)[2]  # 3사분위수
q4_value = max(data)  # 4사분위수는 최대값

print("1사분위수:", q1_value)
print("2사분위수 (중앙값):", q2_value)
print("3사분위수:", q3_value)
print("4사분위수:", q4_value)

# IQR (Interquartile Range) 사분위 범위 구하기
iqr_value = q3_value - q1_value
print("IQR(사분위 범위):", iqr_value)
```

데이터 분산성 (분산, 표준편차, 사분위수)

- 데이터 분산성
  - statistics

계산 항목	pandas 메서드 예시
분산	df['컬럼명'].var()
표준편차	df['컬럼명'].std()
사분위수 Q1	df['컬럼명'].quantile(0.25)
사분위수 Q2	df['컬럼명'].quantile(0.50)
사분위수 Q3	df['컬럼명'].quantile(0.75)
IQR	Q3 - Q1

```
# pandas를 이용한 분산, 표준편차, 사분위수
import pandas as pd

data = {'점수': [55, 60, 65, 70, 75, 80, 85, 90, 95]}

df = pd.DataFrame(data)

# 분산 구하기
var_value_pd = df['점수'].var()
print("pandas 분산:", var_value_pd)

# 표준편차 구하기
std_dev_value_pd = df['점수'].std()
print("pandas 표준편차:", std_dev_value_pd)

# 사분위수 구하기
q1_value_pd = df['점수'].quantile(0.25)  # 1사분위수
q2_value_pd = df['점수'].quantile(0.5)   # 2사분위수 (중앙값)
q3_value_pd = df['점수'].quantile(0.75)  # 3사분위수
q4_value_pd = df['점수'].max()  # 4사분위수는 최대값

print("pandas 1사분위수:", q1_value_pd)
print("pandas 2사분위수 (중앙값):", q2_value_pd)
print("pandas 3사분위수:", q3_value_pd)
print("pandas 4사분위수:", q4_value_pd)

# IQR (Interquartile Range) 사분위 범위 구하기
iqr_value_pd = q3_value_pd - q1_value_pd
print("pandas IQR(사분위 범위):", iqr_value_pd)
```

데이터 분포 (정규분포, 왜도, 첨도)

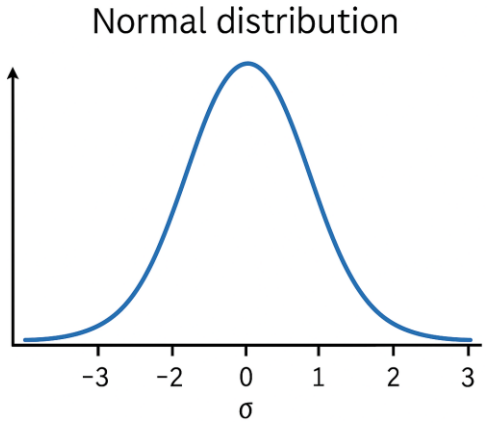
- 데이터 분포

  - 데이터 분포는 데이터가 공간적으로 어떻게 퍼져 있는지를 나타냄
  - 정규분포, 왜도, 첨도는 데이터의 분포 특성을 이해하는 데 중요한 개념

데이터 분포 (정규분포, 왜도, 첨도)

- **데이터 분포**

  - 정규분포 (Normal Distribution)
    - 정규분포는 데이터가 평균을 중심으로 대칭적으로 분포된 종 모양의 분포
    - 데이터의 대부분이 평균 근처에 몰려 있고, 평균에서 멀어질수록 빈도가 줄어듬
    - 특징
      - 평균, 중앙값, 최빈값이 동일.
      - 데이터의 68%는 평균 ±1 표준편차 내에 있음.
      - 데이터의 95%는 평균 ±2 표준편차 내에 있음.



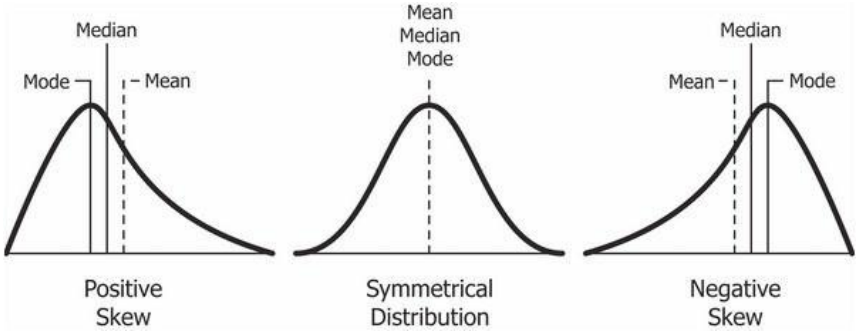
데이터 분포 (정규분포, 왜도, 첨도)

- **데이터 분포**

  - 정규분포 (Normal Distribution)
    - 사례 : 학생들의 수학 시험 점수가 정규분포를 따른다고 가정
      - 평균 점수: 70점
      - 표준편차: 10점
      - 이 경우
        - » 68% 학생은 70±10, 즉 60점에서 80점 사이에 있음.
        - » 95% 학생은 70±20, 즉 50점에서 90점 사이에 있음.

데이터 분포 (정규분포, 왜도, 첨도)

- **데이터 분포**
  - 왜도 (Skewness)
    - 왜도는 데이터의 비대칭성을 측정하는 값
    - 분포가 한쪽으로 치우쳐 있으면 왜도가 발생
      - 왜도 > 0 (양의 왜도): 데이터가 오른쪽으로 꼬리가 길게 늘어남.
      - 왜도 < 0 (음의 왜도): 데이터가 왼쪽으로 꼬리가 길게 늘어남.
      - 왜도 = 0: 데이터가 대칭적임 (정규분포).
    - 특징
      - 왜도를 통해 분포의 비대칭성을 파악.
      - 양의 왜도: 평균 > 중앙값 > 최빈값.
      - 음의 왜도: 평균 < 중앙값 < 최빈값.



데이터 분포 (정규분포, 왜도, 첨도)

- **데이터 분포**

  - 왜도 (Skewness) 사례
    - 양의 왜도 (Positive Skewness)
      - 주택 가격 데이터

        대부분의 집값은 중간값 근처에 있지만, 일부 고급 주택이 매우 높은 가격대를 형성해 꼬리가 오른쪽으로 길어짐.

        예: $300K$,$320K$,$350K$,$1M$,$2M$ → 평균 >> 중앙값.
    - 음의 왜도 (Negative Skewness)
      - 시험 점수 데이터

        대부분의 학생이 높은 점수를 받았지만, 일부 학생의 점수가 매우 낮아 꼬리가 왼쪽으로 길어짐.

        예: 90,92,94,95,20 → 평균 << 중앙값.

- 왜도를 통해 분포의 비대칭성을 파악.
- 양의 왜도: 평균 > 중앙값 > 최빈값.
- 음의 왜도: 평균 < 중앙값 < 최빈값.

데이터 분포 (정규분포, 왜도, 첨도)

- **데이터 분포**

  - 첨도 (Kurtosis)
    - 첨도는 분포의 꼭대기가 얼마나 뾰족하거나 평평한지를 측정하는 값
      - 첨도 > 3 (Leptokurtic): 꼭대기가 뾰족하고 꼬리가 두꺼움.
        - » 예: 주식 수익률 (극단값이 자주 발생).
      - 첨도 = 3 (Mesokurtic): 정규분포와 비슷한 모양.
      - 첨도 < 3 (Platykurtic): 꼭대기가 평평하고 꼬리가 얇음.
        - » 예: 시험 점수 분포 (극단값이 적음).

데이터 분포 (정규분포, 왜도, 첨도)

- **데이터 분포**

  - 첨도 (Kurtosis) 사례
    - 첨도 > 3 (Leptokurtic)

      주식 투자 수익률 데이터 : 대부분의 수익률이 평균에 가까우나, 일부 극단적인 큰 이익/손실이 발생.
      - 데이터 예: 50%, -10%, 0%, 10%, 200%
    - 첨도 < 3 (Platykurtic)

      학생들의 키 데이터 : 대부분의 키가 고르게 분포되어 있으며 극단적인 값이 거의 없음.
      - 데이터 예: $150cm$, $155cm$, $160cm$, $165cm$, $170cm$
    - 첨도 = 3 (Mesokurtic)

      완벽한 정규분포 형태.

- 첨도가 크면 극단값(꼬리)이 많아 분석에 주의가 필요.
- 첨도가 작으면 데이터가 평균 근처에 고르게 분포.

데이터 분포 (정규분포, 왜도, 첨도)

- 데이터 분포

  - 비교와 활용

척도	정의 및 특징	활용 상황
정규분포	평균을 중심으로 대칭적 분포	자연현상 분석, 통계 분석
왜도	분포의 비대칭성 측정	데이터 치우침 확인 (예: 주택 가격 분석)
첨도	분포의 꼭대기 뾰족함과 꼬리 두께 측정	극단값의 빈도 및 위험 분석 (예: 주식 수익률)

데이터 분포 (정규분포, 왜도, 첨도)

- 데이터 분포

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
from scipy.stats import skew, kurtosis

# 정규분포 데이터 생성 (평균=0, 표준편차=1, 1000개 샘플)
data = np.random.normal(loc=0, scale=1, size=1000)

# 데이터프레임으로 변환 (선택사항)
df = pd.DataFrame(data, columns=['값'])

# 한글 폰트 설정
plt.rcParams['font.family'] = 'Malgun Gothic'
# 정규분포 히스토그램 시각화
plt.hist(df['값'], bins=30, edgecolor='black', alpha=0.7)
plt.title('정규분포 히스토그램')
plt.xlabel('값')
plt.ylabel('빈도수')
plt.grid(True)
plt.show()

# 왜도(Skewness) 계산
skewness = skew(df['값'])
print(f"왜도 (Skewness): {skewness:.4f}")

# 첨도(Kurtosis) 계산
kurt = kurtosis(df['값'])  # Fisher 방식 (정규분포면 0이 나옴)
print(f"첨도 (Kurtosis): {kurt:.4f}")
```

first coding

데이터 통합 및 시각화

데이터 통합 및 조인 (merge, concat)

데이터 분포 확인 (히스토그램, 박스플롯)

데이터 탐색적 분석(EDA)을 위한 시각화 기법

데이터 통합 및 조인 (merge, concat)

- **데이터 통합 및 조인**

  - 데이터 분석을 할 때, 여러 데이터셋을 하나로 합쳐야 할 때가 많음
  - 이 과정에서 merge와 concat은 매우 중요한 역할을
  - 두 방법 모두 **데이터를 합치는 기능**을 하지만 어떤 기준으로 데이터를 합치는지에 따라 사용 방식이 달라 짐

데이터 통합 및 조인 (merge, concat)

- 데이터 통합 및 조인

  - merge
    - 기준 열을 기반으로 데이터를 합치기
      - SQL의 JOIN과 유사
      - 두 데이터프레임의 공통 열(key)을 기준으로 데이터를 합칠 수 있음

    - merge 방식
      - Inner Join (교집합): 두 데이터프레임의 공통된 값만 유지.
      - Outer Join (합집합): 공통된 값이 없어도 데이터 유지 (NaN으로 채움).
      - Left Join: 왼쪽 데이터프레임의 값을 모두 유지.
      - Right Join: 오른쪽 데이터프레임의 값을 모두 유지.

데이터 통합 및 조인 (merge, concat)

- 데이터 통합 및 조인 예제

```
Inner Join:
    customer_id   name  order_id product
0             1  Alice       101  Laptop
1             2    Bob       102  Tablet

Left Join:
    customer_id     name  order_id product
0             1    Alice     101.0  Laptop
1             2      Bob     102.0  Tablet
2             3  Charlie       NaN     NaN
```

```python
import pandas as pd

# 고객 정보 데이터프레임
customers = pd.DataFrame({
    'customer_id': [1, 2, 3],
    'name': ['Alice', 'Bob', 'Charlie']
})

# 주문 정보 데이터프레임
orders = pd.DataFrame({
    'order_id': [101, 102, 103],
    'customer_id': [1, 2, 4],
    'product': ['Laptop', 'Tablet', 'Smartphone']
})

# Inner Join (공통된 customer_id로만 합침)
result_inner = pd.merge(customers, orders, on='customer_id', how='inner')

# Left Join (customers 데이터프레임의 모든 값 유지)
result_left = pd.merge(customers, orders, on='customer_id', how='left')

# 결과 출력
print("Inner Join:\n", result_inner)
print("\nLeft Join:\n", result_left)
```

데이터 통합 및 조인 (merge, concat)

- 데이터 통합 및 조인

  - concat
    - 데이터 방향으로 쌓기
      - 데이터를 방향(축)으로 단순히 이어 붙임
      - 기준 열 없이 데이터를 추가하고 싶을 때 유용

    - concat 방식
      - Row-wise Concatenation (위아래로 쌓기): axis=0
      - Column-wise Concatenation (왼쪽에서 오른쪽으로 합치기): axis=1

데이터 통합 및 조인 (merge, concat)

- 데이터 통합 및 조인 예제

```
      month     product  sales
0   January      Laptop    100
1   January      Tablet    150
2   January  Smartphone    200
0  February      Laptop    120
1  February      Tablet    160
2  February  Smartphone    220
```

```python
import pandas as pd

# 1월 매출 데이터
sales_jan = pd.DataFrame({
    'month': ['January', 'January', 'January'],
    'product': ['Laptop', 'Tablet', 'Smartphone'],
    'sales': [100, 150, 200]
})

# 2월 매출 데이터
sales_feb = pd.DataFrame({
    'month': ['February', 'February', 'February'],
    'product': ['Laptop', 'Tablet', 'Smartphone'],
    'sales': [120, 160, 220]
})

# Row-wise Concatenation (1월과 2월 데이터를 이어붙임)
sales_combined = pd.concat([sales_jan, sales_feb], axis=0)

# 결과 출력
print(sales_combined)
```

데이터 통합 및 조인 (merge, concat)

- 데이터 통합 및 조인

  - merge와 concat의 차이점

기능	merge	concat
기준	공통된 열(key)을 기준으로 데이터프레임 합침	기준 없이 단순히 데이터를 이어 붙임
방향	보통 열 방향으로 합침	행 방향(axis=0) 또는 열 방향(axis=1)
사용 목적	데이터를 조합하거나 관계를 분석할 때 사용	데이터를 확장하거나 쌓을 때 사용

데이터 분포 확인 (히스토그램, 박스플롯)

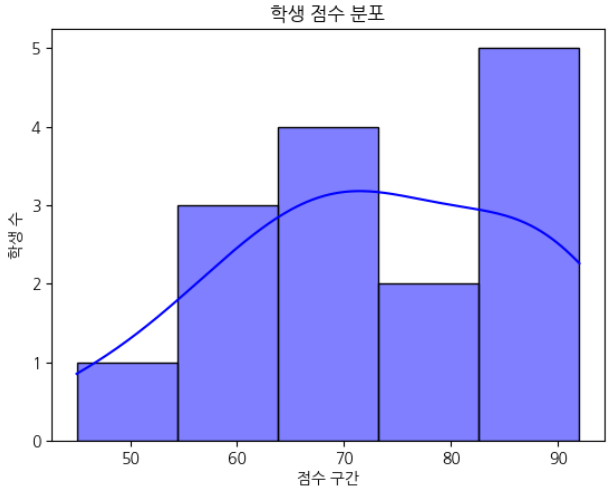
- 데이터 분포 확인

  - 히스토그램 (Histogram)
    - 히스토그램은 데이터의 분포를 나타내는 데 사용
    - 연속형 데이터를 구간(bin)으로 나누고 각 구간에 해당하는 데이터의 빈도를 보여줌
    - 사용 목적
      - 데이터 값이 어떻게 퍼져 있는지 확인 (분포 확인)
      - 데이터의 대략적인 형태 (정규분포, 치우침 등) 파악
      - 이상치(outlier) 탐지

데이터 분포 확인 (히스토그램, 박스플롯)

- 데이터 분포 확인
  - 히스토그램 (Histogram) 예제

    → 학생들의 시험 점수 분포



  - 히스토그램은 점수가 50-60점 사이에 몇 명이 있는지, 80-90점 사이에 몇 명이 있는지 등 데이터의 밀도를 직관적으로 보여줌

```python
import seaborn as sns
import matplotlib.pyplot as plt

# 학생 점수 데이터
scores = [56, 67, 45, 89, 90, 77, 68, 88, 92, 76, 58, 70, 73,
85, 62]

# Seaborn 히스토그램 생성
plt.rc('font', family='NanumGothic') # For Windows

sns.histplot(scores, bins=5, kde=True, color='blue',
edgecolor='black’)
# kde=True는 밀도곡선 추가
# bins=5 5개 구간으로 구분

plt.title("학생 점수 분포")
plt.xlabel("점수 구간")
plt.ylabel("학생 수")
plt.show()
```

데이터 분포 확인 (히스토그램, 박스플롯)

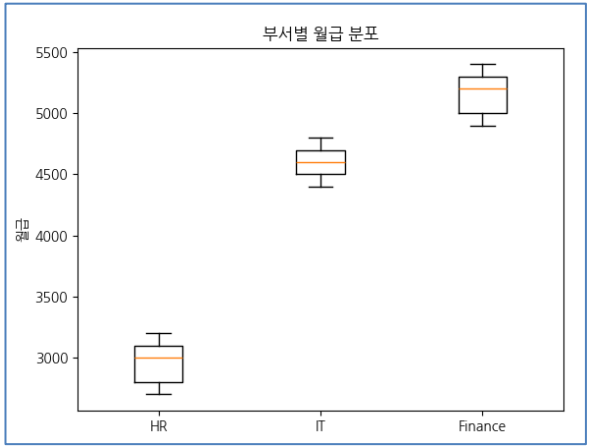
- 데이터 분포 확인

  - 박스플롯 (Boxplot)
    - 데이터의 중앙값, 사분위수, 이상치를 시각화, 데이터 분포를 요약하고, 이상치를 쉽게 탐지할 수 있는 도구
    - 사용 목적
      - 데이터의 중앙값과 변동성 확인, 이상치(outlier) 탐지, 데이터 간 비교
    - 박스플롯의 주요 구성 요소
      - 중앙값(Median): 데이터의 중간값 (박스의 중앙선)
      - 1사분위수(Q1): 하위 25% 데이터 값
      - 3사분위수(Q3): 상위 25% 데이터 값
      - IQR(Interquartile Range): Q3 - Q1 (데이터 변동성 측정)
      - 이상치(Outliers): Q1 - 1.5×IQR 이하, 또는 Q3 + 1.5×IQR 이상 값

데이터 분포 확인 (히스토그램, 박스플롯)

- 데이터 분포 확인
  - 박스플롯 (Boxplot) 예제

    → 학생들의 시험 점수 분포




```python
import matplotlib.pyplot as plt
# 데이터 변환: Seaborn은 Long Format 데이터가 필요

# 부서별 월급 데이터
departments = ['HR', 'IT', 'Finance']
salaries = [
    [3000, 3200, 3100, 2800, 2700],  # HR
    [4500, 4600, 4700, 4400, 4800],  # IT
    [5200, 5300, 5000, 4900, 5400]   # Finance
]

# 박스플롯 생성
plt.rc('font', family='NanumGothic') # For Windows
plt.boxplot(salaries, labels=departments)
plt.title("부서별 월급 분포")
plt.ylabel("월급")
plt.show()
```

  - 각 부서의 월급 중앙값과 분포를 확인할 수 있음.
  - HR 부서: 월급 분포가 좁고 중앙값이 낮음.
  - Finance 부서: 월급이 전반적으로 높음. 이상치가 있다면 박스플롯에 점으로 표시

데이터 분포 확인 (히스토그램, 박스플롯)

- 데이터 분포 확인

  - 히스토그램과 박스플롯의 차이점

특징	히스토그램	박스플롯
목적	데이터의 **빈도 분포**를 시각화	데이터의 **요약 통계값**과 이상치 탐지
데이터 유형	연속형 데이터	연속형 데이터
주요 정보	구간(bin)의 데이터 개수	중앙값, 사분위수, 이상치
비교 가능성	단일 데이터셋 시각화에 적합	여러 데이터셋 간 비교에 적합

데이터 탐색적 분석(EDA)을 위한 시각화 기법

- **데이터 탐색적 분석(EDA)을 위한 시각화 기법**

  - EDA(탐색적 데이터 분석)는 데이터를 시각적으로 분석하여
  - 패턴, 관계, 이상치, 분포 등을 탐구하는 과정
  - 다양한 시각화 기법을 통해 데이터를 더 깊이 이해하고 분석 방향을 설정할 수 있음

데이터 탐색적 분석(EDA)을 위한 시각화 기법

- 데이터 탐색적 분석(EDA)을 위한 시각화 기법

  - 주요 시각화 기법과 사례
    - 산점도 (Scatter Plot)
      - 데이터 간의 관계 확인목적

        → 두 변수 간의 관계를 확인

        예를 들어,

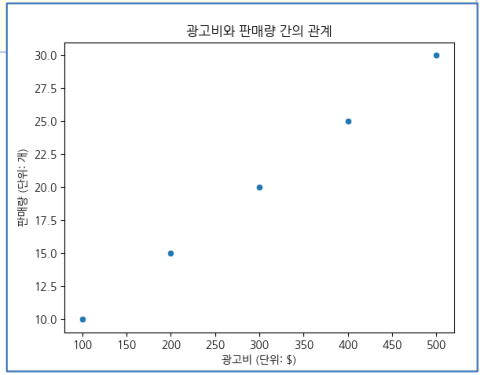
        변수 A와 변수 B가 양의 상관관계인지,

        음의 상관관계인지,

        아니면 상관이 없는지 확인

    - 사용 사례 : 광고비와 판매량 간의 관계를 분석.
      - 광고비가 증가할수록 판매량도 증가하는 양의 상관관계를 관찰할 수 있음

```python
import seaborn as sns
import matplotlib.pyplot as plt

# 데이터
import pandas as pd
data = pd.DataFrame({
    'Ad_Spend': [100, 200, 300, 400, 500],
    'Sales': [10, 15, 20, 25, 30]
})

plt.rc('font', family='NanumGothic') # For Windows
# 산점도 생성
sns.scatterplot(x='Ad_Spend', y='Sales', data=data)
plt.title("광고비와 판매량 간의 관계")
plt.xlabel("광고비 (단위: $)")
plt.ylabel("판매량 (단위: 개)")
plt.show()
```



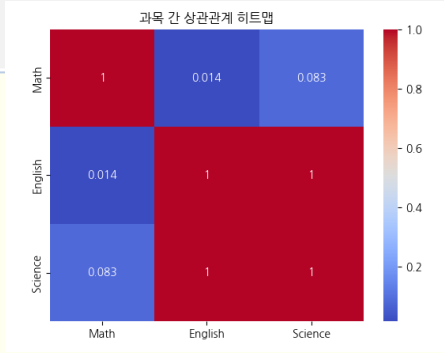
데이터 탐색적 분석(EDA)을 위한 시각화 기법

- 데이터 탐색적 분석(EDA)을 위한 시각화 기법

  - 주요 시각화 기법과 사례
    - **히트맵 (Heatmap)** : 변수 간 상관관계 분석
      - 목적

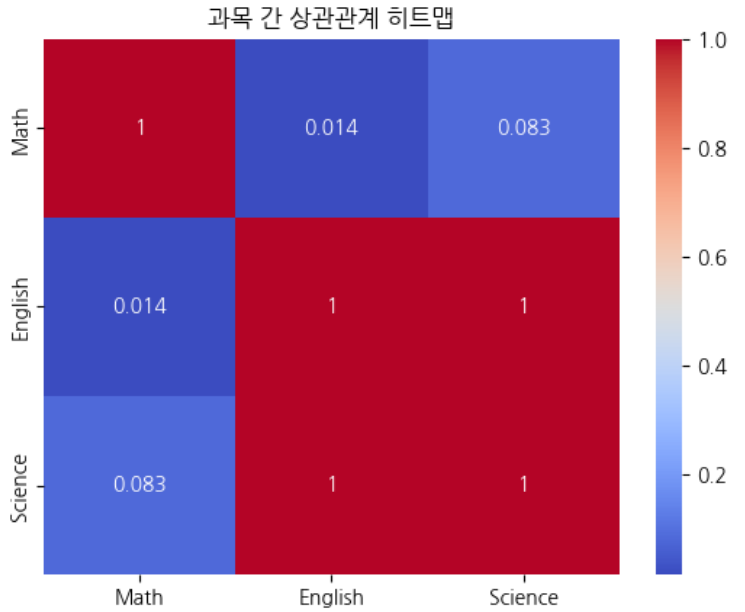
        여러 변수 간의 상관관계를 확인
      - 사용 사례
        - » 학생들의 시험 점수 데이터에서 과목 간 상관관계를 확인.
    - 학생 점수 상관관계
      - 현재 데이터에서는 과목 간 상관관계가 제한적이며,
      - 영어와 과학의 상관관계(1)는 데이터 패턴의 제한된 특성에서 비롯된 것
      - 보다 신뢰성 있는 결론을 위해 데이터의 크기와 다양성을 확장하고 이상치를 처리하는 과정이 필요

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

# 데이터 생성
data = pd.DataFrame({
    'Math': [90, 80, 70, 60, 85],
    'English': [85, 75, 65, 55, 10],
    'Science': [88, 78, 68, 58, 23]
})

# 상관계수 계산
correlation = data.corr()

plt.rc('font', family='NanumGothic') # For Windows
# 히트맵 생성
sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title("과목 간 상관관계 히트맵")
plt.show()
```



데이터 탐색적 분석(EDA)을 위한 시각화 기법

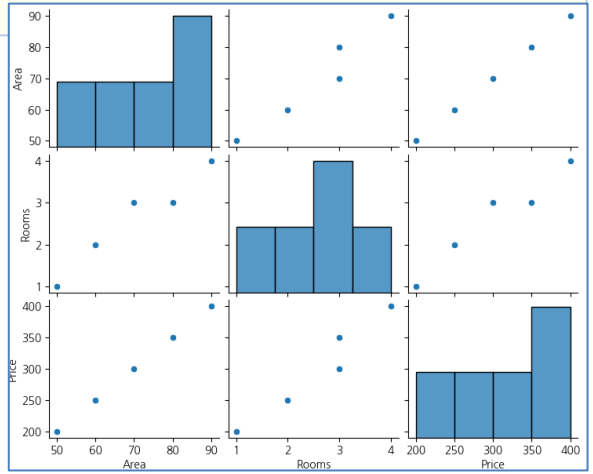
- 데이터 탐색적 분석(EDA)을 위한 시각화 기법
- 데이터 해석
  - Math(수학)는 다른 과목과 거의 독립적
  - Math-English 상관계수 0.014는
    수학 점수와 영어 점수 간의 관계가 거의 없음을 의미
  - Math-Science 상관계수 0.083은
    약간의 양의 상관관계를 보여주지만, 실제 데이터 상에서는
    거의 무시할 만한 수준
  - English(영어)와 Science(과학)는 상관계수가 1로
    동일한 값을 갖음
    이는 두 과목 점수가 동일한 패턴을 보이거나 데이터 자체에서 값들이 강하게 일치한 결과로 나타났을 가능성이
    큼



데이터 탐색적 분석(EDA)을 위한 시각화 기법

- 데이터 탐색적 분석(EDA)을 위한 시각화 기법
  - 주요 시각화 기법과 사례
    - **페어플롯 (Pair Plot)**
      - 변수 간 관계를 한눈에 확인
      - 데이터셋의 모든 수치형 변수 간의 관계를 시각화
    - 사용 사례 : 주택 데이터 분석
      - 방 개수가 많을수록 면적이 커지고, 가격도 높아지는 경향을 관찰 가능.

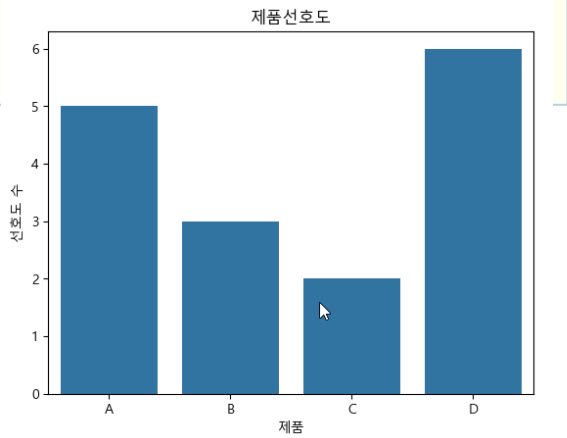
```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# 데이터
data = pd.DataFrame({
    'Area': [50, 60, 70, 80, 90],
    'Rooms': [1, 2, 3, 3, 4],
    'Price': [200, 250, 300, 350, 400]
})
# 페어플롯 생성
sns.pairplot(data)
plt.suptitle("주택 데이터 변수 간 관계", y=1.02)
plt.show()
```



데이터 탐색적 분석(EDA)을 위한 시각화 기법

- 데이터 탐색적 분석(EDA)을 위한 시각화 기법
  - 주요 시각화 기법과 사례
    - **카운트플롯 (Count Plot)**
      - 범주형 데이터 시각화
      - 범주형 데이터의 빈도를 확인

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

# data set
data = pd.DataFrame({
    'Product' : ['A', 'B', 'A', 'C', 'B', 'A', 'B', 'C', 'A', 'A',
'D', 'D', 'D', 'D', 'D', 'D']
})

plt.rc('font', family='Malgun Gothic')
sns.countplot(x='Product', data=data)
plt.title('제품선호도')
plt.xlabel('제품')
plt.ylabel('선호도 수')
plt.show()
```



- 사용 사례 : 제품 선호도 설문 결과를 시각화.
  - 제품 A가 가장 선호도가 높고,
    제품 C가 상대적으로 낮은 선호도를 가짐.

- 데이터 탐색적 분석(EDA)을 위한 시각화 기법

  – 시각화 기법의 선택 기준

시각화 기법	사용 목적
산점도 (Scatter Plot)	두 변수 간의 관계를 시각적으로 탐구
히트맵 (Heatmap)	여러 변수 간의 상관관계를 시각화
박스플롯 (Boxplot)	데이터의 분포와 이상치 탐지
페어플롯 (Pair Plot)	데이터셋의 모든 변수 간 관계 탐색
카운트플롯 (Count Plot)	범주형 데이터의 빈도 탐색

first coding

데이터 분석 1 - 기초 분석

데이터 탐색적 분석(EDA)

기초 통계량 계산

데이터의 기본 구조 파악

데이터 탐색적 분석(EDA)

- 데이터 탐색적 분석 (Exploratory Data Analysis, EDA)

  - EDA는 데이터 분석 과정에서 데이터를 이해하고 통찰(insights)을 얻기 위한 초기 분석 단계
    - **데이터셋의 구조와 특성을 탐색**
    - **문제가 되는 부분을 확인**
    - **분석 방향을 설정하는 데 필수적인 과정**

  - 데이터를 "알아가는" 과정

데이터 탐색적 분석(EDA)

- EDA의 주요 목적

  - **데이터의 기본 특성 이해**
    - 데이터를 시각적으로 관찰하고, 수치로 요약하고 데이터 분포, 이상치, 결측치 등을 확인
  - **데이터 정리**
    - 결측 값이나 이상치를 처리하고, 데이터를 분석 가능한 상태로 준비
  - **가설 수립 및 데이터 기반 의사 결정**
    - 데이터를 관찰하며 분석 방향과 가설을 수립

      예를 들어, 매출 데이터에서 특정 요일에 판매량이 높다는 패턴을 찾을 수 있음
  - **문제 발견**
    - 분석 중 문제가 될 수 있는 요소를 확인

      예: 값이 비정상적으로 높거나 낮은 이상치(outliers), 비어 있는 데이터(null values).

데이터 탐색적 분석(EDA)

- EDA의 주요 단계

① 데이터의 기본 특성 이해

- 데이터 불러오기 및 기본 정보 확인데이터셋의 크기(행과 열의 수), 변수명, 데이터 유형(숫자, 문자 등)을 확인
- Python에서 pandas 라이브러리의 head(), info(), describe() 등을 사용

```
import pandas as pd

# 데이터 불러오기
df = pd.read_csv('data.csv')

# 기본 정보 확인
print(df.head())          # 데이터 상위 5행 확인
print(df.info())          # 데이터 타입과 결측치 확인
print(df.describe())      # 수치형 데이터의 요약 통계량
```

데이터 탐색적 분석(EDA)

- EDA의 주요 단계

② 결측 값과 이상치 확인

- **결측 값**
  - 데이터가 비어 있는 경우 (NaN 값).
- **이상치**
  - 비정상적으로 높거나 낮은 값.
- **이를 확인하고 처리 방법을 결정**
  - 삭제
  - 대체 (평균값, 중앙값, 또는 특정 값으로)

```
# 결측값 확인
print(df.isnull().sum())

# 이상치 시각화 예제
import matplotlib.pyplot as plt
import seaborn as sns

sns.boxplot(x=df['Age'])
plt.show()
```

데이터 탐색적 분석(EDA)

- EDA의 주요 단계

② 결측 값과 이상치 확인

- **결측 값**
  - 데이터가 비어 있는 경우 (NaN 값).
- **이상치**
  - 비정상적으로 높거나 낮은 값.
- **이를 확인하고 처리 방법을 결정**
  - 삭제
  - 대체 (평균값, 중앙값, 또는 특정 값으로)

```
# 결측값 확인
print(df.isnull().sum())

# 이상치 시각화 예제
import matplotlib.pyplot as plt
import seaborn as sns

sns.boxplot(x=df['Age'])
plt.show()
```

데이터 탐색적 분석(EDA)

- EDA의 주요 단계

③ 데이터 분포 확인

- 변수의 분포를 이해하기 위해 히스토그램이나 커널 밀도 그래프를 사용
- 데이터가 정규 분포(normal distribution)를 따르는 지 확인할 수 있음

```
# 데이터 분포 확인
# 히스토그램
df['Age'].hist(bins=30)
plt.show()
```

데이터 탐색적 분석(EDA)

- EDA의 주요 단계

④ 변수 간 관계 탐색

- 두 변수 사이의 상관관계를 확인하거나, 그룹별 통계량을 계산

  예: 매출 데이터에서 제품별 매출 합계를 비교

데이터 탐색적 분석(EDA)

- EDA의 주요 단계

⑤ 시각화

- 시각화는 데이터를 더 직관적으로 이해할 수 있게 도와줌
- 사용하는 도구: Matplotlib, Seaborn

데이터 탐색적 분석(EDA)

- EDA의 주요 단계

  - EDA의 결과물
    - EDA를 통해 데이터를 충분히 이해하면 다음을 얻을 수 있음
      - 데이터셋의 구조와 특성에 대한 이해
      - 이상치와 결측치 처리 계획
      - 데이터 분포와 변수 간 관계에 대한 통찰
      - 분석 방향성 설정 및 가설 정립

데이터 탐색적 분석(EDA)

- EDA 활용 예시

  - 데이터셋: 판매 데이터

날짜	제품명	판매량	가격
2024-01-01	A	100	5000
2024-01-02	B	NaN	7000
2024-01-03	A	150	5000
2024-01-04	C	200	8000

① **결측 값 처리**

  - 판매량에 NaN이 있으므로 평균값으로 대체.

② **판매량 분포 확인**

  - 각 제품의 판매량 히스토그램으로 확인.

③ **변수 간 관계**

  - 가격과 판매량의 상관관계 확인.

데이터의 기본 구조 파악

- 데이터의 기본 구조 파악

  - 데이터 분석을 시작하기 전에 데이터의 기본 구조를 이해하는 것은 매우 중요
  - 데이터를 구조적으로 이해하면 분석 목표에 맞는 전략을 세우고, 데이터를 올바르게 처리할 수 있음
  - 데이터의 기본 구조를 파악하는 과정
    - **데이터의 모양(형태)**
    - **데이터 타입**
    - **변수의 분포 및 관계**
    - **결측 값 유무**

데이터의 기본 구조 파악

- **데이터의 기본 구조**

① 데이터셋의 크기

- 데이터셋이 몇 개의 행(row)과 열(column)로 구성되어 있는지 확인
- 데이터를 요약할 때 데이터의 전반적인 크기를 알고 있어야 효율적으로 작업할 수 있음

② 데이터 타입

- 각 열의 데이터 유형을 이해 → 데이터 유형은 분석 방법에 영향을 미침
  - 수치형 데이터(Numeric): 정수형, 부동소수점.
  - 범주형 데이터(Categorical): 특정 그룹이나 클래스를 나타냄.
  - 문자형 데이터(String): 텍스트 데이터.
  - 날짜/시간 데이터(DateTime): 시간 기반 데이터.

데이터의 기본 구조 파악

- **데이터의 기본 구조**

③ 변수의 이름과 의미

- 데이터셋에 포함된 변수(컬럼)의 이름과 그 의미를 파악

④ 결측값

- 데이터셋에 비어 있는 값이 있는지 확인

  → 이는 분석 결과에 영향을 미칠 수 있으므로 반드시 처리해야 함

⑤ 데이터 분포

- 변수의 분포를 파악하여 데이터가 어떻게 구성되어 있는지 이해

⑥ 중복값

- 동일한 행이 여러 번 반복되는지 확인

데이터의 기본 구조 파악

- 데이터 기본 구조를 파악하는 방법

① 데이터 준비

```
import pandas as pd

# 샘플 데이터
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emma'],
    'Age': [25, 30, None, 35, 29],
    'Salary': [50000, 60000, 75000, 80000, None],
    'Department': ['HR', 'IT', 'Finance', 'IT', 'HR']
}

df = pd.DataFrame(data)
```

데이터의 기본 구조 파악

- 데이터 기본 구조를 파악하는 방법

② 데이터셋의 크기 확인

```
# 데이터 크기 확인
print(df.shape)
```

```
(5, 4) (5행, 4열)
```

③ 데이터의 기본 정보 확인

```
# 데이터 타입 및 결측값 확인
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Name        5 non-null      object
 1   Age         4 non-null      float64
 2   Salary      4 non-null      float64
 3   Department  5 non-null      object
```

데이터의 기본 구조 파악

- 데이터 기본 구조를 파악하는 방법

④ 통계적 요약

```
# 수치형 데이터 통계 요약
print(df.describe())
```

	Age	Salary
count	4.000000	4.000000
mean	29.750000	66250.000000
std	4.349329	13726.217763
min	25.000000	50000.000000
25%	28.500000	57500.000000
50%	29.500000	67500.000000
75%	30.750000	76250.000000
max	35.000000	80000.000000

⑤ 결측값 확인

```
# 결측값 확인
print(df.isnull().sum())
```

```
Name          0
Age           1
Salary        1
Department    0
```

데이터의 기본 구조 파악

- 데이터 기본 구조를 파악하는 방법

⑥ 데이터 타입 확인

```
# 데이터 타입 확인
print(df.dtypes)
```

```
Name           object
Age           float64
Salary        float64
Department     object
```

⑦ 중복값 확인

```
# 중복값 확인
print(df.duplicated().sum())
```

```
0 (중복값이 없음)
```

데이터의 기본 구조 파악

- **데이터의 기본 구조를 파악한 후 할 수 있는 일**

  - **분석 전략 설정**
    - 각 변수의 타입에 따라 분석 방법을 설정

      예: 수치형 데이터 → 평균, 중앙값 계산.

      범주형 데이터 → 빈도 분석.
  - **데이터 전처리**
    - 결측값 처리: 평균값으로 대체, 삭제, 또는 예측값으로 대체.
    - 데이터 타입 변환: 필요에 따라 숫자를 범주형으로 변환하거나 반대 작업 수행.
  - **데이터 시각화**
    - 변수 간 관계를 시각적으로 탐색

      예: 산점도(scatter plot)로 수치형 데이터 관계 확인.

      막대그래프(bar chart)로 범주형 데이터 분포 확인.

데이터의 기본 구조 파악

- 데이터의 기본 구조를 파악한 후 할 수 있는 일

  – 데이터: 직원 정보

Name	Age	Salary	Department
Alice	25	50000	HR
Bob	30	60000	IT
Charlie	NaN	75000	Finance
David	35	80000	IT
Emma	29	NaN	HR

① **결측값 확인**

  – Age, Salary에서 각각 1개의 결측값 발견.

② **데이터 타입**

  – Age와 Salary는 수치형, Department와 Name은 범주형 데이터.

③ **데이터 크기**

  – 데이터셋은 5개의 행과 4개의 열로 구성.

기초 통계량 계산

- **주요 기초 통계량**

① 중심 경향성 (Central Tendency)

- 데이터의 대표 값을 나타냄
  - **평균 (Mean)**
    : 데이터의 합을 데이터 개수로 나눈 값.
  - **중앙값 (Median)**
    : 데이터를 크기 순으로 정렬했을 때 가운데 위치한 값.
  - **최빈값 (Mode)**
    : 데이터에서 가장 자주 나타나는 값.

기초 통계량 계산

- **주요 기초 통계량**

② 산포도 (Dispersion)

- 데이터가 얼마나 흩어져 있는지 측정
  - **분산 (Variance)**
    : 데이터가 평균으로부터 얼마나 떨어져 있는지 나타내는 값.
  - **표준편차 (Standard Deviation)**
    : 분산의 제곱근으로, 데이터의 흩어짐 정도를 나타냄
  - **범위 (Range)**
    : 데이터의 최대값과 최소값의 차이.

기초 통계량 계산

- **주요 기초 통계량**

③ 데이터 분포 특성

- 데이터 분포의 모양과 특성을 이해
- **왜도 (Skewness, 기울기)**

  : 데이터 분포가 비대칭인지 나타냄

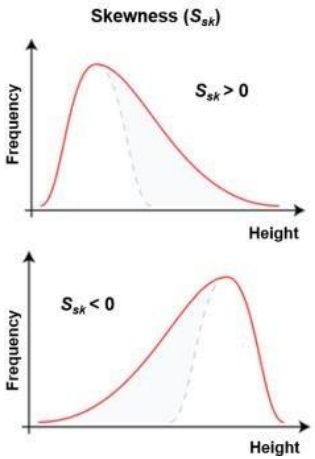
  – 양수: 오른쪽으로 꼬리가 긴 분포.

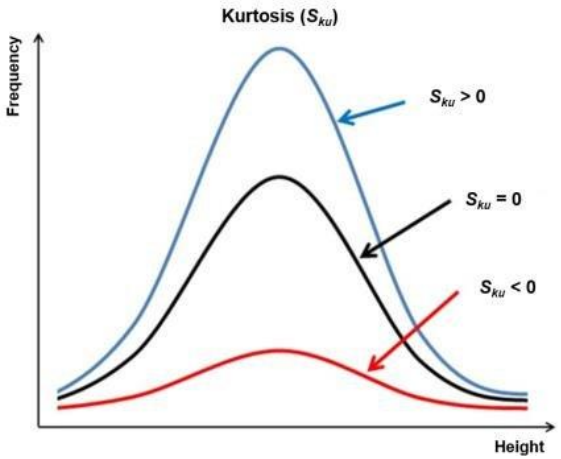
  – 음수: 왼쪽으로 꼬리가 긴 분포.
- **첨도 (Kurtosis)**

  : 분포의 뾰족한 정도를 나타냄

  – 양수: 분포가 뾰족함.

  – 음수: 분포가 평평함.





기초 통계량 계산

- Python을 활용한 기초 통계량 계산

① 데이터 준비

pip install **scipy**

```
import pandas as pd

# 예제 데이터
data = {
    'scores': [85, 90, 78, 92, 88, 76, 95, 89, 77, 85]
}

df = pd.DataFrame(data)
```

기초 통계량 계산

- Python을 활용한 기초 통계량 계산

② 주요 통계량 계산

```
# 평균
mean = df['scores'].mean()
print(f"평균: {mean}")

# 중앙값
median = df['scores'].median()
print(f"중앙값: {median}")

# 최빈값
mode = df['scores'].mode()[0]
print(f"최빈값: {mode}")

# 분산
variance = df['scores'].var()
print(f"분산: {variance}")

# 표준편차
std_dev = df['scores'].std()
print(f"표준편차: {std_dev}")

# 범위
range_value = df['scores'].max() - df['scores'].min()
print(f"범위: {range_value}")
```

기초 통계량 계산

- Python을 활용한 기초 통계량 계산

③ 데이터 분포 특성 계산

```
from scipy.stats import skew, kurtosis

# 왜도
skewness = skew(df['scores'])
print(f"왜곡도: {skewness}")

# 첨도
kurt = kurtosis(df['scores'])
print(f"첨도: {kurt}")
```

기초 통계량 계산

- Python을 활용한 기초 통계량 계산
  - 평균

    : 데이터 전체의 균형점. 예를 들어 시험 점수 평균이 85라면, 대부분의 데이터가 이 근처에 분포한다고 볼 수 있음.
  - 중앙값

    : 극단값(이상치)의 영향을 받지 않는 대표값. 데이터가 치우쳐 있을 때 평균 대신 중앙값을 사용.
  - 최빈값

    : 빈도가 가장 높은 값으로, 특정 값이 얼마나 자주 나타나는지 확인.
  - 분산과 표준편차

    : 데이터가 얼마나 흩어져 있는지 측정. 표준편차가 크면 데이터의 변동성이 크다는 뜻.
  - 왜도

    : 데이터가 왼쪽 또는 오른쪽으로 치우쳤는지 이해.
  - 첨도

    : 데이터의 분포가 평평한지 뾰족한지 나타냄.

first coding

데이터 분석 2 - 그룹화와 집계

그룹별 집계와 피벗 테이블 생성

데이터 간 비교 분석

그룹별 집계와 피벗 테이블 생성

- 그룹별 집계(Grouping and Aggregation)

  - 데이터를 특정 기준으로 묶은 후(그룹화) 각 그룹에 대해 합계, 평균, 최대값 등 다양한 연산(집계)을 수행
    예를 들어, 학교 학생 데이터를 과목별로 그룹화한 뒤 각 과목에서 평균 점수를 계산하는 작업

  - 언제 사용할까?
    - 대량의 데이터를 요약하고 패턴을 파악
    - 데이터를 시각적으로 이해하거나 보고서를 작성
  - 사용 예시
    - 쇼핑몰 매출 데이터를 분석해 카테고리별 총 매출
    - 회사의 직원 데이터를 부서별로 나누어 평균 급여

```
import pandas as pd

# 샘플 데이터
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Department': ['HR', 'IT', 'HR', 'IT', 'Finance'],
    'Salary': [50000, 60000, 55000, 70000, 65000]
}
df = pd.DataFrame(data)
# 그룹화 및 집계: 부서별 평균 급여 계산
grouped = df.groupby('Department')['Salary'].mean()
print(grouped)
```

```
Department
Finance    65000.0
HR         52500.0
IT         65000.0
Name: Salary, dtype: float64
```

그룹별 집계와 피벗 테이블 생성

- 피벗 테이블(Pivot Table)

  - 데이터를 행(row)과 열(column)로 재구성하여 원하는 방식으로 요약하는 테이블
  - 여러 집계 연산을 한 번에 수행할 수 있어 유연성이 뛰어남

  - 언제 사용할까?
    - 다차원 데이터를 요약하거나 여러 기준에서 분석
    - 직관적으로 데이터를 비교하고 분석
  - 사용 예시
    - 매장별 월별 판매 데이터 요약
    - 부서별, 직급별 평균 급여 분석

```
             Salary
Department
Finance     65000.0
HR          52500.0
IT          65000.0
                mean     sum
              Salary  Salary
Department
Finance      65000.0   65000
HR           52500.0  105000
IT           65000.0  130000
```

```
# 같은 데이터프레임을 사용
# 피벗 테이블 생성: 부서별 평균 급여 계산
pivot_table = pd.pivot_table(df,
      values='Salary', index='Department', aggfunc='mean')
print(pivot_table)

pivot_table = pd.pivot_table(df, values='Salary’,
                index='Department', aggfunc=['mean', 'sum'])
print(pivot_table)
```

그룹별 집계와 피벗 테이블 생성

- 피벗 테이블(Pivot Table)

  - 데이터

```
import pandas as pd

data = {
    'City': ['Seoul', 'Seoul', 'Busan', 'Busan',
'Daegu'],
    'Year': [2021, 2022, 2021, 2022, 2021],
    'Sales': [500, 700, 200, 300, 400]
}

df = pd.DataFrame(data)
```

  - 그룹화와 집계

```
grouped = df.groupby('City')['Sales'].sum()
print(grouped)
```

```
City
Busan     500
Daegu     400
Seoul    1200
Name: Sales, dtype: int64
```

  - 피벗 테이블 생성

```
pivot = pd.pivot_table(df, values='Sales',
index='City', columns='Year', aggfunc='sum')
print(pivot)
```

```
Year   2021  2022
City
Busan   200   300
Daegu   400     0
Seoul   500   700
```

데이터 간 비교 분석

- **데이터 간 비교 분석**

  - 두 개 이상의 데이터 집합을 비교하여 차이점, 유사점, 상관관계 등을 분석하는 작업
  - 데이터의 경향이나 패턴을 이해하고 의사 결정을 돕는 데 사용

  - 비교 분석이 중요성
    - 경향 파악

      : 데이터 간 변화를 이해하면 미래를 예측하거나 문제를 해결
    - 효율적 의사 결정

      : 데이터를 비교하면 어떤 선택이 더 효과적인지 판단
    - 실생활 적용

      : 마케팅 성과 비교, 제품 판매량 변화 분석, 지역별 성과 분석 등 다양한 분야에서 활용

데이터 간 비교 분석

- **데이터 비교 분석 방법**

① 기술 통계 비교

- 각 데이터 세트의
  기본 통계(평균, 중간값, 최댓값 등)를
  계산하여 비교

```
import pandas as pd

# 샘플 데이터
data1 = {'Sales': [100, 200, 150, 300]}
data2 = {'Sales': [120, 220, 180, 260]}

df1 = pd.DataFrame(data1, columns=['Sales'])
df2 = pd.DataFrame(data2, columns=['Sales'])

# 기술 통계 계산
print("Dataset 1:\n", df1.describe())
print('-' * 50)
print("Dataset 2:\n", df2.describe())
```

```
Dataset 1:
            Sales
count    4.000000
mean   187.500000
std     85.391256
min    100.000000
25%    137.500000
50%    175.000000
75%    225.000000
max    300.000000
```

```
Dataset 2:
            Sales
count    4.000000
mean   195.000000
std     62.449980
min    120.000000
25%    165.000000
50%    200.000000
75%    225.000000
max    260.000000
```

데이터 간 비교 분석

- **데이터 비교 분석 방법**

② 시각화 비교

- 데이터를 그래프로 표현하여 직관적으로 비교

```
import matplotlib.pyplot as plt

# 데이터
x = ['Q1', 'Q2', 'Q3', 'Q4']
y1 = [100, 200, 150, 300]
y2 = [120, 220, 180, 260]

# 그래프 생성
plt.plot(x, y1, label='Dataset 1', marker='o')
plt.plot(x, y2, label='Dataset 2', marker='o')
plt.title('Quarterly Sales Comparison')
plt.xlabel('Quarter')
plt.ylabel('Sales')
plt.legend()
plt.show()
```

데이터 간 비교 분석

- **데이터 비교 분석 방법**

③ 상관 관계 분석

- 두 데이터 세트가
  서로 어떤 관계를 가지고 있는지 분석
- 상관 계수(Correlation Coefficient)를 사용
- 범위는 -1에서 1까지
  - 1: 완전한 양의 상관 관계
  - 0: 관계 없음
  - -1: 완전한 음의 상관 관계

```
import numpy as np

# 데이터
dataset1 = [100, 200, 150, 300]
dataset2 = [120, 220, 180, 260]

# 데이터 차이 계산
difference = [b - a for a, b in zip(dataset1, dataset2)]
print("Difference between datasets:", difference)
```

```
Difference between datasets: [20, 20, 30, -40]
```

데이터 간 차이를 계산하면 어느 시점에서 변화가 발생했는지 알 수 있음

데이터 간 비교 분석

- **데이터 비교 분석 방법**

④ 차이 분석

- 두 데이터 집합 간
  차이를 계산하여 변화량을 분석

```
import numpy as np

# 데이터
dataset1 = [100, 200, 150, 300]
dataset2 = [120, 220, 180, 260]

# 상관 계수 계산,
# 양의 상관 관계: 키가 커질수록 몸무게가 증가하는 경우.
correlation = np.corrcoef(dataset1, dataset2)[0, 1]
print(f"Correlation Coefficient: {correlation}")
```

```
Correlation Coefficient: 0.993
```

데이터 간 비교 분석

- **데이터 비교 분석 방법**

⑤ 피벗 테이블을 사용한 비교

- 여러 기준에서 데이터를 비교할 때 유용

```
import pandas as pd

# 데이터
data = {
    'Region': ['East', 'East', 'West', 'West', 'East'],
    'Year': [2021, 2022, 2021, 2022, 2021],
    'Sales': [500, 700, 200, 300, 600]
}
df = pd.DataFrame(data)

pivot = pd.pivot_table(df, values='Sales',
index='Region', columns='Year', aggfunc='sum')
print(pivot)
```

```
Year    2021  2022
Region
East    1100   700
West     200   300
```

데이터 간 비교 분석

- **데이터 비교 분석의 활용**

  - 비즈니스 의사 결정
    - 제품별 매출 데이터를 비교하여 어느 제품에 더 많은 자원을 투자할지 결정.
  - 과학 연구
    - 실험 전후 데이터를 비교하여 결과의 유효성 판단.
  - 교육 분석
    - 학급별 성적 비교로 학습 경향 파악.

  - 데이터 간 비교 분석은 다양한 방법(통계, 시각화, 상관 관계, 차이 분석)을 통해 데이터의 차이점과 경향을 파악
  - pandas와 matplotlib는 데이터 비교 분석을 간단하고 효과적으로 수행하는 도구
  - 필요에 따라 다양한 방법을 조합하면 더욱 깊이 있는 분석이 가능

first coding

상관관계와 회귀 분석 개요

상관계수와 상관행렬

단순 회귀 분석 개념

변수 간 관계 분석

상관계수와 상관행렬

- **상관계수**

  - 상관계수는 두 변수 간의 선형 관계의 강도와 방향을 나타내는 값
    - 값의 범위는 -1부터 1까지
      - 1: 두 변수는 완벽한 양의 상관관계 (하나가 증가하면 다른 것도 증가).
      - -1: 두 변수는 완벽한 음의 상관관계 (하나가 증가하면 다른 것은 감소).
      - 0: 두 변수는 상관관계가 없음 (관계가 없거나 비선형적).

상관계수와 상관행렬

- **상관계수**

  - 사례 1
    - 아이스크림 판매량과 기온
      - 여름철에 아이스크림 판매량이 증가한다고 가정
      - 기온이 올라갈수록 사람들이 아이스크림을 더 많이 사 먹음
        - » 기온이 올라갈수록 아이스크림 판매량도 증가 → 양의 상관관계.
        - » 상관계수 값: 0.8 (0.8은 강한 양의 상관관계를 의미).

상관계수와 상관행렬

- **상관계수**
  - 사례 2
    - 우산 판매량과 날씨
      - 비가 많이 올수록 우산 판매량이 늘어난다고 가정
      - 비가 오는 날씨의 빈도와 우산 판매량은 양의 상관관계.
      - 상관계수 값: 0.7.

상관계수와 상관행렬

- **상관계수**

  - 사례 3
    - 공부 시간과 시험 점수
      - 공부 시간이 늘어날수록 시험 점수가 증가할 가능성이 큼
        - » 이 경우 두 변수 간에는 양의 상관관계가 있음
        - » 상관계수 값: 0.9 (매우 강한 양의 상관관계).

상관계수와 상관행렬

- **상관계수**

  - 사례 4
    - 외출 시간과 집에 있는 시간
      - 외출 시간이 늘어나면 집에 있는 시간은 줄어듦
        - » 이 경우 두 변수는 음의 상관관계.
        - » 상관계수 값: -0.6 (중간 정도의 음의 상관관계).

상관계수와 상관행렬

- 상관계수
  - Perfect Positive (r=1)

: 데이터 점들이 완벽히 직선 위에 놓여 있고,
x와 y가 동일하게 증가

  - Strong Positive (r=0.8)

: 데이터 점들이 양의 관계를 나타내며, 약간의 분산이 있음

  - No Correlation (r=0)

: 데이터 점들이 무작위로 흩어져 있어 관계를 찾을 수 없음

  - Strong Negative (r=-0.8)

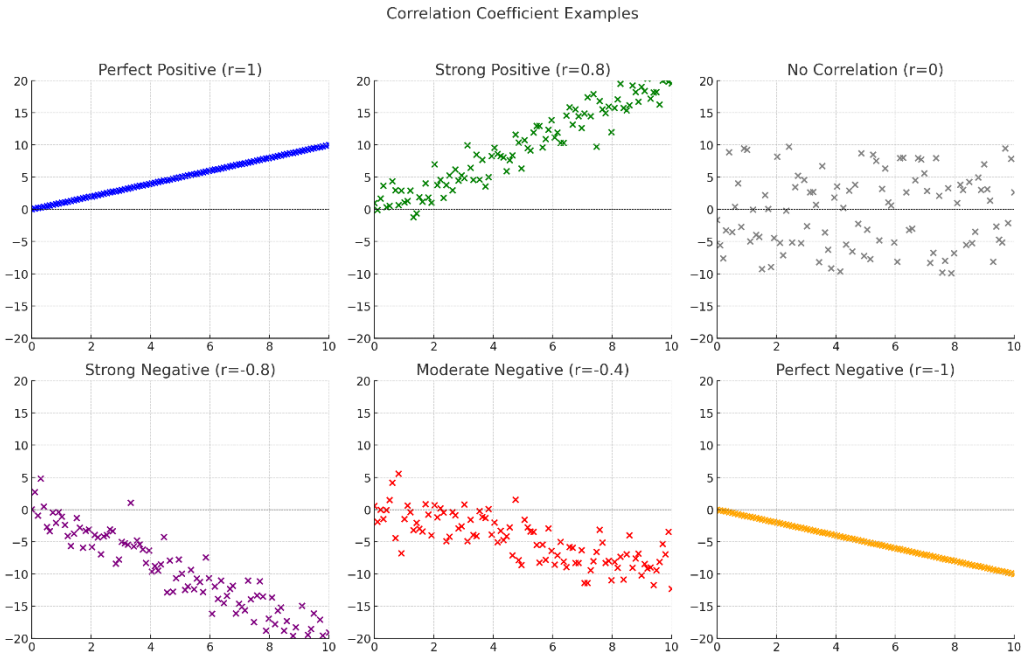
: 데이터 점들이 음의 관계를 나타내며, 약간의 분산이 있음

  - Moderate Negative (r=-0.4)

: 데이터 점들이 약한 음의 관계를 나타냄

  - Perfect Negative (r=-1)

: 데이터 점들이 완벽히 직선 위에 있으며, x가 증가하면 y가 동일하게 감소함



상관계수와 상관행렬

- **상관행렬**

  - 상관행렬은 여러 변수 간의 상관계수를 표 형태로 보여줌
  - 예시

    : 학생 데이터 학생들의 학습 데이터를 분석한다고 가정

    - 변수는 다음과 같다 :
      - **공부 시간**
      - **시험 점수**
      - **잠자는 시간**
      - **SNS 사용 시간**

구분	공부 시간	시험 점수	잠자는 시간	SNS 사용 시간
공부 시간	1	0.85	-0.2	-0.6
시험 점수	0.85	1	-0.15	-0.55
잠자는 시간	-0.2	-0.15	1	0.1
SNS 사용 시간	0.6	-0.55	0.1	1

상관계수와 상관행렬

구분	공부 시간	시험 점수	잠자는 시간	SNS 사용 시간
공부 시간	1	0.85	-0.2	-0.6
시험 점수	0.85	1	-0.15	-0.55
잠자는 시간	-0.2	-0.15	1	0.1
SNS 사용 시간	0.6	-0.55	0.1	1

- **공부 시간과 시험 점수**
  - 상관계수

    0.85 → 강한 양의 상관관계

    (공부를 많이 하면 시험 점수가 높아짐).
- **공부 시간과 SNS 사용 시간**
  - 상관계수

    -0.6 → 음의 상관관계

    (공부 시간이 많을수록 SNS 사용 시간은 줄어듦).
- **잠자는 시간과 SNS 사용 시간**
  - 상관계수

    0.1 → 거의 상관관계가 없음

    (두 변수는 관계가 약함).

단순 회귀 분석 개념

- **단순 회귀 분석**

  - 하나의 독립 변수(X)와 하나의 종속 변수(Y) 사이의 관계를 모델링하여, X를 이용해 Y를 예측하는 방법
    - 데이터가 선형적 관계를 가지고 있을 때 유용
    - 결과는 다음과 같은 1차 방정식으로 표현

$$Y = aX + b$$

- a: 기울기 (독립 변수가 1 증가할 때 종속 변수가 얼마나 증가하는지).
- b: 절편 (독립 변수가 0일 때 종속 변수의 예상 값).

단순 회귀 분석 개념

- 단순 회귀 분석

  - 사례
    - 공부 시간과 시험 점수학생들이 하루에 공부한 시간을 바탕으로 시험 점수를 예측한다고 가정

공부 시간 (X)	시험 점수 (Y)
1시간	50점
2시간	55점
3시간	65점
4시간	70점
5시간	75점

- 이 데이터를 바탕으로 회귀 분석을 수행하면 다음과 같은 회귀식이 나옴

→ $Y = 5X + 4$

- 해석

  - 공부 시간을 1시간 늘리면 시험 점수가 5점 증가
  - 공부를 전혀 안 하면 시험 점수는 45점일 것으로 예상됨

단순 회귀 분석 개념

- **단순 회귀 분석**

  - 사례
    - 회사가 광고에 투자한 금액과 그로 인해 발생한 매출 간의 관계를 분석한다고 가정

광고비 (X, 만 원)	매출 (Y, 만 원)
10	100
20	150
30	200
40	250
50	300

- 이 데이터를 바탕으로 회귀 분석을 수행하면 다음과 같은 회귀식이 나옴

→ $Y = 5X + 50$

- 해석

  - 광고비를 10만 원 더 쓰면 매출이 50만 원 증가
  - 광고비가 0일 때 매출은 50만 원으로 예측

단순 회귀 분석 개념

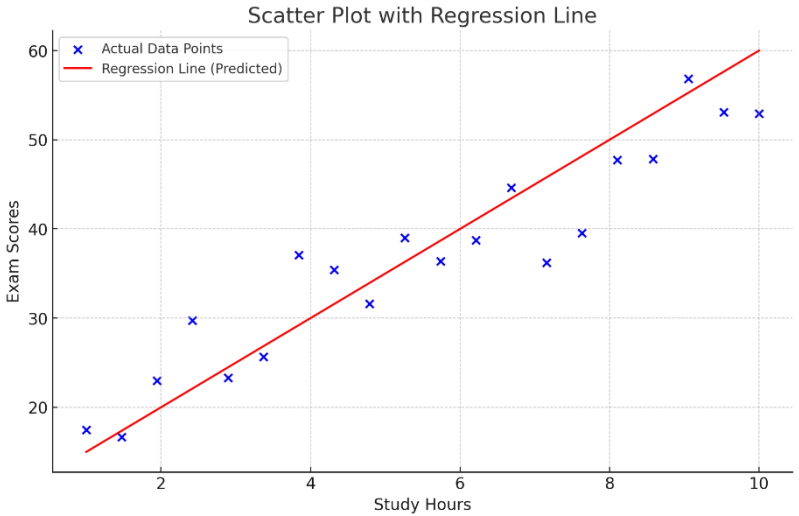
- **회귀 분석의 주요 개념**

  - 잔차 (Residuals)
    - 실제 값과 회귀선(예측 값) 간의 차이를 의미
    - 예) 한 학생이 3시간 공부했는데 실제 시험 점수는 60점이었지만 예측 값은 65점이었다면
      → 잔차는 -5
  - 최소제곱법 (Least Squares Method)
    - 잔차의 제곱합이 최소가 되도록 회귀선을 찾는 방법
    - 잔차 제곱합이 최소일 때 가장 적합한 선형 방정식을 얻을 수 있음
  - R^2 값 (결정 계수)
    - 회귀 모델이 데이터를 얼마나 잘 설명하는지를 나타냄
    - 값의 범위는 0~1이며, 1에 가까울수록 회귀 모델이 데이터를 잘 설명

단순 회귀 분석 개념

- **회귀 분석의 주요 개념**

  - 시각적 이해
    - 데이터를 산점도로 표현하고, 그 위에 회귀선을 추가
    - 예를 들어, 공부 시간 vs. 시험 점수를 시각화
      - x-축: 공부 시간
      - y-축: 시험 점수
      - 데이터 점들 사이에 회귀선이 그려져 있으며, 이 선이 예측 모델을 나타냄



변수 간 관계 분석

- **변수 간 관계 분석**

  - 두 변수 또는 여러 변수 간의 상관성이나 영향력을 이해하고,
  - 한 변수가 다른 변수에 어떻게 영향을 미치는지를 분석하는 과정
  - 관계는 상관 관계와 인과 관계로 나눌 수 있음
    - **상관 관계**

      : 두 변수 간의 변화가 함께 나타나는 경우 (서로 연관성이 있음).
    - **인과 관계**

      : 한 변수의 변화가 다른 변수에 직접적인 영향을 미치는 경우.

- **변수 간 관계 분석**

  - 상관 관계와 인과 관계의 차이
    - 상관 관계는 인과 관계가 아닐 수 있음!
    - 상관 관계는 단지 두 변수 간에 규칙적으로 변하는 패턴이 있음을 나타냄.
    - 예:

      아이스크림 판매량과 에어컨 판매량의 상관 계수는 양의 관계를 보일 수 있지만,

      이는 인과 관계가 아님.

      (둘 다 여름철과 관련이 있음)

변수 간 관계 분석

- **변수 간 관계 분석**

  - 사례 1: 공부 시간과 시험 점수
    - 질문: 학생의 공부 시간이 시험 점수에 어떤 영향을 미칠까?

공부 시간 (X)	시험 점수 (Y)
1시간	50점
2시간	55점
3시간	65점
4시간	70점
5시간	75점

- **상관 관계**
  공부 시간이 증가할수록 시험 점수가 증가
  (양의 상관 관계)

- **인과 관계**
  공부 시간이 직접 시험 점수에 영향을 미침
  (인과 관계 존재)

- **변수 간 관계 분석**

  - 사례 2: 광고비와 매출
    - 질문: 광고비를 늘리면 매출이 늘어날까?

광고비 (X, 만 원)	매출 (Y, 만 원)
10	100
20	150
30	200
40	250
50	300

- **상관 관계**
  광고비와 매출 간에는 **강한 양의 상관 관계**가 존재.

- **인과 관계**
  광고비를 늘리면 매출이 증가한다는 인과 관계가 존재.

- **변수 간 관계 분석**

  - 사례 3: 키와 체중
    - 질문: 사람의 키와 체중 간에는 어떤 관계가 있을까?

키 (cm)	체중 (kg)
150	50
160	55
170	65
180	75
190	85

- **상관 관계**
  키가 클수록 체중이 늘어나는 경향
  (**양의 상관 관계**).

- **인과 관계**
  키와 체중 간에는 **직접적인 인과 관계는 없음**,
  단지 체형이나 유전적 요인이 영향을 줄 수 있음.

- **변수 간 관계 분석**

  - 사례 4: 우산 판매량과 날씨
    - 질문: 날씨가 우산 판매량에 영향을 미칠까?

강우량 (X, mm)	우산 판매량 (Y)
0	10
10	20
20	40
30	60
40	80

- **상관 관계**
  강우량과 우산 판매량 간에는 강한 양의 상관 관계.

- **인과 관계**
  강우량이 많아지면 우산 판매량이 증가하는 **직접적인 인과 관계** 존재.

변수 간 관계 분석

- 변수 간 관계 분석

  - 관계 분석의 도구
    - 상관 계수
      - 두 변수 간의 관계를 수치화 하여 -1에서 1 사이의 값으로 표현.
      - 예: 공부 시간과 시험 점수의 상관 계수가 0.85라면, 두 변수 간에는 강한 양의 상관 관계가 존재.
    - 산점도
      - 두 변수의 데이터를 시각화 하여 관계를 직관적으로 이해.
      - 예: 광고비와 매출의 데이터를 산점도로 표현하면, 데이터가 선형적으로 증가하는 모습을 확인 가능.
    - 회귀 분석
      - 두 변수 간 관계를 선형 방정식으로 나타내어 한 변수를 기반으로 다른 변수를 예측.
      - 예: 광고비(X)와 매출(Y) 간 회귀식이 $Y=5X+50$라면, 광고비 10만 원을 더 쓰면 매출이 50만 원 증가.